Conferences in Research and Practice in
Information Technology

Volume 49

Database Technologies 2006

Australian Computer Science Communications, Volume 28, Number 2.

AUSTRALIAN
COMPUTER
SOCIETY

acsw

COmputing
Research
& Education

Database Technologies 2006

Proceedings of the
17th Australasian Database Conference (ADC2006),
Hobart, Tasmania, Australia, 16-19 January 2006

Gillian Dobbie and James Bailey, Eds.

Volume 49 in the Conferences in Research and Practice in Information Technology Series.
Published by the Australian Computer Society Inc.

Published in association with the ACM Digital Library.

Proceedings of the Seventeenth Australasian Database Conference (ADC2006), Hobart, Tasmania, Australia, 16-19 January 2006

Conferences in Research and Practice in Information Technology, Volume 49.

Copyright ©2006, Australian Computer Society. Reproduction for academic, not-for profit purposes permitted provided the copyright text at the foot of the first page of each paper is included.

Editors: Gillian Dobbie
Department of Computer Science
University of Auckland
Auckland,
New Zealand
Email: gill@cs.auckland.ac.nz

James Bailey
Department of Computer Science and Software Engineering
University of Melbourne
Melbourne,
Australia
Email: jbailey@cs.mu.oz.au

Series Editor: John F. Roddick,
Conferences in Research and Practice in Information Technology
Flinders University,
PO Box 2100, Adelaide 5001
South Australia.
crpit@infoeng.flinders.edu.au

Publisher: Australian Computer Society Inc.
PO Box Q534, QVB Post Office
Sydney 1230
New South Wales
Australia.

Conferences in Research and Practice in Information Technology, Volume 49.
ISSN 1445-1336.
ISBN 1-920-68231-7.

Printed, November 2005 by Flinders Press, PO Box 2100, Bedford Park, SA 5042, South Australia.
Cover Design by Modern Planet Design, (08) 8340 1361.

The *Conferences in Research and Practice in Information Technology* series aims to disseminate the results of peer-reviewed research in all areas of Information Technology. Further details can be found at http://crpit.com/.

Table of Contents

Proceedings of the Seventeenth Australasian Database Conference (ADC2006), Hobart, Tasmania, Australia, 16-19 January 2006

Keynote Paper

Invited Paper

Full Papers

Advanced Foundations of Databases

Data Mining and Knowledge Discovery

XML and Databases

Information Retrieval

Query Processing and Optimization

Advanced Database Applications

Performance Issues in Databases

Preface

The Australasian Database Conference (ADC) series is an annual forum, exploring research, development and novel applications of databases systems. This volume contains papers presented at the Seventeenth ADC in Hobart, Australia. ADC 2006 is a specialist conference in the Australasian Computer Science Week which ran from Jan 16th to 19th, 2006.

The ADC 2006 call for papers solicited contributions in all areas of database research. This years conference is truly international with papers submitted from Australia (32), New Zealand (8), Canada (3), USA (3), Denmark (2), Japan (2), France (1), India (1), Pakistan (1), Thailand (1), and UK (1).

The topics addressed by the submitted papers illustrate the broadness of the database discipline. The authors categorised their submissions into one or more of the following topics:

- Data mining/knowledge discovery (14 papers)
- Information retrieval (14 papers)
- XML and databases (11 papers)
- Query processing and optimization (9 papers)
- Advanced foundations of databases (8 papers)
- Advanced database applications (7 papers)
- Semistructured data (7 papers)
- Logic in databases (6 papers)
- Data warehousing (5 papers)
- Database system integration issues (5 papers)
- Federated, distributed and parallel databases (5 papers)
- High dimensional and temporal data (5 papers)
- Performance issues of databases (4 papers)
- Web information systems (4 papers)
- Database schema integration (3 papers)
- Databases for bioinformatics (3 papers)
- Extended data type management (3 papers)
- Transaction processing (3 papers)
- Image/video retrieval and databases (2 papers)
- Query languages (2 papers)
- Spatial data processing and management (2 papers)

All papers were sent to at least three programme committee members for review and every effort was made to obtain as many reviews as possible. Of the 55 papers submitted, 19 were selected for presentation at the conference. The programme committee invited Professor James Bezdek to give a keynote on *Approximate Clustering for Data Mining in Very Large Databases*. Professor Bezdek is the William Craig Nystul Professor of Computer Science at the University of West Florida. The committee also invited Dr Andrew Turpin to present *Suffix Arrays: What Are They Good For?*. Dr Turpin is a Queen Elizabeth II Fellow at RMIT University.

We thank all authors who submitted papers and all conference participants for helping to make the conference a success. We also thank the members of the programme committee and the external referees for their expertise in carefully reviewing the papers. We are grateful to Sharon Liu and Scott Lee for their work in managing the reviewing system and processes. Last, we express our gratitude to our hosts in Tasmania.

Gillian Dobbie
University of Auckland
James Bailey
University of Melbourne
ADC 2006 Program Chairs
January, 2006

Programme Committee

Chairs

Gillian Dobbie, University of Auckland, New Zealand
James Bailey, University of Melbourne, Australia

Members

Stijn Dekeyser, University of Southern Queensland, Australia
David Edmond, QUT, Australia
Raj P Gopalan, Curtin University of Technology, Australia
Guido Governatori, University of Queensland, Australia
Sven Hartmann, Massey University, New Zealand
David Hawking, CSIRO Canberra, Australia
Annika Hinze, University of Waikato, New Zealand
Patrick Hung, University of Ontario Institute of Technology, Canada
Hasan Jamil, Wayne State University, USA
Beng Chin Ooi, National University of Singapore
John Roddick, Flinders University, Australia
Klaus-Dieter Schewe, Massey University, New Zealand
John Shepherd, University of New South Wales, Australia
Markus Stumptner, University of South Australia, Australia
Saied Tahaghoghi, RMIT University, Australia
Kian-Lee Tan, National University of Singapore
Egemen Tanin, University of Melbourne, Australia
Rodney Topor, Griffith University, Australia
Andrew Turpin, RMIT University, Australia
Wei Wang, University of New South Wales, Australia
Gerald Weber, University of Auckland, New Zealand
Hugh E. Williams, Microsoft Corporation, USA
Raymond Wong, University of New South Wales, Australia
Jeffrey Yu, Chinese University of Hong Kong
Yanchun Zhang, Victoria University, New Zealand
Xiaofang Zhou, University of Queensland, Australia

Organising Committee

Welcome

On behalf of the Tasmanian Organising Committee of ACSW2006 I would like to welcome all the delegates to the conferences of this busy and interesting week, in particular those coming from overseas.

The location of the various conferences and other events at the Wrest Point Hotel allows delegates to move quickly from event to event, and to easily and comfortably gather in groups for those conversations and interactions that are so important for the exchange of ideas and the promotion of cooperation, not to mention social pleasure.

We trust you will have a thoroughly enjoyable time.

Professor Young Ju Choi
Chair, Organising Committee
January, 2006

General Chair

Professor Young Ju Choi, School of Computing, University of Tasmania, Australia

Organising Committee Members

Ms Nicole Clark
Dr Julian Dermoudy
Mr Tony Gray
Mr Neville Holmes
Mr Ian McMahon
Ms Julia Mollison
Professor Arthur Sale
Ms Soon-ja Yeom

CORE - Computing Research and Education

CORE welcomes all delegates to ACSW2006 in Hobart.

ACSW, the Australasian Computer Science Week continues to grow with new conferences becoming entrenched in the week. As the premier annual Computer Science event in Australia and New Zealand, it provides an unparalleled opportunity for the wide community of Computer Science academics and researchers to meet, network, promote IT research and be exposed to the latest research in other areas of IT. The research presented at each conference is of the highest standard and essential for the growth and future of our region, in an ever more competitive world.

CORE is expanding its awards. The Distinguished Service Award first offered in late 2004 will be offered every second year and next at the 2007 Conference. Along with the Chris Wallace Research Award, we are offering an annual teaching award for the first time.

CORE has continued to play a part in the Federation of Australian Scientific and Technological Societies and by participating in events such as Science Meets Parliament, CORE is becoming recognised by the wider community and will continue to do so. A major contribution from many members in 2005 was a submission to the RQF Forum with some of our ideas appearing in the draft. CORE and members of the Executive have also been interviewed as representatives of the Computer Science community for several other Government and industry inquiries and initiatives.

Thank you all for your contributions in 2005 and we look forward to an exciting 2006.

Jenny Edwards
President, Computing Research and Education
January, 2006

ACSW Conferences and the Australian Computer Science Communications

The Australasian Computer Science Week of conferences has been running in some form continuously since 1978. This makes it one of the longest running conferences in computer science. The proceedings of the week have been published as the *Australian Computer Science Communications* since 1979 (with the 1978 proceedings often referred to as *Volume 0*). Thus the sequence number of the Australasian Computer Science Conference is always one greater than the volume of the Communications. Below is a list of the conferences, their locations and hosts.

2008. Communications Volume Number 30. Proposed Host and Venue - University of Wollongong, NSW.
2007. Volume 29. Host and Venue - University of Ballarat, VIC.
2006. Volume 28. Host and Venue - University of Tasmania, TAS.
2005. Volume 27. Host - University of Newcastle, NSW. APBC held separately from 2005.
2004. Volume 26. Host and Venue - University of Otago, Dunedin, New Zealand. First running of APCCM.
2003. Volume 25. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Adelaide Convention Centre, Adelaide, SA. First running of APBC. Incorporation of ACE. ACSAC held separately from 2003.
2002. Volume 24. Host and Venue - Monash University, Melbourne, VIC.
2001. Volume 23. Hosts - Bond University and Griffith University (Gold Coast). Venue - Gold Coast, QLD.
2000. Volume 22. Hosts - Australian National University and University of Canberra. Venue - ANU, Canberra, ACT. First running of AUIC.
1999. Volume 21. Host and Venue - University of Auckland, New Zealand.
1998. Volume 20. Hosts - University of Western Australia, Murdoch University, Edith Cowan University and Curtin University. Venue - Perth, WA.
1997. Volume 19. Hosts - Macquarie University and University of Technology, Sydney. Venue - Sydney, NSW. ADC held with DASFAA (rather than ACSW) in 1997.
1996. Volume 18. Host - University of Melbourne and RMIT University. Venue - Melbourne, Australia. CATS joins ACSW.
1995. Volume 17. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Glenelg, SA.
1994. Volume 16. Host and Venue - University of Canterbury, Christchurch, New Zealand. CATS run for the first time separately in Sydney.
1993. Volume 15. Hosts - Griffith University and Queensland University of Technology. Venue - Nathan, QLD.
1992. Volume 14. Host and Venue - University of Tasmania, TAS. (ADC held separately at La Trobe University).
1991. Volume 13. Host and Venue - University of New South Wales, NSW.
1990. Volume 12. Host and Venue - Monash University, Melbourne, VIC. Joined by Database and Information Systems Conference which in 1992 became ADC (which stayed with ACSW) and ACIS (which now operates independently).
1989. Volume 11. Host and Venue - University of Wollongong, NSW.
1988. Volume 10. Host and Venue - University of Queensland, QLD.
1987. Volume 9. Host and Venue - Deakin University, VIC.
1986. Volume 8. Host and Venue - Australian National University, Canberra, ACT.
1985. Volume 7. Hosts - University of Melbourne and Monash University. Venue - Melbourne, VIC.
1984. Volume 6. Host and Venue - University of Adelaide, SA.
1983. Volume 5. Host and Venue - University of Sydney, NSW.
1982. Volume 4. Host and Venue - University of Western Australia, WA.
1981. Volume 3. Host and Venue - University of Queensland, QLD.
1980. Volume 2. Host and Venue - Australian National University, Canberra, ACT.
1979. Volume 1. Host and Venue - University of Tasmania, TAS.
1978. Volume 0. Host and Venue - University of New South Wales, NSW.

Conference Acronyms

ACE. Australian/Australasian Conference on Computing Education.
ACSAC. Asia-Pacific Computer Systems Architecture Conference (previously Australian Computer Architecture Conference (ACAC).
ACSC. Australian/Australasian Computer Science Conference.
ACSW. Australian/Australasian Computer Science Week.
ADC. Australian/Australasian Database Conference.
APBC. Asia-Pacific Bioinformatics Conference.
APCCM. Asia-Pacific Conference on Conceptual Modelling.
AUIC. Australian/Australasian User Interface Conference.
CATS. Computing - The Australian/Australasian Theory Symposium.

Note that various name changes have occurred, most notably the change of the names of conferences to reflect a wider geographical area.

ACSW and ADC 2006 Sponsors

We wish to thank the following sponsors for their contribution towards this conference. For an up-to-date overview of sponsors of ACSW 2006 and ADC 2006, please see http://www.comp.utas.edu.au/acsw06/.

UTAS

University of Tasmania, Australia

AUSTRALIAN
COMPUTER
SOCIETY

Australian Computer Society

COmputing
Research
& Education

CORE - Computing Research and Education

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

University of Auckland, New Zealand

THE UNIVERSITY OF
MELBOURNE

University of Melbourne, Australia

KEYNOTE PAPER

Approximate Data Mining in Very Large Relational Data

James C. Bezdek[1], Richard J. Hathaway[2],
Christopher Leckie[3], Ramamohanarao Kotagiri[3]

[1]Department of Computer Science, University of West Florida, Pensacola, FL 32514, USA
[2]Department of Mathematical Sciences, Georgia Southern University, Statesboro, GA 30460, USA
[3]Department of Computer Science and Software Engineering, University of Melbourne, Victoria, 3010, Australia
jbezdek@uwf.edu, r.hathaway@ieee.org, caleckie@cs.mu.oz.au, rao@cs.mu.oz.au

Abstract

In this paper we discuss **eNERF**, an extended version of *non-Euclidean relational fuzzy c-means* (**NERFCM**) for approximate clustering in very large (unloadable) relational data. The **eNERF** procedure consists of four parts: (i) selection of distinguished features by algorithm **DF** to be monitored during progressive sampling; (ii) progressively sampling a square $N \times N$ relation matrix R_N by algorithm **PS** until an $n \times n$ sample relation R_n passes a goodness of fit test; (iii) Clustering R_n using algorithm **LNERF**; and (iv), extension of the LNERF results to R_N-R_n by algorithm **xNERF**, which uses an iterative procedure based on LNERF to compute fuzzy membership values for all of the objects remaining after LNERF clustering of the accepted sample. Three of the four algorithms are new - only LNERF (called NERFCM in the original literature) precedes this article.

Keywords: Cluster analysis, data mining, very large data, non-Euclidean relational fuzzy c-means, progressive sampling, relational data, gene product similarities.

1 Introduction

According to Huber (1996), who defines large data sets as an order of magnitude of 10^8 bytes, "Some simple standard database management tasks with computational complexity $\boldsymbol{O}$(n) or $\boldsymbol{O}$(nlogn) remain feasible beyond terabyte monster sets, while others (e.g., clustering) blow up already near large data sets." The next generation of clustering algorithms must not "blow up" when handling large or even very large data (>> 10^{12} bytes).

Consider a set of N objects $\{o_1,...,o_N\}$. Numerical *object* data has the form $X = \{\mathbf{x}_1,..., \mathbf{x}_N\} \subset \Re^s$, where the coordinates of $\mathbf{x}_i$ provide feature values (e.g., weight, length, etc.) describing object o_i. The other type of data commonly found in data mining is numerical *relational* data, which consists of N^2 pair-wise dissimilarities (or similarities), represented by a matrix $D = [d_{ij} = \text{dissimilarity}(o_i, o_j) \mid 1 \le i, j \le N]$. Many clustering algorithms are known and used for both kinds of data (Bezdek et al., 1999). X can be converted into *dissimilarity* data D = D(X) by computing $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ in any vector norm on $\Re^s$, so most relational clustering algorithms are (implicitly) applicable to object data. However, there are both similarity and dissimilarity relational data sets that do not begin as object data, and for these, we have no choice but to use a relational algorithm. In many application domains, a relational representation may reflect the way the data is collected and stored. Consider the problems of clustering actors based on whether they have performed together in the same film, or documents based on whether they have similar word usage (Tasker et al., 2001). In each case, a natural representation for the data is as a binary relation that specifies the similarity between objects. A similar problem arises in marketing, when trying to group product lines that are frequently purchased together. Rather than storing every purchasing transaction, a more efficient representation is to simply record the frequency with which individual pairs of products have been purchased together.

Many algorithms have been proposed for clustering in VL object data (Ng and Han, 1994, Bradley et al. 1998, Ganti et al. 1999, Hathaway and Bezdek, 2005), but no algorithms (that we are aware of) exist for "pure" relational data (i.e., $D \neq D(X)$ for some X). One way to attack the problem of clustering in VL data is discussed in Pal and Bezdek (2002), where the technical notion of *extensibility* is introduced. Roughly speaking, an *extended clustering scheme* applies a clustering algorithm to a (loadable) sample of the full data set, and then non-iteratively extends the sample result to obtain (approximate) clusters for the remaining data. A *literal* scheme applies the clustering algorithm without modification to the full data set.

When the data set is very large (VL), sampling and extension makes clustering feasible for cases where it is not otherwise possible. If the data set is merely large (L), but still loadable, then an extended scheme may offer an approximate solution comparable to the literal solution at a significantly reduced computational cost - in other words, it accelerates the corresponding literal scheme. The benefits for the two cases can be summarized as *feasibility* for VL data sets and *acceleration* for L data sets. Both situations are depicted in Figure 1 where the data set to be clustered is either D_L or D_{VL}.

Copyright (c) 2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

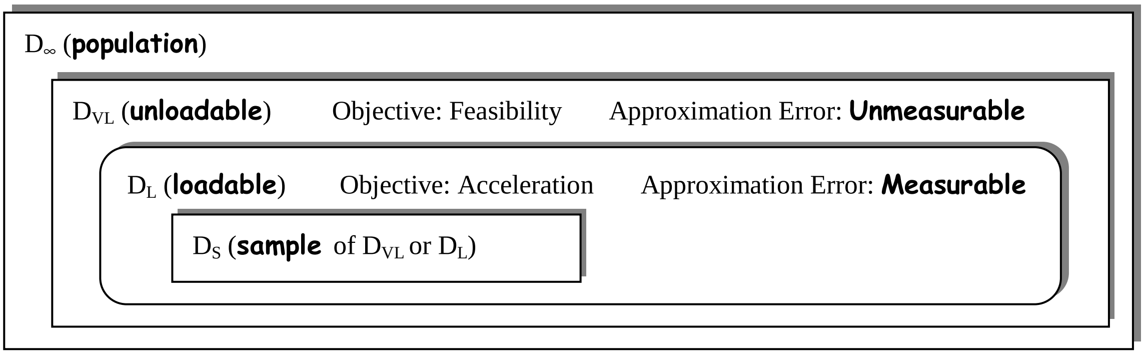

D_∞ (population)
D_{VL} (unloadable) Objective: Feasibility Approximation Error: Unmeasurable
D_L (loadable) Objective: Acceleration Approximation Error: Measurable
D_S (sample of D_{VL} or D_L)

Figure 1: Population D_∞ and samples D_{VL}, D_L, D_{SS}

Our test for judging whether D_S is representative of the source may require some basic processing of the full sample. We can load D_L into primary memory and do the required processing (to test the sample). We cannot load D_{VL}, but may have to page through D_{VL} *once* to gather simple statistics (e.g., bin counts for a histogram) needed to assess the quality of candidate samples.

For D_L, we can assess the approximation error by measuring the difference between the clustering solutions obtained using the corresponding extended and literal schemes. On the other hand, the *only* solution generally available for D_{VL} is that obtained by the extended scheme, in which case the approximation error cannot be measured. Thus, our confidence in the accuracy of extended clusters in the unverifiable case (D_{VL}) is necessarily derived from the verified good behavior we can observe by conducting various D_L experiments.

2 eNERF

2.1 Labels and Partitions. Let c be the number of classes, $1 < c < n$. *Crisp label vectors* in $\Re^c$ look like $\mathbf{y}_i = (0,\ldots,1,\ldots,0)^T$, with a 1 in the ith place meaning that objects with this label belong to class i. Fuzzy or probabilistic label vectors ("soft" labels) look like y = $(0.1,\ 0.6,\ 0.3)^T$ (where c=3 classes, for example); they have entries in [0, 1] that sum to 1. We need names for the sets of all soft and hard label vectors, so we follow the usual notation:

$$N_{fc} = \{\mathbf{y} = (y_1,\ldots y_c)^T \in \Re^c : \Sigma\, y_i = 1; 0 \le y_i \le 1 \forall\, i\};$$

$$N_{hc} = \{\mathbf{y} \in N_{fc} : y_i \in \{0,1\} \forall\, i\} \subset N_{fc}.$$

Assume that $D_N = D = [d_{ij}]$ satisfies, for $1 \le i, j \le N$,

$$d_{ij} \ \ge\ 0;\ d_{ij} = d_{ji};\ d_{ii} = 0. \qquad (1)$$

We do *not* assume that D is transitive, because this restriction simply does not hold for most real VL relational data. Clustering in O (or somewhat sloppily, in D) is the assignment of (hard or fuzzy or probabilistic) label vectors to the objects in O. A c-partition of O (or D) is a set of (cN) values $\{u_{ik}\}$ arrayed as a $c \times N$ matrix U = $[\mathbf{U}^1\ \mathbf{U}^2\ \ldots\ \mathbf{U}^N] = [u_{ik}]$, where $\mathbf{U}^k$, *the* k^{th} *column of U*, is the label vector in N_{fc} for o_k. The ikth element u_{ik} of U is the membership of o_k in cluster i. Each column of U sums to 1, and each row of U must have at least one non-zero entry.

Table 1 contains an example of crisp (U_1) and soft (U_2) 2-partitions of n=6 objects. U_1 identifies 3 objects in each of the two crisp clusters. Objects have partial memberships (or posterior probabilities) in U_2. For example, the first object is more similar to class 1 ($u_{2,11}$= 0.7) than to class 2 ($u_{2,21}$=0.3). Because fuzzy partitions may indicate partial memberships in several clusters, they often provide valuable information about *twixters* (in-between objects) that is not available in a crisp partition. The most visible twixters in U_2 are objects 3 and 4. When we want crisp labels from soft partitions, the usual method of "hardening" is to simply replace the maximum entry in each column of U by a 1, and place 0's in the remaining (c-1) columns (this is just Bayes rule when U is a probabilistic partition). We denote the *hardening* of U by H(U). The last column in Table 1 has an example for the hardening of U_2. If we compare $H(U_2)$ to U_1, we see that there is one *mismatch* (object 3). This is not necessarily an *error* – just a mismatch (it *would* be an error if U_1 contained ground truth labels – i.e., labels known to be right).

Table 1: Crisp, Soft and Hardened 2-partitions of n objects

Crisp	Soft	Hardened U_2
$U_1 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$	$U_2 = \begin{pmatrix} 0\ .7 & 0.2 & 0.45 & 0.4 & 0.3 & 0.9 \\ 0.3 & 0.8 & 0.55 & 0.6 & 0.7 & 0.1 \end{pmatrix}$	$H(U_2) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$

2.2 Distinguished Features. We want to cluster a sample of n objects O_n from the VL set O_N, or equivalently, we want clusters in a submatrix D_n of D_N. The progressive sampling scheme we have in mind is an adaptation of the object data scheme in Hathaway and Bezdek (2005). To use this approach we interpret the relational data as object data by regarding the i^{th} column $\mathbf{D}^i$ of D_N (equivalently the i^{th} row, since D_N is symmetric) as a feature vector for object o_i. $D_N = [\mathbf{D}^1\ \mathbf{D}^2 \ldots \mathbf{D}^N]$. Any road map that has a table of distances between pairs of cities on the map is a relation of this type. This interpretation of $\mathbf{D}^i$ is analogous to giving the location of city i (= object o_i) by specifying - instead of, say, its rectangular coordinates - its distance to N-1 other cities (i.e., the N-1 distances between city i and the other cities on the map are its "features").

Which of these N "feature vectors" should we use? We believe it is reasonable to pick *distinguished features* (DFs) that are *very different* from each other. We choose h, the number of distinguished features to be selected, and H, which restricts the distinguished features to come only from rows 1 through H of D. This restriction allows the distinguished features selection portion of eNERF to be performed with only the top $H \times H$ portion of D, which may be loadable when D is not.

Algorithm **DF (Select h distinguished features from H rows of D_N)**

Choose: h = # of distinguished features to select
H = # of candidate rows for the h distinguished features, $h \le H$

Input: An $H \times H$ dissimilarity matrix D_H

(**DF**1) Define $m_1 = 1$. Initialize the search array
$\boldsymbol{\delta}^1 = [\ \delta_1^1, \delta_2^1, \ldots, \delta_H^1\ ]^T = [d_{11}, \ldots, d_{1H}]^T$.

(**DF**2) Define $m_2 = j$ where $\delta_j^1 \ge \delta_k^1$ for $1 \le k \le H$.

(**DF**3) Define next search array $\boldsymbol{\delta}^2 = [\delta_1^2, \delta_2^2, \ldots, \delta_H^2\ ]^T = [\min\{\delta_1^1, d_{m_2 1}\}, \ldots, \min\{\delta_H^1, d_{m_2 H}\}]^T$

(**DF**4) Define $m_3 = i$ where $\delta_i^2 \ge \delta_k^2$ for $1 \le k \le H$.

(**DF**5) After j steps, use $\boldsymbol{\delta}^j = [\ \delta_1^j, \delta_2^j, \ldots, \delta_H^j\ ]^T = [\min\{\delta_1^{j-1}, d_{m_j 1}\}, \ldots, \min\{\delta_H^{j-1}, d_{m_j H}\}]^T$ to choose the $j+1^{st}$ feature as that row among the remaining candidates whose index points to the maximum element of $\boldsymbol{\delta}^j$.

If there is not a unique minimizing argument at some stage, ties can be broken by any rule. Why use the term "distinguished features"? Well, algorithm **DF** is a *feature selection* algorithm, but not in the usual pattern recognition sense of the term – i.e., we are not selecting good features for clustering or classifier design – rather, we are selecting good features for progressive sampling. It is our hope, of course, that these features will lead us to good clusters in D, but the quality of the features for clustering does not determine their selection. However, we can relate algorithm **DF** to clusters in D in a very specific way.

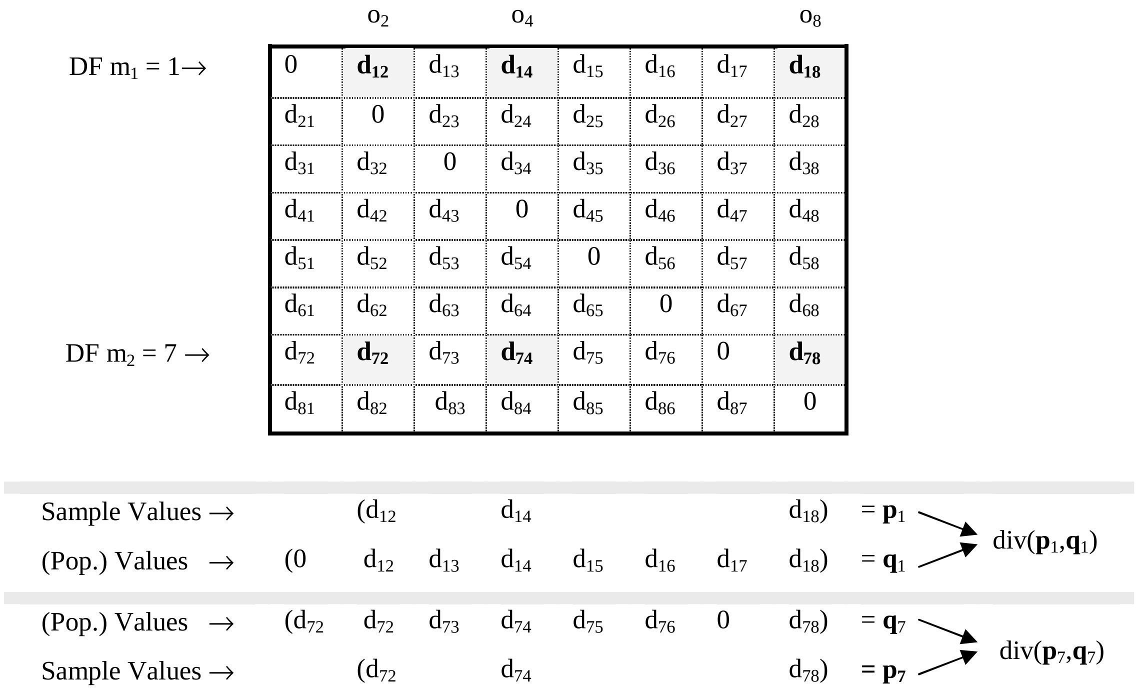
Algorithm **DF** is justified in terms of sampling potential clusters by relating it to Dunn's index of separation (Dunn, 1976) which characterizes sets of clusters that are compact and well separated by a geometric criterion. The relevant result is

Proposition DF. If O_H has c compact and separated clusters, then the first c distinguished features chosen by algorithm **DF** will consist of one row corresponding to an object from each of the c clusters.

Proof. Given in Hathaway et al. (2005)

2.3 Progressive sampling. The criterion for acceptability of O_n is based on comparing the distribution of the distinguished features found by algorithm DF in the columns corresponding to O_n and O_N using the divergence test statistic. The scheme is best illustrated by a small example, shown in Figure 2, where we depict an 8×8 dissimilarity matrix D_8 enclosed in the heavy square, and 2 (arbitrarily chosen) distinguished features, m_1 = feature 1 and m_2 = feature 7. Imagine now that the current candidate sample is $O_3 = \{o_2, o_4, o_8\}$. In essence, we will accept O_3 as a representative sample of O_8 if for *each* distinguished feature m_k, the distribution of the O_3 feature values (corresponding to the shaded cells in row m_k) closely approximates the distribution of the corresponding feature values for the full sample (corresponding to all entries in row m_k).

More specifically, we accept O_3 if the histogram of $\{d_{12}, d_{14}, d_{18}\}$ is *close enough* to that of $\{0, d_{12}, d_{13}, d_{14}, d_{15}, d_{16}, d_{17}, d_{18}\}$ <u>and</u> the histogram of $\{d_{72}, d_{74}, d_{78}\}$ closely approximates that of $\{d_{71}, d_{72}, d_{73}, d_{74}, d_{75}, d_{76}, 0, d_{78}\}$. The approximation is "close enough" when the divergence test statistic is in the left tail of the appropriate chi-square distribution for each distinguished feature. In the bottom portion of Figure 2, we have extracted the relevant values needed for the divergence tests and shown them as vectors **p** (sample values) and **q** ("population" values). We will accept the sample if, and only if, div($\mathbf{p}_1$,$\mathbf{q}_1$) AND div($\mathbf{p}_7$,$\mathbf{q}_7$) are both less than or equal to $F^{-1}(1\text{-}\varepsilon)$, where F is the *cumulative distribution function* (cdf) for the chi-square distribution with b–1 degrees of freedom (b and ε are discussed below). The use of a histogram-based test for acceptability requires selection of histogram bin intervals. Let b denote the desired number of histogram bins. The histogram interval widths are based only on the distinguished feature values of the initial sample, and the widths vary, as necessary, so that each histogram bin captures (as nearly as possible) the same number of initial sample observations. For notational simplicity we drop the second level of subscripts and let $d_{(1)} \le d_{(2)} \le \ldots \le d_{(n)}$ denote the order statistics for the values of distinguished feature m_k from the columns of D corresponding to an initial sample $O_n = \{ o_{i_1}, o_{i_2}, \ldots, o_{i_n} \}$.



		o_2		o_4				o_8
DF $m_1 = 1$→	0	$\mathbf{d_{12}}$	d_{13}	$\mathbf{d_{14}}$	d_{15}	d_{16}	d_{17}	$\mathbf{d_{18}}$
	d_{21}	0	d_{23}	d_{24}	d_{25}	d_{26}	d_{27}	d_{28}
	d_{31}	d_{32}	0	d_{34}	d_{35}	d_{36}	d_{37}	d_{38}
	d_{41}	d_{42}	d_{43}	0	d_{45}	d_{46}	d_{47}	d_{48}
	d_{51}	d_{52}	d_{53}	d_{54}	0	d_{56}	d_{57}	d_{58}
	d_{61}	d_{62}	d_{63}	d_{64}	d_{65}	0	d_{67}	d_{68}
DF $m_2 = 7$ →	d_{72}	$\mathbf{d_{72}}$	d_{73}	$\mathbf{d_{74}}$	d_{75}	d_{76}	0	$\mathbf{d_{78}}$
	d_{81}	d_{82}	d_{83}	d_{84}	d_{85}	d_{86}	d_{87}	0

Sample Values → $(d_{12}$ d_{14} $d_{18}) = \mathbf{p}_1$

(Pop.) Values → $(0$ d_{12} d_{13} d_{14} d_{15} d_{16} d_{17} $d_{18}) = \mathbf{q}_1$

div($\mathbf{p}_1$,$\mathbf{q}_1$)

(Pop.) Values → $(d_{72}$ d_{72} d_{73} d_{74} d_{75} d_{76} 0 $d_{78}) = \mathbf{q}_7$

Sample Values → $(d_{72}$ d_{74} $d_{78}) = \mathbf{p}_7$

div($\mathbf{p}_7$,$\mathbf{q}_7$)

Figure 2: Progressive sampling termination testing using DFs 1 and 7 for sample objects 2, 4 and 8

The b bin intervals of the histograms used to assess the suitability, according to distinguished feature m_k, of the original and all subsequent candidate samples are

$$\left[0, d_{\left(1+\left\lceil \frac{n}{b}\right\rceil\right)}\right), \left[d_{\left(1+\left\lceil \frac{n}{b}\right\rceil\right)}, d_{\left(1+\left\lceil \frac{2n}{b}\right\rceil\right)}\right),$$
$$\left[d_{\left(1+\left\lceil \frac{2n}{b}\right\rceil\right)}, d_{\left(1+\left\lceil \frac{3n}{b}\right\rceil\right)}\right), \ldots, \left[d_{\left(1+\left\lceil \frac{(b-1)n}{b}\right\rceil\right)}, \infty\right)$$

where $\lceil\ \rceil$ denotes the ceiling function. We refer to bins chosen this way as *equal content* bins, since the intervals divide the sample data exactly, or very nearly so, equally into the b bins.

Once the bin intervals are selected using the initial O_n, the same intervals are used - they are *not* redefined - for all subsequent candidates. For an example suppose that the initial sample is $O_{45} = \{o_{i_1}, o_{i_2}, \ldots, o_{i_{45}}\}$ with the 45 corresponding sorted values $d_{(1)} \le d_{(2)} \le \ldots \le d_{(45)}$ for distinguished feature m_k. Then for b = 4, the intervals defining the 4 bins are $[0, d_{(13)})$, $[d_{(13)}, d_{(24)})$, $[d_{(24)}, d_{(35)})$, and $[d_{(35)}, \infty)$. Assuming the 45 data values are all distinct, the (nearly equal) respective counts for the 4 intervals are 12, 11, 11, and 11.

There is one more issue concerning the sample data matrix that will be sent to LNERF for clustering. When $O_n = \{o_{i_1}, \ldots, o_{i_n}\}$, or equivalently, the corresponding columns of D_n, is accepted, then exactly what data are passed to LNERF? The sample used by LNERF consists of the set of all pair-wise dissimilarities of objects in $\{o_{i_1}, o_{i_2}, \ldots, o_{i_n}\}$, which is conveniently arrayed as an n × n submatrix of the full sample matrix D_N. We denote this submatrix by D_n, where element $d_{n_{jk}}$ is the pair-wise dissimilarity between sample objects o_{i_j} and o_{i_k}. For example, if the accepted sample is $O_3 = \{o_{i_1}, o_{i_2}, o_{i_3}\}$ = $\{o_2, o_4, o_8\}$ as in Figure 2, then sample data matrix of dissimilarities that will be processed by LNERF is $D_3 = \begin{bmatrix} 0 & d_{24} & d_{28} \\ d_{42} & 0 & d_{48} \\ d_{82} & d_{84} & 0 \end{bmatrix}$. Recall that we let F denote the cdf for the chi-square distribution with b–1 degrees of freedom. The sampling scheme is:

Algorithm PS : Relational Progressive Sampling

Inputs: An $h \times N$ dissimilarity matrix D. The rows of D correspond to the h distinguished features $\{m_1, m_2, \ldots, m_h\}$ selected by algorithm **DF** on D_H

Constraints: D satisfies conditions (1)

Choose: b = # of histogram intervals
p = the initial sample percentage
Δp = the incremental percentage

ε_{PS} = termination criterion

(**PS**1) Randomly select (without replacement) $n = \lceil (pN)/100 \rceil$ column indices $I_n = \{c_1, ..., c_n\}$ from $I_N = \{1, 2, \ldots, N\}$.

(**PS**2) For k=1 to h: define EC histogram bins for distinguished feature m_k with $\{d_{m_k c_1}, d_{m_k c_2}, ..., d_{m_k c_n}\}$.

(**PS**3) For i=1 to b; for k=1 to h:

Calculate N_i^k, the full set count for bin i and m_k, using row m_k of D.

Calculate n_i^k, the sample count for bin i and m_k, using $\{d_{m_k c_1}, d_{m_k c_2}, ..., d_{m_k c_n}\}$.

(**PS**4) For k=1 to h: calculate the divergence test criterion for distinguished feature m_k

$$\mathrm{div}_k = n\sum_{i=1}^{b}\left(\frac{N_i^k}{N} - \frac{n_i^k}{n}\right)\ln\left(\frac{nN_i^k}{Nn_i^k}\right) .$$

(PS5) WHILE ($\mathrm{div}_k > F^{-1}(1-\varepsilon)$ **for at least one k ∈ {1,2,...,h})**

$\Delta n = \min\{N-n, (\Delta p\, N)/100\}$: $n = n + \Delta n$

Randomly select $\Delta D = (\Delta n)$ previously unselected columns of D

$D_n = D_n + \Delta D$ % Adding Δn columns to D_n also adds Δn rows to D_n

Calculate $\mathrm{div}_k = n\sum_{i=1}^{b}\left(\frac{N_i^k}{N} - \frac{n_i^k}{n}\right)\ln\left(\frac{nN_i^k}{Nn_i^k}\right)$; k = 1,...,h

Output : $n \times n$ (Sample) Dissimilarity matrix D_n

2.4 LNERF. Clustering in D_n is done with the literal NERF algorithm from Hathaway and Bezdek (1994). In the following, $\|\cdot\|$ is the Euclidean norm on $\Re^n$, $\mathbf{e}_k$ denotes the k^th unit vector in $\Re^n$, M is the n × n matrix with 0's on the main diagonal and 1's elsewhere ($M = [\mathbf{1}] - I_n$), and M_{fcn} is the set of all fuzzy partition matrices $U \in \Re^{cn}$ which satisfy $u_{ik} \in [0,1]$, $\sum_{i=1}^{c} u_{ik} = 1$ for k = 1, ..., n and $\sum_{k=1}^{n} u_{ik} > 0$ for i = 1,...,c. Don't confuse $U^{(q)}$, the q^th estimate of the $c \times n$ matrix U, with our notation $\mathbf{U}^q$, which denotes the q^th column of matrix U.

Algorithm LNERF : Fuzzy clusters in dissimilarity matrix D_n (Hathaway and Bezdek, 1994)

Inputs: An $n \times n$ dissimilarity matrix D_n

Constraints: D_n satisfies conditions (1) and $\alpha \in \Re$, $M = [\mathbf{1}] - I_n$

Choose: c = # of clusters, $2 \le c < n$
m = fuzzy weighting exponent, $m > 1$
ε_L = termination criterion
$\|U^{(q)} - U^{(q-1)}\|$ = termination norm
Q_M = maximum number of iterations

Initialize: q = 0; β = 0; $U^{(0)} \in M_{fcn}$
$U_{difference} = 2^*\varepsilon_L$

WHILE ($U_{difference} > \varepsilon_L$ **AND** $q < Q_M$)

(**LN**1) For 1 < I < c, calculate "mean" vector $\mathbf{v}_i^{(r)}$

$$\mathbf{v}_i^{(q)} = ((u_{i1}^{(q)})^m, \ldots, (u_{in}^{(q)})^m)^T \Big/ \sum_{j=1}^{n}((u_{ij}^{(q)})^m \quad , \qquad (2a)$$

(**LN**2) Calculate (cn) object to cluster "distances"

$$\delta_{ik} = ((D_n + \beta M)\mathbf{v}_i^{(q)})_k - (\mathbf{v}_i^{(q)^T}(D_n + \beta M)\mathbf{v}_i^{(q)})/2 \qquad (2b)$$

IF $\delta_{ik} < 0$ for any i and k, **THEN** calculate

$$\Delta\beta = \max_{i,k}\left\{-2\delta_{ik} / \left\|\mathbf{v}_i^{(q)} - \mathbf{e}_k\right\|^2\right\} \quad ; \qquad (2c)$$

$$\delta_{ik} \leftarrow \delta_{ik} + (\Delta\beta/2) \cdot \left\|\mathbf{v}_i^{(q)} - \mathbf{e}_k\right\|^2 \quad ; \qquad (2d)$$

$$\beta \leftarrow \beta + \Delta\beta \quad . \qquad (2e)$$

(**LN**3) Update $U^{(q)}$ to $U^{(q+1)} \in M_{fcn}$ for all k =1, …, n:

IF $\boldsymbol{\delta_{ik}}$ **> 0 for any i = 1 to c** THEN

$$u_{ik}^{(q+1)} = 1 \Big/ \left[\sum_{j=1}^{c}\left(\delta_{ik}/\delta_{jk}\right)^{1/(m-1)}\right] \quad ; \qquad (2f)$$

ELSE

$$u_{ik}^{(q+1)} = 0 \text{ for all k with } \delta_{ik} > 0$$

$$u_{ik}^{(q+1)} \in [0,1] \text{ such that } \sum_{j=1}^{c} u_{jk}^{(q+1)} = 1 \qquad (2g)$$

$q \leftarrow q + 1$

$U_{difference} = \|U^{(q)} - U^{(q-1)}\|$

Outputs : Membership matrix $U_{LNERF,n} \in M_{fcn}$;
"prototype" vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_c\} \subset \Re^n$

The main result in Hathaway and Bezdek (1996) is that the sequence of partition matrices produced by LFCM on an object data set X is identical to the sequence of partition matrices produced by LNERF on the corresponding relational matrix D(X) of pair-wise *squared* Euclidean distances derived from X, i.e., $[d_{ij}(X)] = \left[\|\mathbf{x}_i - \mathbf{x}_j\|^2\right]$. The "*non*-Euclidean" terminology indicates that LNERF is applicable to non-Euclidean data.

Continuation of the algorithm is achieved by β-spreading D to $D_\beta = D_n + \beta M$ via equations (2b)-(2e) which, when triggered, add spread term β to the off diagonal elements of D. Addition of the term βM produces relational dissimilarity data that is very nearly Euclidean, while preserving most of the cluster structure of the original D. The $\{\delta_{ik}\}$ shown in equations (2) are analogous to distances in LFCM, the object data dual of LNERF, but are not true distances in the relational data setting. Finally, the specific norm used in LNERF for termination is important only in that different norms may stop the algorithm at different elements in the iterate sequence.

2.5 Extension. The final component of eNERF is extension of LNERF results on O_n to the objects in (O_N - O_n). We temporarily denote the LNERF clustering results by (U = U_{LNERF}, **V**) without subscripts, where U = $[u_{ik}] \in M_{fcn}$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2,..., \mathbf{v}_c] \in \Re^{nc}$. The result of extension is an augmented $c \times N$ matrix U_{app} obtained by appending (N – n) columns from xNERF to the $c \times n$ matrix U. U_{app} is the approximation to U_{lit}, the partition of D_{VL} that we seek, but cannot compute. *Extended NERF* (xNERF) calculates $\mathbf{U}^j = (u_{1j},\ldots,u_{cj})^T$, the membership column in the output matrix U_{xNERF} corresponding to o_j, for n+1 ≤ j ≤ N. Unlike the direct calculation used to extend FCM in the image and general object data cases, xNERF extension requires iteration to produce a label vector for each of the N-n unlabelled objects. This sounds worse, complexity-wise, than it is. Several vector computations are done just once. Subsequently, only simple scalar quantities are recalculated, so the iteration that produces each $\mathbf{U}^j$ is fairly inexpensive.

Calculating memberships for o_j begins by appending some of o_j 's corresponding relational data to the sample relational matrix D_n to obtain an (n+1) × (n+1) augmented relational matrix. Then LNERF is applied to this augmented matrix, but the sample-based estimates of U and **V** from LNERF are *fixed*.

Let β_n denote the final shifting value obtained from (2e) while clustering D_n using LNERF; and let D_{β_n} denote the β_n shifted version of D_n, $D_{\beta_n} = D_n + \beta_n M$. Let $D_N = [d_{ik}]$ denote the original N×N full relation matrix. For convenience, *we assume without loss of generality* that the objects in O_N have been re-indexed so that the unlabeled objects are adjacent, $O_N - O_n = \{o_{n+1}, o_{n+2}, \ldots, o_N\}$. To assign a fuzzy label vector to the jth object in O_N - O_n, we first define some auxiliary quantities built from the values of row or column j of D_N and the outputs of LNERF on D_n. Let $\mathbf{d}_j = (d_{1j},\ldots,d_{nj})^T$, $\boldsymbol{\beta}_n = (\beta_n,\ldots,\beta_n)^T$ and $\mathbf{z}_j = (\mathbf{d}_j + \boldsymbol{\beta}_n)$ be n-vectors. Next, define the (n+1) × (n+1) augmented matrix D_j as

$$D_j = \begin{bmatrix} D_{\beta_n} & \mathbf{z}_j \\ \mathbf{z}_j^T & 0 \end{bmatrix} \quad . \qquad (11)$$

For i = 1, ..., c, let

$$a_i = \sum_{k=1}^{n} u_{ik}^m \text{ using the terminal U from (2g, h) ,} \qquad (3a)$$

$$b_i = \mathbf{v}_i^T D_{\beta_n} \mathbf{v}_i \text{ using the terminal } \mathbf{v}_i\text{'s from (2a),} \qquad (3b)$$

$$c_i = \mathbf{z}_j^T \mathbf{v}_i = \langle \mathbf{z}_j, \mathbf{v}_i \rangle \quad . \qquad (3c)$$

These three quantities are used by xNERF to iteratively estimate two sets of unknowns for the object $o_j \in O_N - O_n$**.** The unknowns that will be estimated are: c values $\{v_{ij} : 1 \le i \le c\}$, which correspond to the n+1st "components" of new prototype vectors gotten by applying LNERF to a distance matrix of size $(n+1)\times(n+1)$; and a $c\times 1$ label vector $\mathbf{U}^j \in N_{fc}$. Next we append the unknown components of the prototypes we seek to the input prototypes from LNERF:

$$\mathbf{v}_{ij} = [\mathbf{v}_i^T, v_{ij}]^T \; ; 1 \le i \le c \quad . \qquad (4)$$

Let $\mathbf{1} = (1,1,...,1)^T \in \Re^n$ and define M_j as

$$M_j = \begin{bmatrix} \mathbf{0}_{n\times n} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \in \Re^{(n+1)\times(n+1)} \quad . \qquad (5)$$

We use $\mathbf{1}^T\mathbf{v}_i$ = 1, due to normalization of the sample prototype vectors during LNERF processing of D_n, to simplify our description of **xNERF**. Let $\tau \in \Re$ be any real number, and calculate

$$\left(\left(D_j + \tau M_j\right)\mathbf{v}_{ij}\right)_{n+1} = \left(\begin{bmatrix} D_{\beta_S} & \mathbf{z}_j + \tau\mathbf{1} \\ \mathbf{z}_j^T + \tau\mathbf{1}^T & 0 \end{bmatrix}\begin{bmatrix} \mathbf{v}_i \\ v_{ij} \end{bmatrix}\right)_{n+1} ; \quad (6a)$$

$$= \mathbf{z}_j^T \mathbf{v}_i + \tau\mathbf{1}^T\mathbf{v}_i = c_i + \tau$$

Also,

$$(\mathbf{v}_{ij})^T (D_j + \tau M_j)(\mathbf{v}_{ij}) = \left[\mathbf{v}_i^T, v_{ij}\right]\begin{bmatrix} D_{\beta_S} & \mathbf{z}_j + \tau\mathbf{1} \\ \mathbf{z}_j^T + \tau\mathbf{1}^T & 0 \end{bmatrix}\begin{bmatrix} \mathbf{v}_i \\ v_{ij} \end{bmatrix}$$

$$= \left[\mathbf{v}_i^T, v_{ij}\right]\begin{bmatrix} D_{\beta_S}\mathbf{v}_i + v_{ij}(\mathbf{z}_j + \tau\mathbf{1}) \\ c_i + \tau \end{bmatrix}$$

$$\begin{aligned} &= \mathbf{v}_i^T D_{\beta_S}\mathbf{v}_i + v_{ij}(\mathbf{v}_i^T\mathbf{z}_j + \tau\mathbf{v}_i^T\mathbf{1}) + v_{ij}(c_i + \tau) \\ &= b_i + 2v_{ij}(c_i + \tau) \end{aligned} \quad . \qquad (6b)$$

Using (6) we can simplify the quantity

$$\left(\left(D_j + \tau M_j\right)\mathbf{v}_{ij}\right)_{n+1} - \frac{1}{2}\left(\mathbf{v}_{ij}\right)^T\left(D_j + \tau M_j\right)\mathbf{v}_{ij} =$$

$$c_i + \tau - \frac{1}{2}\left(b_i + 2v_{ij}(c_i + \tau)\right) = (c_i + \tau)(1 - v_{ij}) - \frac{b_i}{2} \quad . \qquad (7)$$

Now we are ready to state the extension procedure.

Algorithm xNERF (Extension of LNERF to label object $o_j \in O_N - O_n : j = n+1,..., N$)

Inputs: From algorithm (**PS**): D_n

```
            From algorithm (LNERF):
                $\beta_n ; D_{\beta_n} = D_n + \beta_n(M);$
                $U = U_{LNERF} \in M_{fcn} ; \mathbf{V}_{c\times n}$
             $m$ = fuzzy weighting exponent, $m > 1$

Choose:         $\varepsilon_x$ = termination criterion
                $Q_M$ = maximum number of iterations
                $\left\|\mathbf{U}^{(q)} - \mathbf{U}^{(q-1)}\right\|$ = termination norm
Calculate:      $a_i$ and $b_i$ from (12) for i = 1 to c.

FOR j = n+1 to N

 (xN 1) Initialize:  q = 0; $\beta = 0$; $\mathbf{U}_{difference} = 2*\varepsilon_x$ ;
     $\mathbf{U}^{(0)} = (1/c,\ldots,1/c)^T \in N_{fc}$ ;
     Calculate $c_i$ from (3c) for i = 1 to c.

(xN 2 )WHILE ($\mathbf{U}_{difference} > \varepsilon_x$ AND $q < Q_M$)

Calculate new prototypes $\mathbf{v}_{ij}^{(q)} \in \Re^{n+1}$ for i = 1 to c:
  $\mathbf{v}_{ij}^{(q)} = \left[\mathbf{v}_i^T, \frac{k_{ij}}{(a_i + k_{ij})}\right]^T$,   where   $k_{ij} = \left(u_{ij}^{(q)}\right)^m$

Calculate new "distances" (when positivity is violated)
from $o_j$ to the c clusters for i = 1 to c:

IF  $\delta_{ij} < 0$ for any $i \in \{1,...,c\}$, THEN
     $\Delta\beta = -\underbrace{\min}_{i}\{\delta_{ij}\}$
     $\delta_{ij} \leftarrow \delta_{ij} + \Delta\beta$ for i = 1 to c
     $\beta \leftarrow \beta + \Delta\beta$
ENDIF

Calculate $\{u_{ij}^{(q+1)}\}$ for $o_j$ for i = 1 to c:
IF $\delta_{ij} > 0$, i = 1 to c THEN
  $u_{ij}^{(q+1)} = \left(\sum_{h=1}^{c} \frac{\delta_{ij}}{\delta_{hj}}\right)^{\frac{-1}{(m-1)}}$
ELSE
  $u_{ij}^{(q+1)} = 0$ for all k with $\delta_{ik} > 0$, and
  $u_{ij}^{(q+1)} \in [0,1]$ such that $\sum_{j=1}^{c} u_{ij}^{(q+1)} = 1$
ENDIF
   Update $q \leftarrow q + 1$
   Calculate $\mathbf{U}_{difference} = \left\|\mathbf{U}^{(q)} - \mathbf{U}^{(q-1)}\right\|$
END WHILE
   $\mathbf{U}^j = (u_{1j}^{(q)},\ldots,u_{cj}^{(q)})^T$
NEXT j
```

Output : Matrix $U_{xNERF} = [\mathbf{U}^j,\ldots,\mathbf{U}^N] \in M_{fc(N-n)}$

Remarks. (i) it is *never* necessary to load the entire matrix D_N = D_{VL}. Finding the h DFs that monitor sampling uses an $H \times H$ submatrix of D_N. (ii) progressive sampling operates on an h × N portion of D_N, which is just the h rows of D_N corresponding to the h DFs. (iii) LNERF clustering operates on the $n \times n$ matrix D_n, where n is the size of the accepted sample. (iv) xNERF accesses elements in an n × (N-n) portion of D_N.

Figure 3 summarizes the **eNERF** approach to clustering in VL dissimilarity data, which consists of the sequential application of four algorithms: **DF, PS, LNERF** and **xNERF**. The final operation in this sequence is to form a c-partition of O_N (if this is desired), which is done by concatenating $U_{xNERF,N-n}$ with $U_{LNERF,n}$, thereby creating $U_{app} = [U_{LNERF,n} \mid U_{xNERF,N-n}]_{c\times N}$, and the approximation to literal clusters in D_N, which cannot be computed when D_N is unloadable. The shaded cells in the upper left panel in Figure 3 depict the H×H matrix from which the h distinguished features $\{m_k\}$ are chosen by algorithm DF.

The shaded rows in the upper right panel are the full rows of D_N corresponding to the h rows in D_H. This is the input matrix to our progressive sampling scheme. The shaded cells in the lower right panel are the entries of D_n, the sample matrix processed by LNERF. And finally, the lower left panel depicts the reindexed version of D_N. The blocks of shaded cells in the first n columns are the fixed values of D_n, while the n shaded cells in column j, $n+1 \le j \le N$, are the components of the vector $\mathbf{d}_j$. The matrix D_j which is alluded to in this view is built from the shaded cells in this view and the terminal β-spread value β_n from LNERF.

3 Numerical Examples

The termination norms for the two examples are: for LNERF, the sup norm for matrices, regarding them as vectors in $\Re^{cn}$; and for xNERF, the sup norm on $\Re^c$. The first example gives a visual display of the quality of an extended partition using a 194 × 194 matrix consisting of pair-wise similarities from a set of human gene products. The second example demonstrates eNERF on a problem that is too big for the PC used in the calculations, which consisted of 40,000 × 40,000 dissimilarities. Storage of this matrix can be cut in half by storing just the upper triangular part of the symmetric input matrix, but the data are still too large to be handled directly with LNERF.

Example 1. We demonstrate eNERF by clustering a subset of $GDP194_{4.30.04}$ with LNERF, extending the results to the remaining objects with xNERF, and then comparing $U_{app} = [U_{LNERF,n} \mid U_{xNERF}]_{c\times N}$ to $U_{LNERF,N}$.

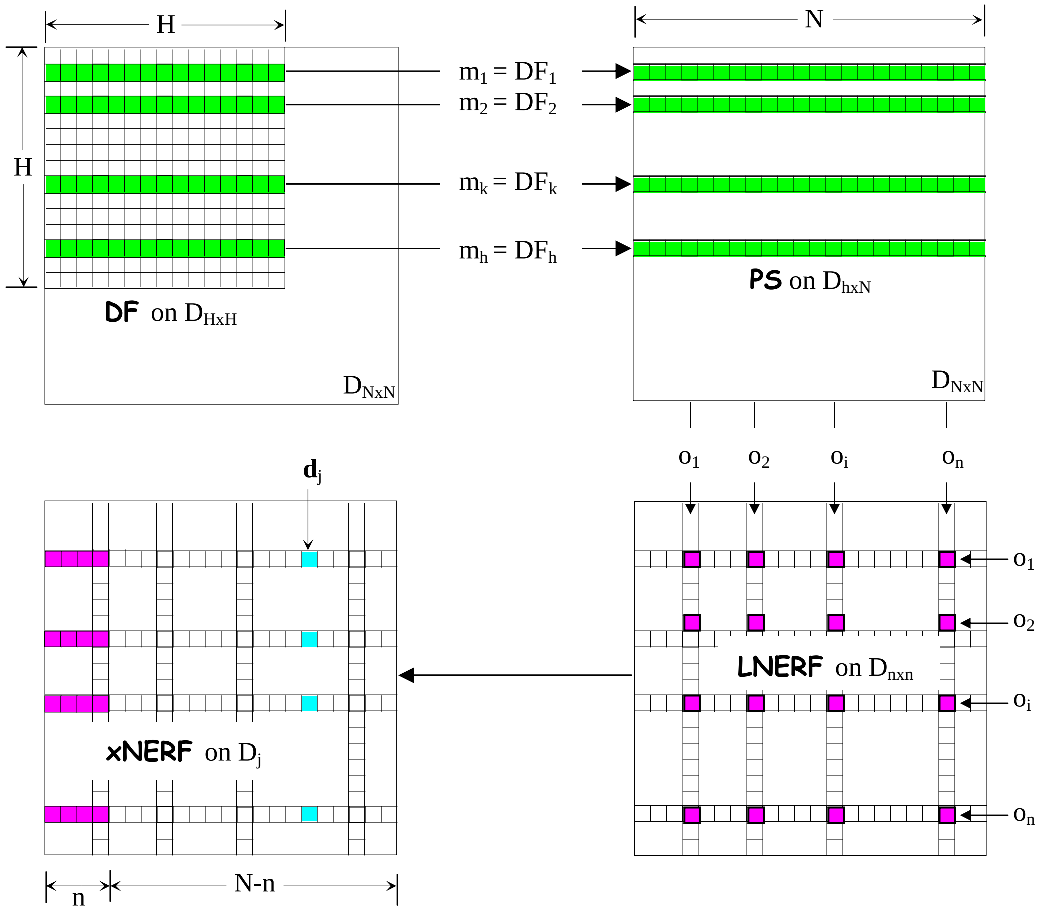


Figure 3: Architecture of eNERF clustering in VL dissimilarity data

$GDP194_{4.30.04}$ is a 194 x 194 matrix whose entries are similarities between pairs of gene products developed on sets of linguistic descriptors of each protein in the set; see Pal et al. (2005) for a complete description of the construction and interpretation of this data. The specific matrix we use here was called LOS4. The subscript indicates the date that the data was originally collected. Since the original similarity values of the data are all in [0,1], we obtained dissimilarities by subtracting each similarity from 1.

There is evidence (from human experts and other clustering algorithms) to believe that the data consists of 3 main clusters, which group them into three protein families {myotubularin, receptor precursor, collagen alpha chain}. The data were ordered so that the first 21 columns (or rows) corresponded to the first cluster, as did the next 87, and the final 86. Hardening an LNERF fuzzy 3-partition of the data produces these same clusters of sizes 21, 87, and 86.

Relational data and clusters in it can be displayed as a grayscale image. The input dissimilarity matrix for $GDP194_{4.30.04}$, in the original order (clustered by human experts), is shown in Figure 4(a). The three dark blocks along the main diagonal correspond to the three main clusters.

The purpose of this example is to demonstrate the visual quality of extended partitions using different percentages of data in the sample. To do this we need to randomly sample a percentage of data from the matrix D corresponding to Figure 4(a). One way to randomly sample from the columns is to randomly permute the rows and columns of D by the same permutation of 1,2,...,194. After permutation, a random sample can be easily obtained by choosing, in order, columns from the scrambled version of D. This approach allows us to better "see" the random sample. Using MATLAB routine randperm, seeded with value 824567, we obtained the scrambled form of the dissimilarity data shown in Figure 4(b), which we denote here as D_{scram}.

Visual information about clusters in a dissimilarity image (Figure 4(a)) is also available by viewing a transformed version of a fuzzy partition found by clustering the data. Figure 5 demonstrates this with images based on extended fuzzy membership matrices computed from the first (a) 10% of D_{scram}, (b) 25% of D_{scram}, (c) 75% of D_{scram}, and (d) 100% of D_{scram} (LNERF on D_{scram}). To appreciate the extended partition images, note that these percentages are based on the ratio of sample size n to full sample size N=194. So the 10% case corresponds to a sample dissimilarity matrix D_{19} consisting of the 19 × 19 leading principal submatrix of D_{scram}.

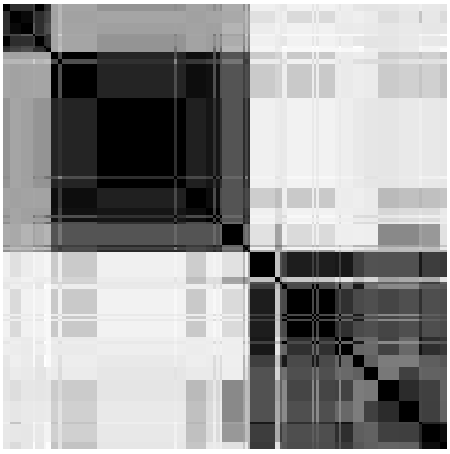


Figure 4a: Input dissimilarity matrix D_{194}

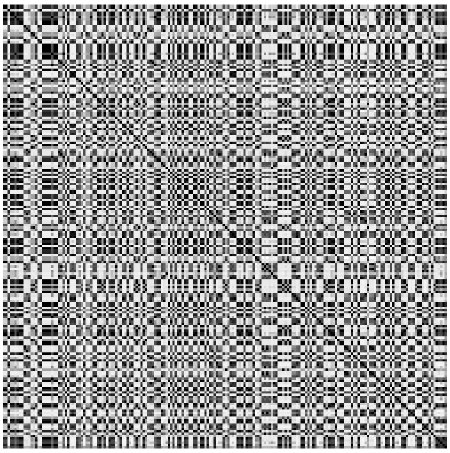


Figure 4b: D_{scram} reordering of D_{194}

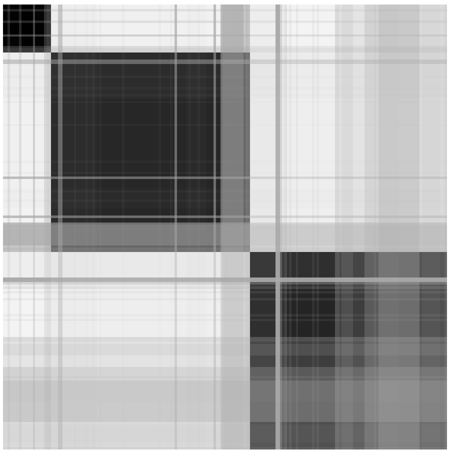


Figure 5a: 10% sample (n=19)

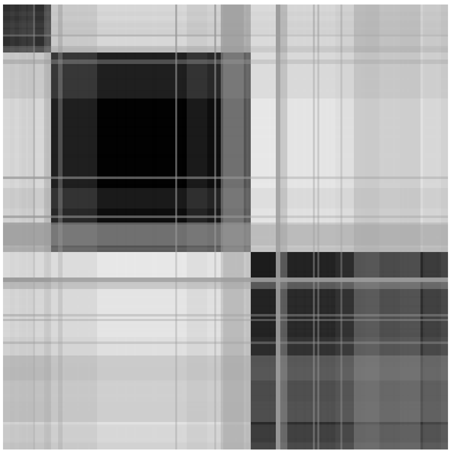


Figure 5c: 75% sample (n=144)

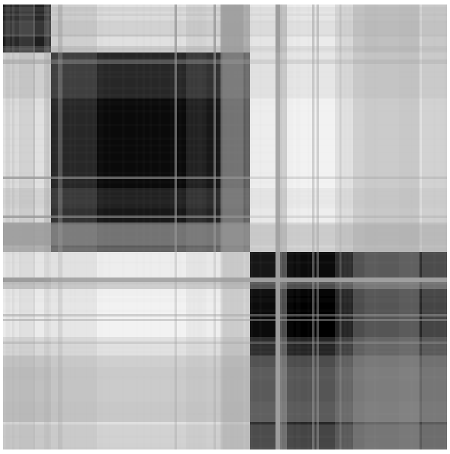


Figure 5b: 25% sample (n=48)

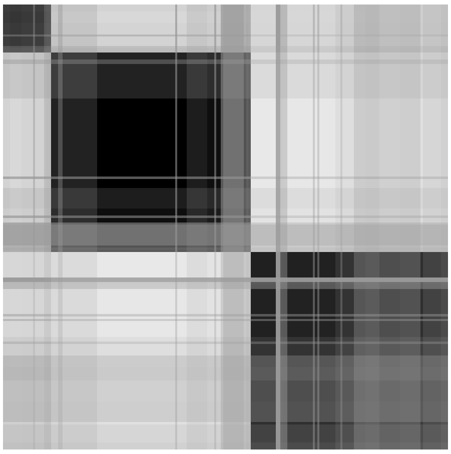


Figure 5d: 100% (n=194: $U_{LNERF,N}$)

The percentage of elements (not columns) of D_{scram} actually used by LNERF in the clustering phase is about 1 percent. Likewise, using 75% of the 194 objects in our scheme is equivalent to using 9/16 ≈ 56% of the elements of D_{scram}. The images in Figure 5 were produced from fuzzy partitions of D_{scram} in the following way. The columns of the fuzzy partition U_{scram} of D_{scram} are reordered (unscrambled) to get U, corresponding to the same object ordering as in Figure 4(a). Then the corresponding image matrix $I(U) = J - (U^TU / \max(U^TU))$ where J denotes the 194 × 194 matrix of 1's and $\max(U^TU)$ denotes the largest element in U^TU. Figures 5(a, d) show that the extended partition based on only 10% of D_{194} is a close (visual) approximation to the result obtained by applying LNERF to D_{194} itself. The extended partition images for successively larger samples of D_{194} in views (a)-(c) are increasingly better approximations to the literal partition seen in view (d). Note, too, the clear similarity between all of the partition-based images in Figure 5 and the dissimilarity-based image of Figure 4 (a). And finally, we observe that when analyzing clustering outputs with images such as these, there is no need to harden fuzzy partitions – these images are built directly with the fuzzy memberships produced by LNERF and xNERF.

Example 2. This example demonstrates the feasibility property of eNERF by clustering a 40,000 × 40,000 relational data matrix derived as pair-wise squared Euclidean distances from a set of 40,000 2-dimensional points which form 8 very well separated, compact clusters. This data set was first used for testing a visual display technique in Huband et al. (2005), and is represented by a scatter plot in Figure 6.

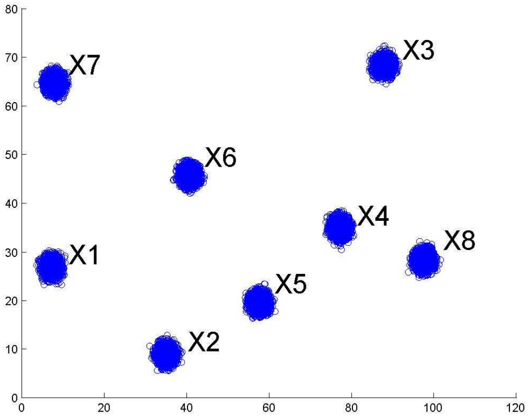


Figure 6. Scatter plot of 8-cluster, 40000-point data set in $\Re^2$ from Huband et al. (2005)

The 8-cluster data set can be clustered by almost any object data clustering algorithm without difficulty on the PC used for this experiment. In particular, we can cluster this data with LFCM in $\Re^2$, so the full data fuzzy partition U_{lit} is obtainable, and by the duality theory, identical to the one obtained by running LNERF on the corresponding distance matrix. But for N=40000, we are not able to calculate, load and process the full distance data matrix D_N. In other words, if we had *only* the relational data, this *would* represent a VERY LARGE clustering problem relative to the computing environment available. In fact, an 800 × 40000 slice (1/50) of D_N requires 244 MB of storage as a MATLAB *.MAT file.

To cluster D using eNERF we first performed progressive sampling with the following parameter values: divergence acceptance threshold $\varepsilon_{PS} = 0.80$; number of DF candidates H = 10; number of DFs h = 1; number of histogram bins b = 10; initial sample percentage p = 1 (% of N = 40000) = 400; and incremental sample percentage Δp = 1 (% of N = 40000) = 400. The choice h = 1 means that, for any choice of H, the only DF chosen is the first feature (row 1). The PS scheme terminated after one increment, the divergence accepting a representative sample of size n = 800.

For LNERF, we used m = 2, $\varepsilon_L = 0.00001$. Initializing with the correct hard 8-partition of the data, the 800 × 800 submatrix D_n was clustered after 3 iterations. The xNERF phase required a 800 × 40000 slice of D which had storage requirements of 244 MB. To avoid "out of memory" errors with MATLAB, the processing was broken up by calling the extension routine 49 times, each time supplying it with D_n and an additional 800 x 800 sub block of D_N. This chunk was used to extend the partition for another 800 objects. The stopping criterion for the extension iteration used $\varepsilon_x = 0.001$, and the final extended result U_{app} satisfied $\left\| U_{\text{lit}} - U_{\text{app}} \right\|_F = 0.2548$, which is quite small since these matrices are 8×40000. This was certainly an easy problem, in terms of how well separated the clusters actually are, but the point here was to demonstrate the feasibility property of eNERF, and this example does that.

4. Discussion

We have shown how to extend NERF clustering to arbitrarily large dissimilarity data. Several opportunities for future work immediately come to mind. An efficient implementation of eNERF, particularly for xNERF, needs to be made to find out to what degree, and for what problems, eNERF can provide acceleration to LNERF. A theoretical analysis of the convergence properties of xNERF would be of value and would likely be possible using the alternating optimization framework of Bezdek and Hathaway (2003). One of the most interesting issues concerns the selection of the optimal distinguished features, those that best provide samples that correlate well with accurate extended partitions. Is an alternative, non-iterative, extension possible? We did very limited experimentation using a direct FCM-based extension scheme, which proved to be faster, but much less accurate. Finally, the interesting discovery regarding the similarity between partition and relational based images in Example 1 suggests a potentially cheaper (since U is smaller than D) approach than those in Huband et al. (2005) for visual displays to assess cluster tendency and validity.

5 References

Bezdek, J.C., Keller, J.M., Krishnapuram, R. and Pal, N.R., (1999): *Fuzzy models and algorithms for pattern recognition and image processing*. Springer, NY.

Bezdek, J. C. and Hathaway, R.J. (2003): Convergence of alternating optimization. *Neural, Parallel and Scientific Computations*, **11**, 351-368.

Bradley, P., Fayyad, U. and Reina, C. (1998): Scaling clustering algorithms to large databases. *Proc. 4th Int'l. Conf. Knowledge Discovery and Data Mining*, 9-15, AAAI Press, Menlo Park, CA,.

Dunn, J. C. (1976): Indices of partition fuzziness and the detection of clusters in large data sets, in *Fuzzy Automata and Decision Processes*, M. M. Gupta (ed), Elsevier, NY.

Fayyad, U. and Smyth, P. (1996): From massive data sets to science catalogs: applications and challenges. *Proc. Workshop on Massive Data Sets*, J. Kettering and D. Pregibon (eds), National Research Council.

Ganti, V., Ramakrishnan, R., Gehrke, J., Powell, A. L. and French, J. C. (1999): Clustering large data sets in arbitrary metric spaces. *Proc. 15th Int'l. Conf. on Data Engineering*, 502-511, IEEE CS Press, Los Alamitos, CA.

Hathaway, R. J. and Bezdek, J. C. (1994): NERF c-means: non-Euclidean relational fuzzy clustering. *Patt. Recog*., **27(3)**, 429-437.

Hathaway, R.J. and Bezdek, J.C. (2005). Approximate clustering in very large data sets. In press, *Comp. Statistics and Data Analysis.*

Hathaway, R. J., Bezdek, J. C., Huband, J. M., Leckie, C. and Kotagiri, R. (2005): Approximate clustering in very large relational data, in review, *Jo. Intell. Syst*.

Huband, J., Bezdek, J. C. and Hathaway, R J. (2005): bigVAT: visual assessment of cluster tendency for large data sets. *Patt. Recog*., **38**, 1875-1886.

Huber, P., (1996): Massive data workshop: The morning after. *Massive Data Sets*, 169-184, National Academy Press.

Ng, R. T. and Han, J. (1994): Efficient and effective clustering methods for spatial data mining. *Proc. 20th Int'l. Conf. On Very Large Databases*, 144-155, Morgan Kauffman, San Francisco.

Pal, N.R. and Bezdek, J.C. (2002): Complexity reduction for "large image" processing. *IEEE Trans. on Systems, Man and Cybernetics*, **B(32)**, 598-611.

Pal, N. R, Keller, J. M., Mitchell, J.A., Popescu, M., Huband, J. M. and Bezdek, J.C. (2005): Gene ontology-based knowledge discovery through fuzzy cluster analysis, in press, *Neural, Parallel and Scientific Computing*.

Taskar, B., Segal, E. and Koller, D. (2001). Probabilistic clustering in relational data. *17th International Joint Conference on Artificial Intelligence*, 870-876, Seattle, USA.

INVITED PAPER

Suffix Arrays: What Are They Good For?

Simon J. Puglisi
Dept. of Computing
Curtin University of Technology
Perth, Australia
Email: puglissj@computing.edu.au

William F. Smyth
McMaster University, Hamilton, Canada
Curtin University of Technology, Perth, Australia
Email: smyth@computing.edu.au

Andrew Turpin
Dept. of Computer Science & IT
RMIT University
Melbourne, Australia
Email: aht@cs.rmit.edu.au

Recently the theoretical community has displayed a flurry of interest in suffix arrays, and compressed suffix arrays. New, asymptotically optimal algorithms for construction, search, and compression of suffix arrays have been proposed. In this talk we will present our investigations into the practicalities of these latest developments. In particular, we investigate whether suffix arrays can indeed replace inverted files, as suggested in recent literature on suffix arrays.

Background

In 1990 Manber & Myers proposed suffix arrays as a space-saving alternative to suffix trees and described the first algorithms for suffix array construction and use (Manber & Myers 1990, Manber & Myers 1993). It has since been shown that any problem whose solution can be computed using suffix trees is solvable with the same asymptotic complexity using suffix arrays (Abouelhoda, Kurtz & Ohlebusch 2004). In addition, suffix arrays use much less memory than suffix trees, even less when they are compressed (Ferragina & Manzini 2000, Sadakane 2002, Grossi, Vitter & Gupta 2004, Puglisi, Turpin & Smyth 2005*a*, Mäkinen & Navarro 2005).

It has recently been shown that given an n character text T and its corresponding suffix array S, with some preprocessing and auxiliary information, it is possible to search for an arbitrary m character pattern P in T using only $O(m)$ time (Sim, Kim, Park & Park 2003). This is superior to non-index based string matching algorithms like that of Knuth, Morris & Pratt (1977) and Boyer & Moore (1977) which are linear in both the pattern and text length, requiring $O(m + n)$ time to find P in T. In conjunction with these time-efficient searching algorithms, time-efficient construction algorithms have also been developed that require only $O(n)$ time to construct the suffix array on an n character text (Puglisi, Turpin & Smyth 2005*b*).

Subsequent research on compressed suffix arrays (Sadakane 2000, Mäkinen 2000, Grossi & Vitter n.d.) and similar structures has revealed that *self-indexing* structures are possible, which can search for and report matches without the need for the original text to be stored (Mäkinen & Navarro 2004, Ferragina & Manzini 2000, Ferragina & Manzini 2001, Navarro 2004, Grossi et al. 2004). These structures typically require about 30% of the space of the text, and so double as a compression scheme as the original text can be discarded. Search times remain linear in the length of the pattern (assuming a fixed alphabet, such as ASCII).

While a great deal of effort has been expended in making suffix arrays smaller, there is still a fundamental problem with their scalability. When searching for a pattern P of length m, one must perform m non-sequential accesses into the suffix array, and m non-sequential access into the text. If the suffix array is on disk, this equates to $2m$ seek operations, which, for anything but small patterns (of the order of 5 characters), limits the technology to a small number of simultaneous users, or small texts that fit in RAM. Even the compressed, self-indexing suffix array of Grossi et al. (2004), which does not require access to the text, requires $O(m + \log n)$ seeks into the structure itself.

Because of the non-sequential access patterns exhibited by current suffix array algorithms, all papers experimenting with such algorithms assume that their structures can fit in memory. This seems to contradict bold claims that suffix arrays are an important technology for searching the World Wide Web, and even large genomic databases (Sadakane & Shibuya 2001, Grossi et al. 2004).

The inverted file, on the other hand, is a data structure that has been adopted by the Web search engine community, and handles data on external storage (Witten, Moffat & Bell 1999). Inverted files have been specifically engineered to scale well, and to minimise the number of expensive disk operations required to find a pattern in a text (Zobel & Moffat n.d.). However, the form of the pattern is restricted. With an inverted file, the form of the pattern must be set prior to index construction. Typically

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ACSC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology, Vol. 49. Gillian Dobbie and James Bailey, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

a word is chosen as the unit of indexing, restricting pattern search to words, prefixes of words, or combinations of words (phrases).

In this talk we will report on experiments with inverted files in direct competition to suffix arrays: that is, all data is in RAM, and arbitrary patterns are the target of the search.

Acknowledgments This work has been supported in part by grants from the Australian Research Council (Turpin) and the Natural Sciences and Engineering Research Council of Canada (Smyth).

References

Abouelhoda, M. I., Kurtz, S. & Ohlebusch, E. (2004), 'Replacing suffix trees with enhanced suffix arrays', *Journ. Discrete Algorithms* **2**, 53–86.

Boyer, R. S. & Moore, J. S. (1977), 'A fast string searching algorithm', *Communications of the ACM* **20**(10), 762–772.

Ferragina, P. & Manzini, G. (2000), Opportunistic data structures with applications, *in* 'Proceedings of the 41st IEEE Symposium on Foundations of Computer Science (FOCS 00)', IEEE Computer Society, Redondo Beach, CA, pp. 390–398.

Ferragina, P. & Manzini, G. (2001), An experimental study of an opportunistic index, *in* 'SODA '01: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms', Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 269–278.

Grossi, R. & Vitter, J. S. (n.d.), 'Compressed suffix arrays and suffix trees with applications to text indexing and string matching', *SIAM Journal on Computing* . To appear. Available from `http://www.cs.duke.edu/~jsv/Papers/catalog/node87.html`.

Grossi, R., Vitter, J. S. & Gupta, A. (2004), When indexing equals compression: Experiments with compressing suffix arrays and applications, *in* 'SODA '04: Proceedings of the fifteenth annual ACM-SIAM Symposium on Discrete algorithms', SIAM, New Orleans, Louisianna, USA, pp. 636–645.

Knuth, D. E., Morris, J. H. & Pratt, V. R. (1977), 'Fast pattern matching in strings', *SIAM Journal on Computing* **6**(2), 323–350.

Mäkinen, V. (2000), Compact suffix array, *in* 'Combinatorial Pattern Matching', Vol. LNCS 1848, pp. 305–319.

Mäkinen, V. & Navarro, G. (2004), Compressed compact suffix arrays, *in* 'Combinatorial Pattern Matching: 15th Annual Symposium, CPM 2004', Vol. LNCS 3109, Springer-Verlag GmbH, pp. 420–433.

Mäkinen, V. & Navarro, G. (2005), 'Succinct suffix arrays based on run-length encoding', *Nordic Journal of Computing* **12**(2), 40–66.

Manber, U. & Myers, G. (1990), Suffix arrays: a new method for on-line string searches, *in* 'SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms', Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 319–327.

Manber, U. & Myers, G. W. (1993), 'Suffix arrays: a new model for on-line string searches', *SIAM Journal of Computing* **22**(5), 935–948.

Navarro, G. (2004), 'Indexing text using the Ziv-Lempel trie', *Journal of Discrete Algorithms* **2**(1), 87–114.

Puglisi, S. J., Turpin, A. H. & Smyth, W. F. (2005*a*), The performance of linear time suffix sorting algorithms, *in* M. Cohn & J. Storer, eds, 'Proceedings of the IEEE Data Compression Conference', IEEE Computer Society Press, Los Alamitos, CA, pp. 358–368.

Puglisi, S. J., Turpin, A. H. & Smyth, W. F. (2005*b*), A taxonomy of suffix array construction algorithms, *in* 'Proceedings of the Prague Stringology Conference', Czech Technical University, Prague, pp. 1–30.

Sadakane, K. (2000), Compressed text databases with efficient query algorithms based on the compressed suffix array, *in* 'Proceedings of ISSAC'00', Vol. LNCS 1969, pp. 410–421.

Sadakane, K. (2002), Succinct representations of lcp information and improvements in the compressed suffix arrays, *in* 'SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms', Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 225–232.

Sadakane, K. & Shibuya, T. (2001), 'Indexing huge genome sequences for solving various problems', *Genome Informatics* **12**, 175–183.

Sim, J. S., Kim, D. K., Park, H. & Park, K. (2003), Linear-time search in suffix arrays, *in* 'Proc. 14th Australian Workshop Combinatorial Alg. (AWOCA)', pp. 139–146.

Witten, I. H., Moffat, A. & Bell, T. C. (1999), *Managing Gigabytes: Compressing and Indexing Documents and Images*, second edn, Morgan Kaufmann Publishing, San Francisco.

Zobel, J. & Moffat, A. (n.d.), 'Inverted files for text search engines'. Submitted for publication.

Full Papers

Horn Clauses and Functional Dependencies in Complex-value Databases

Sven Hartmann, Sebastian Link†

Information Science Research Centre, Massey University
Private Bag 11222, Palmerston North, New Zealand
e-mail: [s.hartmann,s.link]@massey.ac.nz

Abstract

We extend Fagin's result on the equivalence between functional dependencies in relational databases and propositional Horn clauses. It is shown that this equivalence still holds for functional dependencies in databases that support complex values via nesting of records, lists, sets and multisets.
The equivalence has several implications. Firstly, it extends a well-known result from relational databases to databases which are not in first normal form. Secondly, it characterises the implication of functional dependencies in complex-value databases in purely logical terms. The database designer can take advantage of this equivalence to reduce database design problems to simpler problems in propositional logic. An algorithm is presented for such an application. Furthermore, relational database design tools can be reused to solve problems for complex-value databases.

Key Words: Logic in Databases, Functional Dependency, Implication Problem, Complex values, Horn clause

1 Introduction

Functional dependencies, first introduced by Codd (Codd 1970), are a fundamental and widely studied concept in relational database theory. The notion of a functional dependency is intuitively simple and is therefore often applied in practice. According to (Delobel & Adiba 1985) about two thirds of all uni-relational dependencies (dependencies over a single relation schema) defined in practice are functional dependencies. It is well-known that in some ways functional dependencies behave precisely the same as a certain well-studied subset of propositional logic. More precisely, Fagin has shown in (Fagin 1977) that a functional dependency σ is a consequence of a set Σ of functional dependencies that all apply to some relation schema if and only if all Horn clauses that correspond to σ are logical consequences of the Horn clauses that correspond to the functional dependencies in Σ.

Example 1.1. *Consider the relation schema*

LECTURE={*Class, Lecturer, Time, Room*}

together with the following set Σ of functional dependencies on LECTURE*:*

- *Class → Lecturer,*
- *Class, Time → Room,*
- *Lecturer, Time → Class, and*
- *Room, Time → Class.*

Suppose we would like to find out whether Room and Time together form a superkey for LECTURE. *That is, the functional dependency σ:*

Room, Time → Class, Lecturer

is implied by Σ. Fagin's result shows that this decision problem is equivalent to the problem of deciding whether both of the following propositional Horn clauses[1]*, represented in implicational form,*

- *(Room ∧ Time) ⇒ Class*
- *(Room ∧ Time) ⇒ Lecturer*

are logically implied by the following set Σ' of propositional Horn clauses:

- *Class ⇒ Lecturer,*
- *(Class ∧ Time) ⇒ Room,*
- *(Lecturer ∧ Time) ⇒ Class,*
- *(Room ∧ Time) ⇒ Class.*

It is not difficult to see that the answer to these equivalent decision problems is affirmative. To get a little bit more insight into the relationship between Horn clauses and functional dependencies we look at a further example. The functional dependency σ

Class, Lecturer, Room → Time

is not implied by Σ. A counterexample to this implication is given, for instance, by the following two tuple relation r

t_1*=(Databases, H. Simpson, 2:30pm, 3.12),*
t_2*=(Databases, H. Simpson, 4:30pm, 3.12).*

While r satisfies all functional dependencies in Σ it does not satisfy σ. Let's look at the corresponding problem in terms of Horn clauses. In fact, the truth assignment θ that assigns true to the propositional variables Class, Lecturer and Room, and false to the variable Time makes all Horn clauses in Σ' true but leaves the Horn clause

(Class ∧ Lecturer ∧ Room) ⇒ Time

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC), Hobart, Tasmania. Conferences in Research and Practice in Information Technology, Vol. 49. Gill Dobbie and James Bailey, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

† Sebastian Link was supported by Marsden Funding, Royal Society of New Zealand

[1] The attributes of LECTURE are now used as propositional variables.

false. The point is here that the truth assignment θ assigns true to precisely those variables whose corresponding attributes have the same value in the counterexample relation $r = \{t_1, t_2\}$ above. □

Due to this result it is possible to take advantage of artificial intelligence research in the area of theorem proving by converting results in that area into results about relational functional dependencies. The equivalence theorem has had extensions to more general classes of dependencies (Sagiv, Delobel, Parker Jr. & Fagin 1981, Fagin 1982) and resulted in several applications (Sagiv 1980, Parker & Delobel 1979, Delobel & Parker 1978).

Many researchers have remarked that classical database design problems need to be revisited in new data formats (Suciu 2001, Vianu 2001, Vincent 1999). Biskup (Biskup 1998) has listed two particular challenges for database design: finding a unifying framework and extending achievements to deal with advanced database features such as complex data constructors. One possibly unifying framework can result from the classification of data models according to the data constructors that are supported by the model. The relational data model can be captured by a single application of the record constructor, arbitrary nesting of record and set constructor covers aggregation and grouping which are fundamental to many semantic data models as well as the nested relational data model (Hull & King 1987, Abiteboul, Hull & Vianu 1995). The Entity-Relationship Model and its extensions require record, set and (disjoint) union constructor (Chen 1976, Thalheim 2000). A minimal set of constructors supported by any object-oriented data model includes records, lists, sets and multisets (Atkinson, Bancilhon, DeWitt, Dittrich, Maier & Zdonik 1989). Genomic sequence data models call for support of records, lists and sets (Li, Ng & Wong 2002). Finally, XML requires at least record (concatenation), list (Kleene Closure), union (optionality), and reference constructor (Bray, Paoli, Sperberg-McQueen, Maler & Yergeau 2004). The following example illustrates the usage of complex data constructors and what type of dependencies may arise between complex data elements and what difficulties they impose.

Example 1.2. *Consider a simple example of a purchase profile that supermarkets and Online shops may utilise. A single entry consists of the name of the customer, a bag of items the customer bought, and the discount of this purchase received by the customer. Moreover, every item of the customer's bag consists of an article together with the price of that article. A database schema for such an application may look as follows*

Profile(Customer, Bag⟨Item(Article, Price)⟩, Discount).

An actual entry in the database may be

(Homer,⟨(Chocolate,3\$),(Chocolate,3\$),(Beer,4\$),(Beer,5\$)⟩,2\$).

Suppose that Homer received his discount of 2\$ since beer that costs 4\$ or more is on special. Intuitively, customers with the same bag of items should receive the same discount. This is an actual functional dependency that involves complex data objects, in this case a bag. The presence of the bag constructor shows some surprises. Consider for instance a second data element

(Bart,⟨(Chocolate,4\$),(Chocolate,5\$),(Beer,3\$),(Beer,3\$)⟩,0\$).

Bart bought the same bag of articles and has same bag of prices, yet did not receive any discount. This is actually consistent with respect to the functional dependency since Bart did not have the same bag of items as Homer. In order to receive a discount it matters which articles are bought to which price. □

The major goal of this paper is to generalise Fagin's Equivalence theorem to databases that support any combination of records, lists, sets and multisets. Our studies will be based on an abstract data model that defines a database schema as an arbitrarily nested attribute where nesting may refer to records, lists, sets and multisets. It is our intention not to focus on the specifics of any particular data model in order to place emphasis on the data constructors themselves. Functional dependencies have previously been defined in terms of subschemata of the underlying database schema (Hartmann, Link & Schewe 2006). Section 6 of (Hartmann et al. 2006) consists of a detailed comparison of our approach to previous work in various concrete data models such as the nested relational data model and XML. In essence, the expressiveness is complementary. Our approach leads to Brouwerian algebras (McKinsey & Tarski 1946) and provides therefore a mathematically well-founded framework that is sufficiently flexible and powerful to study design problems for different classes of constraints with respect to different combinations of data constructors. The presence of the data constructors, in particular that of set and multiset, requires a very detailed analysis in order to generalise the original proof from the relational data model. Most importantly, the results of this paper show that set and bag constructors must be handled differently from record and list constructor in order to capture the semantics of dependencies consistently.

2 An Abstract Data Model

Complex-value data models have been proposed to overcome severe limitations of the relational data model when designing many practical database applications (Abiteboul et al. 1995).

2.1 Database Schemata

We start with the definition of flat attributes and values for them. A *universe* is a finite set $\mathcal{U}$ together with domains (i.e. sets of values) *dom*(A) for all $A \in \mathcal{U}$. The elements of $\mathcal{U}$ are called *flat attributes.* Flat attributes will be denoted by upper-case characters from the start of the alphabet such as A, B, C etc.

In the following we will use a set $\mathcal{L}$ of labels, and assume that the symbol λ is neither a flat attribute nor a label, i.e., $\lambda \notin \mathcal{U} \cup \mathcal{L}$. Moreover, flat attributes are not labels and vice versa, i.e., $\mathcal{U} \cap \mathcal{L} = \emptyset$.

Database schemata in our data model will be given in form of nested attributes. Let $\mathcal{U}$ be a universe and $\mathcal{L}$ a set of labels. The set $\mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ of *nested attributes over* $\mathcal{U}$ *and* $\mathcal{L}$ is the smallest set satisfying the following conditions: $\lambda \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$, $\mathcal{U} \subseteq \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$, for $L \in \mathcal{L}$ and $N_1, \ldots, N_k \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ with $k \geq 1$ we have $L(N_1, \ldots, N_k) \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$, for $L \in \mathcal{L}$ and $N \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ we have $L[N], L\{N\}, L\langle N \rangle \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$. We call λ *null attribute*, $L(N_1, \ldots, N_k)$ *record-valued attribute*, $L[N]$ *list-valued attribute*, $L\{N\}$ *set-valued attribute* and $L\langle N \rangle$ *multiset-valued attribute.* From now on, we assume that a set $\mathcal{U}$ of attribute names, and a set $\mathcal{L}$ of labels is fixed, and write $\mathcal{N}\mathrm{A}$ instead of $\mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$. The null attriute λ is a distinguished attribute whose domain is the single null value which indicates that some information exists but has currently been left out. Some detailed explanations for the null attribute λ is offered later on.

Example 2.1. *The relation schema* Lecture *with the four flat attributes Class, Lecturer, Time and*

Room can be captured by the record-valued nested attribute

$$\textit{Lecture(Class,Lecturer,Time,Room)}.$$

More generally, a relation schema $R = \{A_1, \dots, A_k\}$ with flat attributes $A_1, \dots, A_k$ may be viewed as the record-valued attribute $R(A_1, \dots, A_k)$ using the name R of the relation schema as a label. □

Example 2.2. *Given flat attributes such as Customer, Article, Price and Discount, and labels such as Profile, Item and Bag, we may construct the record-valued attribute*

$$\textit{Item(Article, Price)},$$

the multiset-valued attribute

$$\textit{Bag}\langle\textit{Item(Article, Price)}\rangle,$$

and finally the record-valued attribute

$$\textit{Profile(Customer, Bag}\langle\textit{Item(Article, Price)}\rangle\textit{, Discount)}.$$

Using the null attribute λ we may also generate record-valued attributes such as

$$\textit{Profile(Customer, Bag}\langle\textit{Item(Article, }\lambda)\rangle,\ \lambda)$$

or

$$\textit{Profile(Customer, Bag}\langle\textit{Item(}\lambda,\ \lambda)\rangle\textit{, Discount)}.$$ □

We can extend the mapping *dom* from flat attributes to nested attributes, i.e., we define a set $dom(N)$ of possible data elements for every nested attribute $N \in \mathcal{N}\mathrm{A}$. For a nested attribute $N \in \mathcal{N}\mathrm{A}$ we define the *domain* $dom(N)$ as follows: $dom(\lambda) = \{ok\}$, $dom(A)$ for $A \in \mathcal{U}$ as before, $dom(L(N_1, \dots, N_k)) = \{(v_1, \dots, v_k) \mid v_i \in dom(N_i) \text{ for } i = 1, \dots, k\}$, i.e., the set of all k-tuples $(v_1, \dots, v_k)$ with $v_i \in dom(N_i)$ for all $i = 1, \dots, k$, $dom(L\{N\}) = \{\{v_1, \dots, v_n\} \mid v_i \in dom(N) \text{ for } i = 1, \dots, n\}$, i.e., the set of all finite sets with elements in $dom(N)$, $dom(L\langle N\rangle) = \{\langle v_1, \dots, v_n\rangle \mid v_i \in dom(N) \text{ for } i = 1, \dots, n\}$, i.e., the set of all finite multisets with elements in $dom(N)$, and $dom(L[N]) = \{[v_1, \dots, v_n] \mid v_i \in dom(N) \text{ for } i = 1, \dots, n\}$, i.e., the set of all finite lists with elements in $dom(N)$. The empty set, multiset and list are denoted by $\emptyset$, $\langle\ \rangle$, and $[\ ]$, respectively. The value *ok* can be interpreted as the null value "some information exists, but is currently omitted".

Example 2.3. *Consider the nested attribute*

$$\textit{Lecture(Class,Lecturer,Time,Room)}.$$

The 4-tuple

$$\textit{(Databases, H.Simpson, 1pm, 3.12)}$$

is an element from the domain of this nested attribute. More generally, the domain of a record-valued attribute $R(A_1, \dots, A_k)$ with flat attributes $A_1, \dots, A_k$ is the set of all k-tuples composed out of elements from the corresponding domains $dom(A_i)$ of the flat attributes A_i. In other words, the record-valued attribute $R(A_1, \dots, A_k)$ represents indeed a relation schema. □

Example 2.4. *The data element*

$$\textit{(Homer,}\langle\textit{(Chocolate,3\\$),(Chocolate,3\\$),(Beer,4\\$),(Beer,5\\$)}\rangle\textit{,2\\$)}$$

is from the domain of

$$\textit{Profile(Customer, Bag}\langle\textit{Item(Article, Price)}\rangle\textit{, Discount)}.$$

Moreover, the data element

$$\textit{(Homer,}\langle\textit{(ok,ok),(ok,ok),(ok,ok),(ok,ok)}\rangle\textit{,2\\$)}.$$

is from the domain of

$$\textit{Profile(Customer, Bag}\langle\textit{Item(}\lambda,\ \lambda)\rangle\textit{, Discount)}.$$ □

2.2 Subschemata

The replacement of some attribute names by the null attribute λ within a nested attribute decreases the amount of information that is modelled by the corresponding schema. This fact allows to introduce an order between database schemata.

The *subattribute relation* $\leq$ on the set of nested attributes $\mathcal{N}\mathrm{A}$ over $\mathcal{U}$ and $\mathcal{L}$ is defined by the following rules, and the following rules only: $N \leq N$, $\lambda \leq A$ for all flat attributes $A \in \mathcal{U}$, $\lambda \leq N$ for all list-valued attributes N, $L(N_1, \dots, N_k) \leq L(M_1, \dots, M_k)$ whenever $N_i \leq M_i$ for all $i = 1, \dots, k$, and $L[N] \leq L[M]$ whenever $N \leq M$. For N, M we say that M is a *subattribute* of N if and only if $M \leq N$ holds. We write $M \not\leq N$ if M is not a subattribute of N, and $M < N$ in case $M \leq N$ and $M \neq N$.

Lemma 2.1. *The subattribute relation is a partial order on nested attributes.* ■

The subattribute relationship between nested attributes generalises the inclusion relationship between sets of attributes in the relational data model.

Example 2.5. *Consider again the record-valued attribute*

$$\textit{Lecture(Class,Lecturer,Time,Room)}.$$

There is a bijection between attribute sets of the relation schema LECTURE *and the subattributes of Lecture(Class,Lecturer,Time,Room). The empty attribute set $\emptyset$, for instance, corresponds to the subattribute*

$$\textit{Lecture}(\lambda,\lambda,\lambda,\lambda).$$

The attribute set $\{$Lecturer,Room$\}$ is correspondent to the subattribute

$$\textit{Lecture}(\lambda,\textit{Lecturer},\lambda,\textit{Room}).$$

We have now seen in several examples that a relation schema is a special case of a database schema represented by a nested attribute. In fact, such a relation schema can be captured by a single application of the record constructor to a finite set of flat attributes. □

Another example of a subattribute is given by

$$\text{Profile(Customer, Bag}\langle\text{Item}(\lambda,\ \lambda)\rangle\text{, Discount)}$$

which is a subattribute of

$$\text{Profile(Customer, Bag}\langle\text{Item(Article, Price)}\rangle\text{, Discount)}.$$

Informally, M is a subattribute of N if and only if M comprises at most as much information as N does. The informal description of the subattribute relation is formally documented by the existence of a projection function $\pi^N_M : dom(N) \to dom(M)$ in case $M \leq N$ holds. For $M \leq N$ the *projection function* $\pi^N_M : dom(N) \to dom(M)$ is defined as follows:

- if $N = M$, then $\pi^N_M = id_{dom(N)}$ is the identity on $dom(N)$,
- if $M = \lambda$, then $\pi^N_\lambda : dom(N) \to \{ok\}$ is the constant function that maps $v \in dom(N)$ to *ok*,
- if $N = L(N_1, \dots, N_k)$ and $M = L(M_1, \dots, M_k)$, then $\pi^N_M = \pi^{N_1}_{M_1} \times \cdots \times \pi^{N_k}_{M_k}$ maps the tuple $(v_1, \dots, v_k) \in dom(N)$ to $(\pi^{N_1}_{M_1}(v_1), \dots, \pi^{N_k}_{M_k}(v_k)) \in dom(M)$,
- if $N = L\{N'\}$ and $M = L\{M'\}$, then $\pi^N_M : dom(N) \to dom(M)$ maps the set $\{v_1, \dots, v_n\} \in dom(N)$ to $\{\pi^{N'}_{M'}(v_1), \dots, \pi^{N'}_{M'}(v_n)\} \in dom(M)$.

- if $N = L\langle N'\rangle$ and $M = L\langle M'\rangle$, then $\pi^N_M : dom(N) \to dom(M)$ maps the multiset $\langle v_1,\dots,v_n\rangle \in dom(N)$ to $\langle \pi^{N'}_{M'}(v_1),\dots,\pi^{N'}_{M'}(v_n)\rangle \in dom(M)$, and
- if $N = L[N']$ and $M = L[M']$, then $\pi^N_M : dom(N) \to dom(M)$ maps the list $[v_1,\dots,v_n] \in dom(N)$ to $[\pi^{N'}_{M'}(v_1),\dots,\pi^{N'}_{M'}(v_n)] \in dom(M)$.

The projection function tells us precisely how to map a database instance of some schema to an instance of any of its subschemata.

Example 2.6. *Consider the 4-tuple*

$$t = \text{(Databases, H.Simpson, 1pm, 3.12)}$$

from the domain of

$$N = \textit{Lecture(Class, Lecturer, Time, Room)}.$$

The projection $\pi^N_X(t)$ of t from N to the subattribute

$$X = \textit{Lecture}(\lambda, \textit{Lecturer}, \lambda, \textit{Room})$$

is

$\pi^N_X(t) =$ *(ok, H.Simpson, ok, 3.12).* □

Example 2.7. *The following data element t*

(Homer,⟨(Chocolate,3\$),(Chocolate,3\$),(Beer,4\$),(Beer,5\$)⟩,2\$)

from the domain of

$N =$ *Profile(Customer, Bag⟨Item(Article, Price)⟩, Discount)*

has projection

$\pi^N_X(t) =$ *(Homer,⟨(ok,ok),(ok,ok),(ok,ok),(ok,ok)⟩,2\$).*

where

$X =$ *Profile(Customer, Bag⟨Item(λ, λ)⟩, Discount).* □

2.3 Brouwerian algebra of Subattributes

The relational data model is based on the powerset $\mathcal{P}(R)$ for a relation schema R. In fact, $\mathcal{P}(R)$ is a powerset algebra with partial order $\subseteq$, set union $\cup$, set intersection $\cap$ and set difference $-$. We will now extend these operations to nested attributes. The inclusion order $\subseteq$ has already been generalised by the subattribute relationship $\leq$. The set $Sub(N)$ of *subattributes* of N is $Sub(N) = \{M \mid M \leq N\}$.

We study the algebraic structure of the poset $(Sub(N), \leq)$. A *Brouwerian algebra* (McKinsey & Tarski 1946) is a lattice $(L, \sqsubseteq, \sqcup, \sqcap, \dot{-}, 1)$ with top element 1 and a binary operation $\dot{-}$ which satisfies $a \dot{-} b \sqsubseteq c$ iff $a \sqsubseteq b \sqcup c$ for all $c \in L$. In this case, the operation $\dot{-}$ is called the *pseudo-difference*. The *Brouwerian complement* $\neg a$ of $a \in L$ is then defined by $\neg a = 1 \dot{-} a$. A Brouwerian algebra is also called a co-Heyting algebra or a dual Heyting algebra. The system of all closed subsets of a topological space is a well-known Brouwerian algebra, see (McKinsey & Tarski 1946). The definition of the subattribute relationship $\leq$ completely determines the operations of join, meet and pseudo-difference. The following theorem generalises the fact that $(\mathcal{P}(R), \subseteq, \cup, \cap, -, \emptyset, R)$ is a Boolean algebra for a relation schema R in the RDM.

Theorem 2.1. $(Sub(N), \leq, \sqcup_N, \sqcap_N, \dot{-}_N, N)$ *forms a Brouwerian algebra for every* $N \in \mathcal{N}\mathrm{A}$. ■

The nested attribute N is the top element of $(Sub(N), \leq)$. The *bottom element* λ_N of $Sub(N)$ is given by $\lambda_N = L(\lambda_{N_1},\dots,\lambda_{N_k})$ whenever $N = L(N_1,\dots,N_k)$, and $\lambda_N = \lambda$ whenever N is not a record-valued attribute.

In order to simplify notation, occurrences of λ in a record-valued attribute are usually omitted if this does not cause any ambiguities. That is, the subattribute $L(M_1,\dots,M_k) \leq L(N_1,\dots,N_k)$ is abbreviated by $L(M_{i_1},\dots,M_{i_l})$ where $\{M_{i_1},\dots,M_{i_l}\} = \{M_j : M_j \neq \lambda_{N_j}$ and $1 \leq j \leq k\}$ and $i_1 < \cdots < i_l$. If $M_j = \lambda_{N_j}$ for all $j = 1,\dots,k$, then we use λ instead of $L(M_1,\dots,M_k)$. The subattribute $L(A,\lambda)$ of $L(A,A)$ cannot be abbreviated by $L(A)$ since this may also refer to $L(\lambda, A)$.

Example 2.8. *The nested attribute*

Profile(Customer, Bag⟨Item(Article, λ)⟩, λ)

is abbreviated by

Profile(Customer, Bag⟨Item(Article)⟩).

The nested attribute

Profile(λ, Bag⟨Item(λ, λ)⟩, Discount)

is abbreviated by

Profile(Bag⟨λ⟩, Discount). □

If the context allows, we omit the index N from the operations $\sqcup_N, \sqcap_N, \dot{-}_N$ and from λ_N.

2.4 Order, Multiplicity and The Null Attribute

We give some more explanations on the null attribute λ. From an algebraic point of view it is simply the bottom element $N \dot{-} N$ of the Brouwerian algebra carried by N. As already seen, replacing occurrences of nested attributes by the null attribute according to the rules of the subattribute relationship results in a subattribute and therefore in a decrease of the amount of information that can be modelled. The null attribute therefore allows to obtain different layers of information generating ultimately the structure of a Brouwerian algebra for a fixed database schema. However, the null attribute also offers some interesting features for database modelling, depending on the presence of certain complex objects. Consider for instance the nested attribute *Speak(Person,Foreign[Language])* which is used to store the list of foreign languages a person speaks (ordered according to some preference). Two elements from the corresponding domain could be (Bernhard,[Russian,English,French]) and (John,[ ]). The projections of these elements on the subattribute *Speak(Person,Foreign[λ])* are (Bernhard,[ok,ok,ok]) and (John,[ ]) still revealing that Bernhard speaks 3 foreign languages and John speaks none. Suppose that instead of using the list-valued attribute *Foreign[Language]* we used a set-valued attribute *Foreign{Language}*, i.e., we are only interested in the foreign languages a person speaks, and not in any preferences for these languages. The element (Bernhard,{Russian,English,French}) is mapped to (Bernhard,{ok}), and the element (John,∅) is mapped to itself. Therefore, the subattribute *Speak(Person,Foreign{λ})* reveals whether a person speaks a foreign language at all. The feature of storing the same data repeatedly therefore enables *counting*, i.e., is supported by lists and multisets, but not by sets.

The second feature is the ability to model order which is supported by lists, but not by sets nor multisets.

This property implies that the projections of any tuple on two subattributes X and Y always determine the projection of that tuple on the join $X \sqcup Y$. In case of set or multiset constructor this property is not valid anymore. Consider for instance the set-valued attribute *Duo*$\{$*Pair(Girl,Boy)*$\}$ which represents sets of dancing couples. A tuple might be $\{$(Don Quixote, Theresa), (Sancho Pansa, Dulcinea)$\}$ and the second tuple $\{$(Don Quixote, Dulcinea), (Sancho Pansa, Theresa)$\}$ results from switching partners. Both tuples coincide in their projection on *Duo*$\{$*Pair*$(\lambda,$ *Boy*$)\}$ as they evaluate to $\{$(ok, Don Quixote), (ok, Sancho Pansa)$\}$) and coincide in their projection on *Duo*$\{$*Pair(Girl,*$\lambda)\}$ as they evaluate to $\{$(Dulcinea, ok), (Theresa, ok)$\}$, but they differ on the join *Duo*$\{$*Pair(Girl,Boy)*$\}$.

2.5 Subattribute Basis

Let $\mathcal{T}$ be some collection of data constructors and N an arbitrary nested attribute composed of data constructors in $\mathcal{T}$ only. What is the minimal set $\mathfrak{A}_{\mathcal{T}}(N) \subseteq Sub(N)$ such that every element $t \in dom(N)$ is uniquely determined by its projections $\{\pi^N_A(t) \mid A \in \mathfrak{A}_{\mathcal{T}}(N)\}$?

Consider the simplest of all cases where $\mathcal{T}$ consists of the record constructor only. Suppose further that nested attributes are generated from flat attributes by a single application of the record constructor. That is, $N = L(A_1, \ldots, A_k)$ for flat attributes $A_1, \ldots, A_k$. This is just a different notation for the relation schema $R = \{A_1, \ldots, A_k\}$. Now, every tuple t over R (or $t \in dom(N)$, respectively) is completely determined by its projections $\{\pi^N_{A_1}(t), \ldots, \pi^N_{A_k}(t)\}$. In fact, in order to store a tuple we store its values on the individual attributes. From an algebraic point of view the subattributes $L(A_1, \lambda, \ldots, \lambda), \ldots, L(\lambda, \ldots, \lambda, A_k)$ are the join-irreducible elements of $(Sub(N), \leq, \sqcup, \sqcap, \lambda_N)$. Recall that an element a of a lattice with bottom element 0 is called *join-irreducible* if and only if $a \neq 0$ and if $a = b \sqcup c$ holds for any elements b and c, then $a = b$ or $a = c$. The *subattribute basis* of N, denoted by $\mathcal{B}(N)$, is the set of join-irreducible elements of $(Sub(N), \leq, \sqcup, \sqcap, \lambda_N)$. Every element of $\mathcal{B}(N)$ is called a *basis attribute of* N. A basis attribute $X \in \mathcal{B}(N)$ is called *maximal* if and only if $X \leq Y$ for any basis attribute $Y \in \mathcal{B}(N)$ implies that $X = Y$ holds. Basis attributes that are not maximal are called *non-maximal.*

Consider now the case where $\mathcal{T}$ consists of record and list constructor, i.e., N may be generated from flat attributes by finitely many recursive applications of record and list constructor. One can show that for any $t_1, t_2 \in dom(N)$ and for any $X, Y \in Sub(N)$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$ and $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ also $\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$ holds. That is, the two projections of a tuple on two subattributes X and Y uniquely determine the projection of that tuple on the join $X \sqcup Y$. This shows, in particular, that $\mathfrak{A}_{\mathcal{T}}(N)$ is still the set of join-irreducible elements of $(Sub(N), \leq, \sqcup, \sqcap, \lambda_N)$.

If we add set or multiset constructor to $\mathcal{T}$, then it becomes insufficient to consider join-irreducible elements. In fact, there is some nested attribute N and distinct elements of $dom(N)$ which agree on all projections to basis attributes of N. We have already seen such a nested attribute N and two such tuples, namely *Duo*$\{$*Pair(Girl,Boy)*$\}$ and $\{$(Don Quixote, Theresa), (Sancho Pansa, Dulcinea)$\}$ as well as $\{$(Don Quixote, Dulcinea), (Sancho Pansa, Theresa)$\}$. A further example is the nested attribute *Bag*$\langle$*Item(Article, Price)*$\rangle$ and the two tuples

(Homer,$\langle$(Chocolate,3\$),(Chocolate,3\$),(Beer,4\$),(Beer,5\$)$\rangle$,2\$)

and

(Bart,$\langle$(Chocolate,4\$),(Chocolate,5\$),(Beer,3\$),(Beer,3\$)$\rangle$,0\$).

In the presence of set and multiset constructor we face the difficulty of characterising those pairs of subattributes X and Y of N for which $t_1, t_2 \in dom(N)$ exist such that $\pi^N_X(t_1) = \pi^N_X(t_2)$, $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ and $\pi^N_{X \sqcup Y}(t_1) \neq \pi^N_{X \sqcup Y}(t_2)$. In other words, what subattributes of N (other than the basis attributes) are necessary to uniquely determine every single tuple over N? What values need to be stored to identify a tuple uniquely? The following definition is used to answer these questions.

Definition 2.1. *Let $N \in \mathcal{NA}$. The subattributes $X, Y \in Sub(N)$ are reconcilable if and only if one of the following conditions is satisfied 1) $Y \leq X$ or $X \leq Y$, 2) $N = L(N_1, \ldots, N_k), X = L(X_1, \ldots, X_k), Y = L(Y_1, \ldots, Y_k)$ where X_i and Y_i are reconcilable for all $i = 1, \ldots, k$ or 3) $N = L[N'], X = L[X'], Y = L[Y']$ where X' and Y' are reconcilable.* □

If N is *Duo*$\{$*Pair(Girl, Boy)*$\}$, and X and Y are *Duo*$\{$*Pair(Girl,* $\lambda)\}$ and *Duo*$\{$*Pair*$(\lambda,$ *Boy*$)\}$, respectively, then X and Y are not reconcilable.

Theorem 2.2. *Let $N \in \mathcal{NA}$. For all $X, Y \in Sub(N)$ we have that X and Y are reconcilable if and only if for all $t, t' \in dom(N)$ with $\pi^N_X(t) = \pi^N_X(t')$ and $\pi^N_Y(t) = \pi^N_Y(t')$ also $\pi^N_{X \sqcup Y}(t) = \pi^N_{X \sqcup Y}(t')$ holds.* ■

It still remains to clarify which projections are necessary and sufficient to identify every tuple over a nested attribute generated by finitely many applications of record, list, set and multiset constructor.

Definition 2.2. *Let $N \in \mathcal{NA}$. The extended subattribute basis $\mathcal{E}(N) \subseteq Sub(N)$ is the smallest set with the properties that $\mathcal{B}(N) \subseteq \mathcal{E}(N)$, and that for all $X, Y \in \mathcal{E}(N)$ which are not reconcilable also $X \sqcup Y \in \mathcal{E}(N)$ holds.*□

The extended subattribute basis is therefore the smallest set that contains the subattribute basis and that is closed under the join of subattributes that are not reconcilable. In the absence of sets and multisets we have $\mathcal{E}(N) = \mathcal{B}(N)$ since every pair of subattributes is reconcilable. If N is a set- or multiset-valued attribute, then $\mathcal{E}(N) = Sub(N)$. If $\mathcal{T}$ consists of records, lists, sets and multisets, then $\mathfrak{A}_{\mathcal{T}}(N) = \mathcal{E}(N)$. This seems now very natural: for any two subattributes $X, Y \in \mathcal{E}(N)$ for which the two projections $\pi^N_X(t)$ and $\pi^N_Y(t)$ do not determine the value of $\pi^N_{X \sqcup Y}(t)$, the subattributes X and Y cannot be reconcilable, and $X \sqcup Y$ is therefore included in $\mathcal{E}(N)$.

There is a relatively simple way to reduce the notion of reconcilability to the notion of comparability with respect to $\leq$. The idea is to choose the *units* U of N such that for all subattributes $V, W \in Sub(N)$ we have that V and W are reconcilable if and only if $V \sqcap U$ and $W \sqcap U$ are comparable with respect to $\leq$ for all units U of N. To spell this out, two subattributes X, Y are comparable with respect to $\leq$ if and only if $X \leq Y$ or $Y \leq X$ holds. This property is achieved by the following definition.

Definition 2.3. *Let $N \in \mathcal{NA}$. A nested attribute $U \in \mathcal{NA}$ is a unit of N if and only if 1) $U \in Sub(N)$, and 2) $\forall X, Y \leq U$ if X and Y are reconcilable, then $X \leq Y$ or $Y \leq X$, and 3) U is $\leq$-maximal with the properties 1) and 2). The set of all units of N is denoted by $\mathcal{U}(N)$.* □

Example 2.9. *The units of*

$$\textit{Profile(Customer, Bag}\langle\textit{Item(Article, Price)}\rangle\textit{, Discount)}$$

are

$$\begin{aligned} U_1 &= \textit{Profile(Customer)},\\ U_2 &= \textit{Profile(Bag}\langle\textit{Item(Article, Price)}\rangle\textit{)}, \text{ and}\\ U_3 &= \textit{Profile(Discount)}. \end{aligned}$$

The two subattributes U_1 and U_3 are reconcilable. In fact, for every $U \in \{U_1, U_2, U_3\}$ we have $U_1 \sqcap U \leq U_3 \sqcap U$ or $U_3 \sqcap U \leq U_1 \sqcap U$. However, the subattributes

$$\begin{aligned} X &= \textit{Profile(Bag}\langle\textit{Item(Article)}\rangle\textit{)}, \text{ and}\\ Y &= \textit{Profile(Bag}\langle\textit{Item(Price)}\rangle\textit{)} \end{aligned}$$

are not reconcilable. In fact, $X = X \sqcap U_2$ and $Y = Y \sqcap U_2$ are incomparable with respect to $\leq$. □

2.6 Functional Dependencies

The following definition is a natural extension of the definition of FDs from the relational data model.

Definition 2.4. *Let $N \in \mathcal{NA}$ be a nested attribute. A functional dependency on N is an expression of the form $\mathcal{X} \rightarrow \mathcal{Y}$ where $\mathcal{X}, \mathcal{Y} \subseteq Sub(N)$ are non-empty. A set $r \subseteq dom(N)$ satisfies the FD $\mathcal{X} \rightarrow \mathcal{Y}$ on N, denoted by $\models_r \mathcal{X} \rightarrow \mathcal{Y}$, if and only if for all $t_1, t_2 \in r$ we have $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ for all $Y \in \mathcal{Y}$ whenever $\pi^N_X(t_1) = \pi^N_X(t_2)$ holds for all $X \in \mathcal{X}$.* □

In case a set of subattributes is the singleton $\{X\}$ we also write simply X instead of $\{X\}$. For $\mathcal{X} \subseteq Sub(N)$ let $\vartheta(\mathcal{X}) = \max_{\leq}\{Y \in \mathcal{E}(N) \mid Y \leq X \text{ for some } X \in \mathcal{X}\}$, i.e., $\vartheta(X)$ contains all the extended basis attributes of N which are subattributes of some element of X and which are $\leq$-maximal with this property. It is not difficult to see that an instance $r \subseteq dom(N)$ satisfies $\mathcal{X} \rightarrow \mathcal{Y}$ if and only if r satisfies $\vartheta(\mathcal{X}) \rightarrow \vartheta(\mathcal{Y})$. Therefore we can assume without loss of generality that every FD $\mathcal{X} \rightarrow \mathcal{Y}$ is of the form $\mathcal{X} = \vartheta(\mathcal{X})$ and $\mathcal{Y} = \vartheta(\mathcal{Y})$.

Example 2.10. *Consider again the nested attribute*

$$\textit{Profile(Customer, Bag}\langle\textit{Item(Article, Price)}\rangle\textit{, Discount)}.$$

Intuitively, the customers who bought the same bag of items should receive the same discount. Formally, this constraint can be specified as the functional dependency

$$\textit{Profile(Bag}\langle\textit{Item(Article, Price)}\rangle\textit{)} \rightarrow \textit{Profile(Discount)}.$$

Note that this FD is completely different from the FD

$$\{\textit{Profile(Bag}\langle\textit{Item(Article)}\rangle\textit{)}, \textit{Profile(Bag}\langle\textit{Item(Price)}\rangle\textit{)}\} \rightarrow \textit{Profile(Discount)}$$

as demonstrated by the two tuples

$$t_1 = (\textit{Homer}, \langle(\textit{Chocolate}, 3\$), (\textit{Chocolate}, 3\$), (\textit{Beer}, 4\$), (\textit{Beer}, 5\$)\rangle, 2\$)$$

and

$$t_2 = (\textit{Bart}, \langle(\textit{Chocolate}, 4\$), (\textit{Chocolate}, 5\$), (\textit{Beer}, 3\$), (\textit{Beer}, 3\$)\rangle, 0\$).$$

In fact, $\{t_1, t_2\}$ satisfy the first FD, but do not satisfy the second FD. □

Let Σ be a set of FDs, and σ a single FD, all defined on some nested attribute N. We say that Σ (finitely) implies σ, denoted by $\Sigma \models \sigma$ ($\Sigma \models_{\text{fin}} \sigma$) if and only if all (finite) $r \subseteq dom(N)$ that satisfy all FDs in Σ also satisfy σ. Furthermore, Σ implies σ in the world of two-element instances if and only if all $r = \{t_1, t_2\} \subseteq dom(N)$ that satisfy all dependencies in Σ also satisfy σ. It is not difficult to see that finite and unrestricted implication coincide for FDs. As it will turn out in this paper, (finite) implication even coincides with implication in the world of two-element instances.

We will now describe the equivalence between the implication of FDs and the logical implication of propositional Horn clauses. To do so we repeat some basic notions regarding boolean propositional logic, and fix some notation.

3 The Equivalence

A *literal* is either a propositional variable V (a positive literal) or the negation $\neg V$ of a propositional variable V (a negative literal). A *Horn clause* is a non-empty disjunction of literals, with at most one positive literal. The Horn clause $\neg V_1 \vee \cdots \vee \neg V_m \vee W$ is represented in implicational form as $V_1 \wedge \cdots \wedge V_m \Rightarrow W$ where the empty conjunction is read as *true*. An implicational statement $V_1 \wedge \cdots \wedge V_m \Rightarrow W_1 \wedge \cdots \wedge W_n$ with $n \geq 1$ and positive literals $W_1, \ldots, W_n$ is equivalent to the n Horn clauses

$$V_1 \wedge \cdots \wedge V_m \Rightarrow W_1, \ldots, V_1 \wedge \cdots \wedge V_m \Rightarrow W_n.$$

Fagin shows in (Fagin 1977) that the FD $A_1 \cdots A_m \rightarrow B_1 \cdots B_m$ is a consequence of a set Σ of FDs on a relation schema R if and only if the corresponding Horn clauses

$$A_1 \wedge \cdots \wedge A_m \Rightarrow B_1, \ldots, A_1 \wedge \cdots \wedge A_m \Rightarrow B_m$$

are logically implied by the corresponding Horn clauses of Σ. The attributes of relation schema R are therefore interpreted as propositional variables. It is shown that the implication of FDs over arbitrary relations is equivalent to the implication of FDs over two-tuple relations. Given a two-tuple relation over R, the truth value of a propositional variable A is assigned *true* if and only if those two tuples agree on the attribute A.

We would like to generalise this result to FDs in complex-value databases including records, lists, sets and multisets. The presence of these data constructors causes significant problems in generalising the original proof. The first problem is to choose which subattributes of the underlying nested attribute N are to be interpreted as propositional truth variables. In general, the right choice is to interpret the elements of $\mathfrak{A}_{\mathcal{T}}(N)$ as propositional variables. That is, the elements of the extended subattribute basis $\mathcal{E}(N)$ do both, generalise the result by Fagin, and do justice to the presence of sets and multisets.

A second problem is caused by the subattribute relationship $\leq$. While attributes in a relation schema form an anti-chain with respect to inclusion, the elements of the extended subattribute basis are partially ordered by $\leq$. If two tuples agree on some extended basis attribute U and $V \leq U$, then they will also agree on V. As the structure of the database schema N is fixed, this results in a fixed set of Horn clauses that need to be satisfied independently from the given set of constraints. That is, Horn clauses are not only used to encode the dependencies, but also the structure of the underlying database schema.

Given an arbitrary truth assignment to the propositional variables, do there always exist two tuples which precisly agree on those extended basis attributes that are assigned the truth value *true*? While the answer is negative in general, it is actually sufficient to look for an affirmative answer in the presence of those Horn clauses which encode the database schema. It comes then down to showing that there are in general two tuples which precisely agree on the elements of a $\leq$-downward closed set of subattributes which is closed under the join of reconcilable attributes.

Before we describe the equivalence in general we will illustrate the basic ideas on an example. It shows the two different ways of reasoning which we will show to be equivalent.

3.1 An Example

Consider the nested attribute

$$N = \text{Dance(Time,Partaker\{Name\},Duo\{Pair(Girl,Boy)\},Rating)}$$

together with the following set Σ of FDs

- Dance(Time) → Dance(Partaker{Name},Duo{Pair(Girl,Boy)},Rating),
- Dance(Partaker{Name}) → {Dance(Duo{Pair(Girl)}),Dance(Duo{Pair(Boy)})},
- {Dance(Duo{Pair(Girl)}),Dance(Duo{Pair(Boy)})} → Dance(Partaker{Name}), and
- Dance(Duo{Pair(Girl,Boy)}) → Dance(Rating).

N models dancing classes in which partakers are grouped into pairs of girls and boys. The first FD says informally that the time of the course determines everything else, i.e., Dance(Time) is a key. The second FD says informally that the set of participants determines the set of boys and the set of girls. Vice versa, the third FD says informally that the set of girls and the set of boys together determine the set of participants. Finally, the last FD says informally that the set of dancing combinations determines the rating. That is, the rating for each class depends on the combination of the dancing partners. Suppose we want to decide if the single FD σ = Dance(Partaker{Name}) → Dance(Rating) is a consequence of Σ. The nested two-tuple relation r consisting of

t_1 = (29.2.1600,{Dulcinea, Theresa, Don Quixote, Sancho},
{(Dulcinea, Don Quixote),(Theresa, Sancho)},10) and
t_2 = (1.3.1600,{Dulcinea, Theresa, Don Quixote, Sancho},
{(Dulcinea, Sancho),(Theresa, Don Quixote)},3)

satisfies all FDs in Σ, but violates σ. We have therefore found a counterexample relation r, and Σ does therefore not imply σ. We consider the problem now from a logical point of view. Therefore, the extended basis attributes in $\mathcal{E}(N)$ are mapped to propositional variables as follows:

- Dance(Time) is V_1,
- Dance(Partaker{Name}) is V_2,
- Dance(Partaker{λ}) is V_3,
- Dance(Duo{Pair(Girl,Boy)}) is V_4,
- Dance(Duo{Pair(Girl)}) is V_5,
- Dance(Duo{Pair(Boy)}) is V_6,
- Dance(Duo{λ}) is V_7, and
- Dance(Rating) is V_8.

The set Π_N of Horn clauses that encodes the structure of the database schema N is then given by

$$V_2 \Rightarrow V_3, V_4 \Rightarrow V_5, V_4 \Rightarrow V_6, V_5 \Rightarrow V_7, V_6 \Rightarrow V_7.$$

The set Σ of FDs has the following corresponding set Π of Horn clauses

- $V_1 \Rightarrow V_2$, $V_1 \Rightarrow V_4$, $V_1 \Rightarrow V_8$,
- $V_2 \Rightarrow V_5$, $V_2 \Rightarrow V_6$,
- $V_5 \wedge V_6 \Rightarrow V_2$, and
- $V_4 \Rightarrow V_8$

and σ corresponds to the single Horn clause $\pi = V_2 \Rightarrow V_8$. The truth assignment θ with $\theta(V_i) = \mathit{true}$ iff $i \in \{2,3,5,6,7\}$ satisfies all clauses in $\Pi \cup \Pi_N$, but violates π. Therefore, π is not a logical consequence of $\Pi \cup \Pi_N$. Most importantly, note the correspondence between the nested counterexample relation r and the truth assignment θ. In fact, the tuples t_1 and t_2 agree exactly on their projections to those extended basis attributes whose corresponding propositional variable was assigned the truth value *true* by θ. This turns out to be the decisive argument for showing the equivalence.

3.2 The Result

Let $\varphi : \mathcal{E}(N) \rightarrow \mathcal{V}$ denote a bijection between the extended basis attributes of the underlying database schema N and the set $\mathcal{V}$ of propositional variables. Consider the FD $\sigma = \mathcal{X} \rightarrow \mathcal{Y}$ on N where $\mathcal{X} = \{X_1, \ldots, X_n\}$. Let $\Phi(\sigma)$ be the smallest set that contains the Horn clauses $\varphi(X_1) \wedge \cdots \wedge \varphi(X_n) \Rightarrow \varphi(Y)$ for all $Y \in \mathcal{Y}$. If Σ is a set of FDs defined on N, let $\Pi = \{\Phi(\sigma) \mid \sigma \in \Sigma\}$ denote the corresponding set of Horn clauses over $\mathcal{V}$. Furthermore, the set

$$\Pi_N = \{\varphi(U) \Rightarrow \varphi(V) \mid U, V \in \mathcal{E}(N), U \text{ covers}^2 V\}$$

denotes those Horn clauses which encode the structure of N. For example, the FD

{Dance(Duo{Pair(Girl)}),Dance(Duo{Pair(Boy)})} →
Dance(Partaker{Name})

results in the Horn clause $V_5 \wedge V_6 \Rightarrow V_2$. The main result of this paper is the following extension of Fagin's Equivalence Theorem from (Fagin 1977).

Theorem 3.1. *[Equivalence Theorem] Let N be a nested attribute, Σ a set of FDs and σ a single FD on N. Let Π_N denote the Horn clauses which encode the structure of N, and Π denote the corresponding set of Horn clauses for Σ. Then*

1) Σ implies σ,

2) Σ implies σ in the world of two-tuple instances, and

3) $\Pi \cup \Pi_N$ logically implies π for all $\pi \in \Phi(\sigma)$

are equivalent.■

3.3 An Outline of the Proof

We will use this subsection to outline the proof of Theorem 3.1. The equivalence between 1) and 2) is not difficult to see.

We will show the equivalence between 2) and 3). First we show that 3) implies 2). Suppose that 2) does not hold, i.e., Σ does not imply $\sigma = \mathcal{X} \rightarrow \mathcal{Y}$ in the world of two-tuple instances over N. That is, there are some $t_1, t_2 \in \mathit{dom}(N)$ with $\models_{\{t_1,t_2\}} \tau$ for all $\tau \in \Sigma$, but $\not\models_{\{t_1,t_2\}} \sigma$. Lemma 3.1 shows then that $\models_{\theta_{\{t_1,t_2\}}} \Pi \cup \Pi_N$, but $\not\models_{\theta_{\{t_1,t_2\}}} \pi$ for some $\pi \in \Phi(\sigma)$, i.e., 3) does not hold neither. In particular, $\models_{\theta_{\{t_1,t_2\}}} \Pi_N$ for the following reason. Let $V \Rightarrow W \in \Pi_N$, i.e., $V = \varphi(X)$ and $W = \varphi(Y)$ with $X, Y \in \mathcal{E}(N)$ and $Y \leq X$. If $\theta_{\{t_1,t_2\}}(V) = \mathit{true}$, then $\pi^N_X(t_1) = \pi^N_X(t_2)$ and thus

[2] U covers V iff $U < V$ and for all $W \in \mathcal{E}(N)$ with $U \leq W \leq V$ we have $U = W$ or $V = W$, this is just the standard definition of a *cover relation* for posets, see (Anderson 1987)

$\pi_Y^N(t_1) = \pi_Y^N(t_2)$ since $Y \leq X$. This, however, means that $\theta_{\{t_1,t_2\}}(W) = \mathit{true}$ holds as well, and therefore $\models_{\theta_{\{t_1,t_2\}}} V \Rightarrow W$. To complete the proof for this direction it remains to show the following lemma.

Lemma 3.1. *Let σ be an FD on the nested attribute N, and $r = \{t_1, t_2\} \subseteq \mathit{dom}(N)$. Then $\models_r \sigma$ if and only if $\models_{\theta_r} \pi$ for all $\pi \in \Phi(\sigma)$, where*

$$\theta_r(V) = \begin{cases} \mathit{true}, & \mathit{if}\ \pi^N_{\varphi^{-1}(V)}(t_1) = \pi^N_{\varphi^{-1}(V)}(t_2) \\ \mathit{false}, & \mathit{else} \end{cases}$$

for all $V \in \varphi(\mathcal{E}(N))$. ■

In order to complete the proof of Theorem 3.1 it remains to show that 2) implies 3). Suppose that 3) does not hold. We will show that 2) does not hold neither. Since 3) does not hold, there is some truth assignment θ which makes every formula in $\Pi \cup \Pi_N$ *true*, but makes some $\pi \in \Phi(\sigma)$ *false*. It is now sufficient to find some $r = \{t_1, t_2\} \subseteq \mathit{dom}(N)$ such that $\theta = \theta_r$. In this case, Lemma 3.1 shows that $\models_r \tau$ for all $\tau \in \Sigma$ and $\not\models_r \sigma$, i.e., 2) does not hold.

Let $\mathcal{U}(N) = \{U_1, \ldots, U_k\}$, and $\mathcal{X}^+_{U_i} = \{X \in \mathcal{E}(N) \mid X \leq U_i \text{ and } \theta(\varphi(X)) = \mathit{true}\} \cup \{\lambda_N\}$. Note that $\mathcal{X}_{U_i}$ is closed downwards with respect to $\leq$ since θ satisfies Π_N. Moreover, let $\mathcal{X}^+ = \{X_1 \sqcup \cdots \sqcup X_k \mid X_i \in \mathcal{X}^+_{U_i} \text{ for } 1 \leq i \leq k\}$. In this case, $\mathcal{X}^+$ is non-empty, closed downwards with respect to $\leq$, and closed under the join of reconcilable elements. $\mathcal{X}^+$ is non-empty since $\lambda_N \in \mathcal{X}^+_{U_i}$ for $i = 1, \ldots, k$. It is closed downwards with respect to $\leq$ since every $\mathcal{X}^+_{U_i}$ has the same property. In order to see that $\mathcal{X}^+$ is closed under the join of reconcilable elements suppose that $X = X_1 \sqcup \cdots \sqcup X_k, X' = X'_1 \sqcup \cdots \sqcup X'_k \in \mathcal{X}^+$ are reconcilable. Since $X \sqcap U_i = X_i$ and $X' \sqcap U_i = X'_i$ for $i = 1, \ldots, k$ it must be the case that X_i and X'_i are comparable with respect to $\leq$ for all $i = 1, \ldots, k$. This, however, means that $X \sqcup X' \in \mathcal{X}^+$ by definition of $\mathcal{X}^+$. The following lemma then shows the existence of two tuples $t_1, t_2 \in \mathit{dom}(N)$ with the property that for all $X \in \mathit{Sub}(N)$ we have $\pi^N_X(t_1) = \pi^N_X(t_2)$ if and only if $X \in \mathcal{X}^+$ holds. Since for all $X \in \mathcal{E}(N)$ we have that $\theta(\varphi(X)) = \mathit{true}$ if and only if $X \in \mathcal{X}^+$ if and only if $\pi^N_X(t_1) = \pi^N_X(t_2)$ holds, it follows that $\theta = \theta_{\{t_1,t_2\}}$. This concludes the proof of Theorem 3.1.

Lemma 3.2. *Let $N \in \mathcal{NA}$, and $\emptyset \neq \mathcal{X} \subseteq \mathit{Sub}(N)$ an ideal with respect to $\leq$ with the property that for reconcilable $X, Y \in \mathcal{X}$ also $X \sqcup Y \in \mathcal{X}$ holds. Then there are $t_N, t'_N \in \mathit{dom}(N)$ such that for all $W \in \mathit{Sub}(N)$ we have $\pi^N_W(t_N) = \pi^N_W(t'_N)$ if and only if $W \in \mathcal{X}$.* ■

The detailed (and challenging) proof of Lemma 3.2 was previously published (Hartmann et al. 2006).

4 First-Literal Unit Resolution

The Equivalence Theorem 3.1 states that the FD σ is a consequence of the set Σ of FDs on the underlying database schema N if and only if each of the corresponding Horn clauses in $\Phi(\sigma)$ is a logical consequence of $\Pi \cup \Pi_N$. The last problem, however, can be easily converted into the well-studied problem of satisfiability of propositional Horn clauses (Horn 1951, Henschen & Wos 1974, Dowling & Gallier 1984). A fast algorithm for the Horn clause satisfiability problem is the "first-literal unit resolution procedure" due to Chang (Chang 1976, Chang & Lee 1987). Using the Equivalence Theorem, we exploit Chang's algorithm to obtain an efficient algorithm that solves the implication problem for FDs in complex-value databases. In the first step of the algorithm, we form a set $\mathcal{S}$ of strings of symbols. Each string consists of extended basis attributes, negation signs ($\sim$) and commas (,). For each FD

$$\{X_1, \ldots, X_n\} \to \{Y_1, \ldots, Y_m\}$$

in Σ we include in $\mathcal{S}$ the m strings

$$\begin{array}{c} \sim X_1, \cdots, \sim X_n, Y_1 \\ \vdots \\ \sim X_1, \cdots, \sim X_n, Y_m \end{array}$$

The string $\sim X_1, \cdots, \sim X_n, Y_i$ corresponds to the Horn clause

$$\neg\varphi(X_1) \vee \cdots \vee \neg\varphi(X_n) \vee \varphi(Y_i).$$

Furthermore, consider each pair $U, V \in \mathcal{E}(N)$ with $V < U$ such that for all $W \in \mathcal{E}(N)$ with $V \leq W \leq U$ we have $V = W$ or $U = W$. For each of these pairs U, V we include in $\mathcal{S}$ the string $\sim U, V$. Let σ be the FD $\mathcal{X} \to \mathcal{Y}$ where $\mathcal{X} = \{X_1, \ldots, X_n\}$. For convenience, we assume that $\mathcal{Y}$ is a singleton. Otherwise, the entire algorithm is repeated for every element of $\mathcal{Y}$. Anyway, for $\mathcal{Y} = \{Y\}$ we include in $\mathcal{S}$ the $(n + 1)$ strings

$$\begin{array}{c} X_1 \\ \vdots \\ X_n \\ \sim Y \end{array}$$

which correspond to the negation of the Horn clause

$$\neg\varphi(X_1) \vee \cdots \vee \neg\varphi(X_n) \vee \varphi(Y).$$

As a simple example consider the nested attribute

$N =$
Dance(Time,Partaker{Name},Duo{Pair(Girl,Boy)},Rating)

and let Σ and σ be as in Subsection 3.1. In this case, $\mathcal{S}$ contains the 14 strings:

$\sim$ Dance(Time), Dance(Partaker{Name})

$\sim$ Dance(Time), Dance(Duo{Pair(Girl,Boy)})

$\sim$ Dance(Time), Dance(Rating)

$\sim$ Dance(Partaker{Name}), Dance(Duo{Pair(Girl)})

$\sim$ Dance(Partaker{Name}), Dance(Duo{Pair(Boy)})

$\sim$ Dance(Duo{Pair(Girl)}), $\sim$ Dance(Duo{Pair(Boy)}), Dance(Partaker{Name})

$\sim$ Dance(Duo{Pair(Girl,Boy)}), Dance(Rating)

$\sim$ Dance(Partaker{Name}), Dance(Partaker{λ})

$\sim$ Dance(Duo{Pair(Girl,Boy)}), Dance(Duo{Pair(Girl)})

$\sim$ Dance(Duo{Pair(Girl,Boy)}), Dance(Duo{Pair(Boy)})

$\sim$ Dance(Duo{Pair(Girl)}), Dance(Duo{λ})

$\sim$ Dance(Duo{Pair(Boy)}), Dance(Duo{λ})

Dance(Partaker{Name})

$\sim$ Dance(Rating)

We call each attribute in $\mathcal{E}(N)$ an "atom" and the concatenation of a negation sign with such an atom a "negative atom". The algorithm proceeds by searching for an atom X such that (a) X is a string in $\mathcal{S}$, and (b) There is a string in $\mathcal{S}$ that begins with $\sim X$. In our example the atom *Dance(Partaker{Name})* satisfies both (a) and (b). If there are several atoms X that satisfy (a) and (b), the algorithm would now arbitrarily select one of them. In the next step of the algorithm, each string that begins with $\sim X$ is shortened by erasing the leading negation sign, the X, and the comma that follows X (if there is such a comma).

$\sim$ Dance(Time), Dance(Partaker{Name})
$\sim$ Dance(Time), Dance(Duo{Pair(Girl,Boy)})
$\sim$ Dance(Time), Dance(Rating)
Dance(Duo{Pair(Girl)})
Dance(Duo{Pair(Boy)})
$\sim$ Dance(Duo{Pair(Girl)}), $\sim$ Dance(Duo{Pair(Boy)}),
Dance(Partaker{Name})
$\sim$ Dance(Duo{Pair(Girl,Boy)}), Dance(Rating)
Dance(Partaker{λ})
$\sim$ Dance(Duo{Pair(Girl,Boy)}), Dance(Duo{Pair(Girl)})
$\sim$ Dance(Duo{Pair(Girl,Boy)}), Dance(Duo{Pair(Boy)})
$\sim$ Dance(Duo{Pair(Girl)}), Dance(Duo{λ})
$\sim$ Dance(Duo{Pair(Boy)}), Dance(Duo{λ})
Dance(Partaker{Name})
$\sim$ Dance(Rating)

Then we repeat the procedure by again searching for an atom X that satisfies both (a) and (b). After selecting *Dance(Duo{Pair(Girl)})* we obtain

$\sim$ Dance(Time), Dance(Partaker{Name})
$\sim$ Dance(Time), Dance(Duo{Pair(Girl,Boy)})
$\sim$ Dance(Time), Dance(Rating)
Dance(Duo{Pair(Girl)})
Dance(Duo{Pair(Boy)})
$\sim$ Dance(Duo{Pair(Boy)}), Dance(Partaker{Name})
$\sim$ Dance(Duo{Pair(Girl,Boy)}), Dance(Rating)
Dance(Partaker{λ})
$\sim$ Dance(Duo{Pair(Girl,Boy)}), Dance(Duo{Pair(Girl)})
$\sim$ Dance(Duo{Pair(Girl,Boy)}), Dance(Duo{Pair(Boy)})
Dance(Duo{λ})
$\sim$ Dance(Duo{Pair(Boy)}), Dance(Duo{λ})
Dance(Partaker{Name})
$\sim$ Dance(Rating)

and by selecting *Dance(Duo{Pair(Boy)})* we get

$\sim$ Dance(Time), Dance(Partaker{Name})
$\sim$ Dance(Time), Dance(Duo{Pair(Girl,Boy)})
$\sim$ Dance(Time), Dance(Rating)
Dance(Duo{Pair(Girl)})
Dance(Duo{Pair(Boy)})
Dance(Partaker{Name})
$\sim$ Dance(Duo{Pair(Girl,Boy)}), Dance(Rating)
Dance(Partaker{λ})
$\sim$ Dance(Duo{Pair(Girl,Boy)}), Dance(Duo{Pair(Girl)})
$\sim$ Dance(Duo{Pair(Girl,Boy)}), Dance(Duo{Pair(Boy)})
Dance(Duo{λ})
Dance(Duo{λ})
Dance(Partaker{Name})
$\sim$ Dance(Rating)

The entire algorithm halts either when

1) the empty string ϵ is generated, or when
2) there is no atom X that satisifies both (a) and (b).

If 1) occurs first, that is, the empty string ϵ is generated, then σ is a consequence of Σ. If 2) occurs first, then σ is not a consequence of Σ. The algorithm always terminates, and gives a correct answer by Chang's theorem on Horn clauses and our Equivalence Theorem.
In our example the algorithm terminates since no further atom satisfying both (a) and (b) can be found. Therefore, σ is not a consequence of Σ.

5 Reusing Relational Design Tools

A direct consequence of Fagin's Equivalence Theorem and Equivalence Theorem 3.1 is that relational database design tools can be applied to solve design problems in complex-value databases. Extended basis attributes of the nested attribute N are interpreted as flat attributes forming the corresponding relation schema, i.e., $R_N = \mathcal{E}(N)$. Each FD $\sigma = X \to Y$ in Σ becomes the relational FD $\sigma' = \vartheta(X) \to \vartheta(Y)$. Given Σ on the nested attribute N, the corresponding set of relational FDs is

$$\begin{aligned}\Sigma' \;=\; & \{\sigma' \mid \sigma \in \Sigma\} \cup \\ & \{U \to V \mid U, V \in \mathcal{E}(N), U \text{ covers } V\}\end{aligned}$$

Corollary 5.1. *Let Σ be a set of FDs and σ be a single FD, all defined on the nested attribute N. Then σ is implied by Σ if and only if σ' is implied by Σ' on R_N.* □

As a simple example consider again the nested attribute

$$N = \text{Dance(Time,Partaker\{Name\},Duo\{Pair(Girl,Boy)\},Rating)}.$$

The extended basis attributes are mapped to flat attribute names as follows:

- Dance(Time) is A,
- Dance(Partaker{Name}) is B,
- Dance(Partaker{λ}) is C,
- Dance(Duo{Pair(Girl,Boy)}) is D,
- Dance(Duo{Pair(Girl)}) is E,
- Dance(Duo{Pair(Boy)}) is F,
- Dance(Duo{λ}) is G, and
- Dance(Rating) is H.

The corresponding relation schema is therefore $R_N = \{A, B, C, D, E, F, G, H\}$. The FDs in Σ from Subsection 3.1 are encoded as

$$A \to BDH, B \to EF, EF \to B, D \to G$$

and the structure of N is encoded by the following FDs:

$$B \to C, D \to E, D \to F, E \to G, F \to G \quad .$$

Suppose we want to decide if σ from Subsection 3.1 is a consequence of Σ. Then this is equivalent to deciding whether $B \to H$ is a logical consequences of Σ'. In order to decide the last problem, we may apply well-known techniques for relational databases (Beeri & Bernstein 1979) to compute the closure $B^+ = CEFG$ of B with respect to Σ'. Since $H \notin B^+$ the answer to the last problem is no. The Equivalence theorem tells us therefore that σ is not a consequence of Σ. It is future work to exploit the reuse of relational tools further.

6 Conclusion and Future Work

We have extended Fagin's well-known equivalence between implications of functional dependencies in relational databases and implications of propositional Horn clauses to functional dependencies in databases that support arbitrary finite nesting using data constructors for records, lists, sets and multisets. Our framework is not based on any specific data model, and the result may therefore lead to a better understanding of complex values in general. The work in (Hartmann et al. 2006) shows that the expressiveness of our functional dependencies is complementary to those investigated in many advanced data models. The extension of the equivalence result follows from a deep case-by-case analysis of the data constructors and the algebraic properties of the database schemata. The results of this article provide two new ways to look at the implication of functional dependencies in complex-value databases: as the logical implication of a corresponding set of Horn clauses, and as the logical implication of a corresponding set of functional dependencies in relational databases. It is therefore possible to utilise already existing tools from these areas for solving database design problems in the presence of several data constructors.

For the near future we would like to extend the equivalence to multivalued dependencies in the presence of records and lists. It will be challenging to investigate multivalued dependencies in the presence of sets, or to extend results to include (disjoint) unions. There is an alternative proof for the original equivalence result for relational FDs (Fagin 1977). That proof is syntactical in the sense that it takes advantage of the fact that the (finite) implication of FDs can be characterised by a finite, sound and complete set of syntactic inference rules (Armstrong 1974). Such sets of inference rules have also been proposed for the setting of the present paper (Hartmann et al. 2006). A generalisation of these syntactical proofs seems also desirable.

References

Abiteboul, S., Hull, R. & Vianu, V. (1995), *Foundations of Databases*, Addison-Wesley.

Anderson, I. (1987), *Combinatorics of finite sets*, Oxford Science Publications, The Clarendon Press Oxford University Press, New York.

Armstrong, W. W. (1974), 'Dependency structures of database relationships', *Information Processing* pp. 580–583.

Atkinson, M., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D. & Zdonik, S. (1989), The object-oriented database system manifesto, *in* 'Proceedings of the International Conference on Deductive and Object-Oriented Databases', pp. 40–57.

Beeri, C. & Bernstein, P. A. (1979), 'Computational problems related to the design of normal form relational schemata', *Transactions on Database Systems (TODS)* pp. 30–59.

Biskup, J. (1998), Achievements of relational database schema design theory revisited, *in* 'Semantics in databases', number 1358 *in* 'Lecture Notes in Computer Science', Springer, pp. 29–54.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E. & Yergeau, F. (2004), 'Extensible markup language (XML) 1.0 (third edition) W3C recommendation 04 february 2004', http://www.w3.org/TR/2004/REC-xml-20040204/.

Chang, C. (1976), Deduce: A deductive query language for relational data bases, *in* C. H. Chen, ed., 'Pattern Recognition and Artificial Intelligence', Academic Press, New York, pp. 108–134.

Chang, C. & Lee, R. (1987), *Symbolic Logic and Mechanical Theorem Proving*, Academic Press.

Chen, P. P. (1976), 'The entity-relationship model: Towards a unified view of data', *Transactions on Database Systems (TODS)* **1**, 9–36.

Codd, E. F. (1970), 'A relational model of data for large shared data banks', *Communications of the ACM* **13**(6), 377–387.

Delobel, C. & Adiba, M. (1985), *Relational database systems*, North Holland.

Delobel, C. & Parker, D. (1978), Functional and multivalued dependencies in a relational database and the theory of Boolean switching functions, Technical report, University of Grenoble, Grenoble, France.

Dowling, W. & Gallier, J. (1984), 'Linear-time algorithms for testing the satisfiability of propositional horn formulae', *Journal of Logic Programming* **1**(3), 267–284.

Fagin, R. (1977), 'Functional dependencies in a relational data base and propositional logic', *IBM Journal of Research and Development* **21**(6), 543–544.

Fagin, R. (1982), 'Horn clauses and data dependencies', *Journal of the ACM* **29**(4), 952–985.

Hartmann, S., Link, S. & Schewe, K.-D. (2006), 'Axiomatisations of functional dependencies in the presence of records, lists, sets and multisets', accepted for Theoretical Computer Science (TCS).

Henschen, L. & Wos, L. (1974), 'Unit refutations and horn sets', *Journal of the ACM* **21**(4), 590–605.

Horn, A. (1951), 'On sentences which are true of direct unions of algebras', *Journal of symbolic logic* **16**, 14–21.

Hull, R. & King, R. (1987), 'Semantic database modeling: Survey, applications and research issues', *ACM Computing Surveys* **19**(3).

Li, J., Ng, S. & Wong, L. (2002), Bioinformatics adventures in database research, *in* 'Proceedings of the International Conference on Database Theory (ICDT)', number 2572 *in* 'Lecture Notes in Computer Science', Springer, pp. 31–46.

McKinsey, J. C. C. & Tarski, A. (1946), 'On closed elements in closure algebras', *Annals of Mathematics* **47**, 122–146.

Parker, D. S. & Delobel, C. (1979), Algorithmic applications for a new result on multivalued dependencies, *in* 'Proceedings of the International Conference on Very Large Data Bases (VLDB)', pp. 67–74.

Sagiv, Y. (1980), 'An algorithm for inferring multivalued dependencies with an application to propositional logic', *Journal of the ACM* **27**(2), 250–262.

Sagiv, Y., Delobel, C., Parker Jr., D. S. & Fagin, R. (1981), 'An equivalence between relational database dependencies and a fragment of propositional logic', *Journal of the ACM* **28**(3), 435–453.

Suciu, D. (2001), 'On database theory and XML', *SIGMOD Record* **30**(3), 39–45.

Thalheim, B. (2000), *Entity-Relationship Modeling: Foundations of Database Technology*, Springer-Verlag.

Vianu, V. (2001), A web odyssey: from Codd to XML, *in* 'Principles of Database Systems', ACM, pp. 1–15.

Vincent, M. (1999), 'Semantic foundation of 4NF in relational database design', *Acta Informatica* **36**, 1–41.

A Further Study in the Data Partitioning Approach for Frequent Itemsets Mining

Son N. Nguyen, Maria E. Orlowska

School of Information Technology and Electrical Engineering
The University of Queensland, QLD 4072, Australia

{nnson, maria)@itee.uq.edu.au

Abstract

Frequent itemsets mining is well explored for various data types, and its computational complexity is well understood. Based on our previous work by Nguyen and Orlowska (2005), this paper shows the extension of the data pre-processing approach to further improve the performance of frequent itemsets computation. The methods focus on potential reduction of the size of the input data required for deployment of the partitioning based algorithms.

We have made a series of the data pre-processing methods such that the final step of the Partition algorithm, where a combination of all local candidate sets must be processed, is executed on substantially smaller input data. Moreover, we have made a comparison among these methods based on the experiments with particular data sets.

Keywords: Data mining, Frequent itemset, Partition, Algorithm, Performance.

1 Introduction

Mining frequent itemsets is important and interesting to the fundamental research in the mining of association rules which is introduced firstly by Agrawal, Imielinski, Swami (1993). Many algorithms and their subsequent improvements have been proposed to solve association rules mining, especially frequent itemsets mining problems.

There are many well-accepted approaches such as “Apriori” by Agrawal, Srikant (1994), ECLAT by Zaki (2000), and more recently “FP-growth” by Han et al. (2004). Another interesting class of solutions is based on the data partitioning approach. This fundamental concept was originally proposed as a Partition algorithm by Savasere, Omiecinski, Navathe (1995), and it was improved later in AS-CPA by Lin, Dunham (1998) and ARMOR by Pudi, Haritsa (2003). A common feature of these results is their target, namely the limitation of I/O operations by considering data subsets dictated by the main memory size.

In our previous work by Nguyen and Orlowska (2005), the incremental clustering method for data pre-processing was proposed to improve the overall performance of mining large data sets by a smarter but not too ‘expensive’ design of the data fragments - rather than determine them by a sequential transaction allocation based on the fragment size only.

The main goal of this paper is to extend the above work by using the different data pre-processing methods for the partitioning approach in order to improve the performance. We design new fragmentation algorithms based on new similarity functions and new cluster constructions for data clustering. These algorithms reduce the computation cost to get more efficient. Our study is supported by a series of experiments which indicate a dramatic improvement in the performance of the partitioning approach with our fragmentation methods.

The remainder of the paper is organised as follows. Section 2 introduces the basic concepts related to frequent itemsets mining. Section 3 reviews the partitioning approach for frequent itemsets mining. We propose the different pre-processing data fragmentation methods in section 4. Section 5 shows the result and a comparison from our experiments. Finally, we present our concluding remarks.

2 Preliminary concepts

For the completeness of this presentation and to establish our notation, this section gives a formal description of the problem of mining frequent itemsets. It can be stated as follows:

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of m distinct literals called items. Transaction database D is a set of variable length transactions over I.

Each transaction contains a set of items $\{i_j, i_k, \ldots, i_h\} \subseteq I$ in an ordered list. Each transaction has an associated unique identifier called TID.

For an itemset $X \subseteq I$, the support is denoted $sup_D(X)$, equals to the fraction of transactions in D containing X.

The problem of mining frequent itemsets is to generate all frequent itemsets X that have $sup_D(X)$ no less than user specified minimum support threshold.

3 Review the Partitioning approach for frequent itemsets mining

3.1 The Partitioning approach

Savasere et al. (1995) proposed the Partition algorithm based on the following principle. A fragment $P \subseteq D$ of

Copyright © 2006, Australian Computer Society, Inc. This paper appeared at the 17th *Australasian Database Conference (ADC 2006)*, Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

the database is defined as any subset of the transactions contained in the database D. Further, any two different fragments are non-overlapping. *Local support* for an itemset is the fraction of transactions containing that itemset in a fragment. *Local candidate itemset* is being tested for minimum support within a given fragment. A *Local frequent itemset* is an itemset whose local support in the fragment is no less than the minimum support. A *Local frequent itemset* may or may not be frequent in the context of the entire database. *Global support, Global candidate itemset, Global frequent itemset* are defined as above except they are in the context of the entire database. The goal is to find all *Global frequent itemsets*.

Firstly, the Partition algorithm divides D into n fragments, and processes one fragment in the main memory at a time. The algorithm first scans fragment P_i, for i = 1,…,n, to find the set of all *Local frequent itemsets* in P_i, denoted as LP_i. Then, by taking the union of LP_i, a set of candidate itemsets over D is constructed, denoted as C^G *(Global candidates)*. There is an important property: ***if itemset, X, is a frequent itemset in database D, then X must be a frequent itemset in at least one of the n fragments P_1, P_2,…, P_n***. As a result, C^G is a superset of the set of all *Global frequent itemsets* in D. Secondly, the algorithm scans each fragment for the second time to calculate the support of each itemset in C^G and to find which candidate itemsets are really frequent itemsets in D (Figure 3.1).

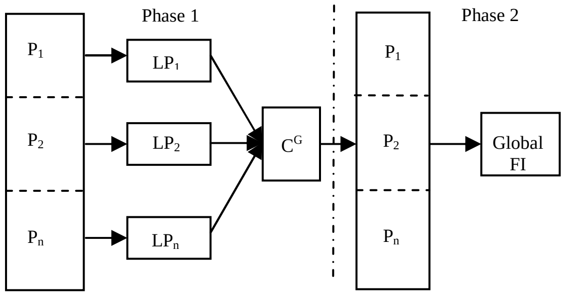


Figure 3.1: The Partitioning approach

The main goal of this division of process into *Local* and later *Global* computation is to process one fragment in the main memory at a time - to avoid multiple scans over D from secondary storage.

3.2 Related work

One of the Partition algorithm derivatives is AS-CPA (*Anti-Skew Counting Partition Algorithm*) by Lin, Dunham (1998). It makes use of the cumulative count of each candidate itemset to achieve a smaller number of Global candidates. The main difference is that AS-CPA provides several effective techniques (*Early Local Pruning, Global Anti-Skew*) to filter out false candidate itemsets at an earlier stage.

Recently, there has been another development based on the partitioning approach in the ARMOR algorithm by Pudi, Haritsa (2003). It uses the array and graph structure to store the cumulative C^G while processing the fragments.

All the above algorithms mainly attempt to reduce the number of false candidates as early as possible. However, they do not consider any features and characteristics of data sets in order to partition the original data set more suitably for further processing.

In our previous work (Nguyen, Orlowska (2005)), we demonstrated that looking more closely into the data itself may deliver good gains in overall performance. We show how to reach a better data partition based on the relative similarity between the transactions forming the data set. As a result, the number of *Local frequent itemsets* can be dramatically reduced. Furthermore, in many cases that leads to a larger number of common *Global candidates* among fragments. Finally, as a consequence, these methods reduce substantially the C^G set which must be checked in the Partition algorithm.

4 Data set pre-processing methods

The data set pre-processing is used to partition the input data set in order to get the special fragments which can generate the smaller number of *Local frequent itemsets* as well as the smaller number of *Global candidate itemsets.* The goal is to generate the fragments which have more dissimilar transactions. We start with simple illustrations.

4.1 Intuitive example

A giving data set D has only 12 transactions as in table 4.1. Like many other authors in this area, we assume that the set of transactions is ordered by items ids.

TID	List of Items	TID	List of Items
T1	1,4,12,32	T7	3,6,13,22
T2	1,4,13,30	T8	3,6,14,24
T3	1,4,13,34	T9	3,6,14,22
T4	1,4,13,34	T10	3,6,14,19
T5	1,4,13,36	T11	3,6,17,29
T6	1,4,12,30	T12	3,6,18,23

Table 4.1: Example data set

We assume further, the minimum support threshold at percentage: 50%. After a simple calculation we get; *Global frequent itemsets* = {{1}, {4}, {3}, {6}, {1,4}, {3,6}} and there are 6 itemsets. The following subsections show clearly how much benefit can be gained by different fragmentations.

4.1.1 Data set partitioned into 2 fragments

We consider two different partitions on the same data set to illustrate the dependency of the fragmentation's composition on the computational cost of the final phase of the Partitioning algorithm.

Let D be partitioned into 2 sequent fragments: P_1 = {T1, T2, T3, T4, T5, T6} and P_2 = {T7,T8,T9,T10, T11, T12}.

As a result:

LP_1 = {{1},{4},{13},{1,4},{1,13},{4,13},{1,4,13}}; $|LP_1| = 7$

LP_2 = {{3},{6},{14},{3,6},{3,14},{6,14},{3,6,14}}; $|LP_2| = 7$

$C^G = LP_1 \cup LP_2$. Therefore, the number of the *Global Candidates*, $|C^G| = 14$. Note that there is no common candidate among LP_i.

However, if D is partitioned differently into 2 'skipping' fragments such that each fragment holds more dissimilar transactions: P_1 = {T1, T2, T3, T7, T8, T9} and P_2 = {T4, T5, T6, T10, T11, T12}.

As a result:

LP_1= {{1}, {4}, {13}, {3}, {6}, {1,4}, {3, 6}}; $|LP_1| = 7$
LP_2= {{1}, {4}, {3}, {6}, {1,4}, {3,6}}; $|LP_2| = 6$

$C^G = LP_1 \cup LP_2$. Therefore, the number of the *Global Candidates*, $|C^G| = 7$. Note that there are 6 common candidates among LP_i, thus only 1 candidate has to be checked in second phase.

4.1.2 Data set partitioned into 3 fragments

This example illustrates the relationship between densities of the fragmentation when applied with the expected computation cost.

If D is partitioned into 3 sequent fragments: P_1 = {T1, T2, T3, T4}; P_2 = {T5,T6,T7, T8}; P_3 = {T9, T10, T11, T12}.

As a result, the number of the *Global candidates* is $|C^G| = 22$ and there is no common candidate.

However, if D is partitioned into 3 'skipping' fragments that have more dissimilar transactions:

P_1 = {T1, T4, T7, T10}; P_2 = {T2, T5, T8, T11}; P_3 = {T3, T6, T9, T12}.

As a result, the number of the *Global candidates* is $|C^G| = 10$ and there are 6 common candidates, thus only 4 candidates have to be checked in the second phase of the Partition algorithm.

This simple example confirms the fact that there are some relationships between the composition of fragments and the amount of computation required at the end. Our goal is to reduce the cost of this computation.

4.2 The incremental clustering algorithm

The original incremental clustering algorithm was proposed by Nguyen, Orlowska (2005). The requirements for a clustering algorithm restrict the number of scans of the data set. The data set will be scanned only once and all clusters (fragments) containing mostly dissimilar transactions are generated at the end of that scan. The following are some basic definitions.

Definition 4.1: Cluster centroid is a set of all items in the cluster, we denote it $C_i = \{I_1, I_2,.., I_n\}$. Additionally, each item in C_i has its associated weight which is its number of occurrences in the cluster; $\{w_1, w_2, \ldots, w_n\}$

Definition 4.2: Similarity function between two item sets, in particular a transaction and a cluster centroid, is denoted Sim (T_i, C_j) and defined as follows;

Sim $(T_i, C_j) \rightarrow R^+$; Calculation of this function:

1. Let S be the intersection between the arguments of Sim function, $S = T_i \cap C_j$

2. If $S = \emptyset$ then Sim $(T_i, C_j) = 0$. Otherwise, $S = \{I_1, I_2, \ldots, I_m\}$ with the corresponding weights $\{w_1, w_2, \ldots, w_m\}$ in cluster C_j, respectively, therefore Sim $(T_i, C_j) = w_1 + w_2 + \ldots + w_m$

Cluster Construction:

Informally, each transaction is evaluated in terms of the following criteria;

a) We assign a new transaction T_i to cluster C_j which has the minimum Sim(T_i, C_j) value among open clusters (*a cluster is open if has not exceeded its expected size in terms of number of transactions*).
b) Each new allocation to a cluster C_j, updates the cluster centroid C_j. In fact the update can be summarized by the statement; all already existing common items' weight is increased by 1, and the other new items are added to C_j with the weight of value 1.

Reasoning about the size of clusters: the cluster sizes will be well balanced in order to control the number of *Local frequent itemsets (*LP_i*)*. If the size of the cluster is too small, then the number of LP_i can be very big mainly due to the fact that the minimum occurrence number of *Local frequent itemsets* is small.

The pseudo incremental clustering algorithm is described as follows.

Algorithm 1 (clustering by incremental processing):

```
Input:  Transaction database: D; k – number of output
        clusters
Output: k clusters based on the above criteria for
        Partition approach.
Begin
  1. Assign the first k transactions to all k clusters, and
     initialize the all cluster centroids: {$C_1, C_2, \ldots, C_k$}
  2. Consider the next k transactions. These k
     transactions are assigned to k different clusters.
     These operations are done based on the following
     criteria: (i) the minimum similarity between the
     new transaction and the suitable clusters; (ii) the
     sizes of these clusters are controlled to keep the
     balance. The following are more detail about this
     processing.
     Let $C^{run} = \{C_1, C_2, \ldots, C_k\}$ is a set of all k clusters;
     $T^{run} = \{T_1, T_2, \ldots, T_k\}$ is a set of all k transactions
     For each transaction $T_i$ in $T^{run}$ : $T_1$ to $T_k$
     Begin
       a) Calculate the similarity functions between $T_i$ and
          all the clusters in $C^{run}$ ; determine the minimum
          similarity function value, denoted Sim($T_i, C_j$)
       b) Assign $T_i$ to cluster $C_j$ which has the minimum
          Sim($T_i, C_j$) value. Update the cluster centroid $C_j$
       c) Remove $C_j$ from the set of all the suitable
          clusters in order to keep the same size constraint.
          That means the next transaction is belonged to
          the existing clusters after removing $C_j$: $C^{run} =
          C^{run} - \{C_j\}$;
     End
  3. Repeat step 2 till all transactions in D are clustered
End
```

The time complexity of this incremental clustering algorithm is O(|D| * k *m) where |D| is the number of all transactions, k is the given number of clusters, and m is the number of all items in D.

4.3 Similarity function based on the minimum support threshold

Based on the proposed ***Algorithm 1***, we want to develop new algorithms for further improvements. The similarity function is one of the key points for clustering algorithms. Traditionally, the similarity between two objects is often determined by their bag intersection, the more elements two objects contain, the more similarity they are considered. There are many different measures in use, for example, let X and Y be two objects, |X| is the number of elements in X, the Jaccard's coefficient (Ganesan et al. 2003), $Sim_{Jacc}(X,Y)$, is defined to be:

$$Sim_{Jacc}(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

However, in the context of frequent itemsets mining we have to consider not only the common items, but also the number of occurrences of items. Therefore, the new similarity function between a transaction and a cluster centroid is defined by a different way as follows:

Given database D and the minimum support threshold min_sup (percentage); k is the number of output clusters.

With this method, we assume that the sizes of all output fragments are the same. As a result, the minimum number of occurrences for *Local frequent itemset* in every fragment can be calculated, denoted as sup_k

$$sup_k = \frac{|D| * \text{min\_sup}}{k}$$; where |D| is number of all transactions in D.

The new similarity function is denoted $New\_Sim(T_i, C_j) \rightarrow R^+$, which can be calculated as below:

1). Let $C_{jk} = \{ I_1, I_2, \ldots, I_m \} \subseteq C_j$; where all items in C_{jk} have the corresponding weights in cluster C_j $\{w_1, w_2, \ldots, w_m\}$ respectively which are no less than $(sup_k - 1)$; We only consider these items because they may become new frequent items if the new transaction is added to the cluster (their occurrences will increase by 1).

Let S_k be the intersection between a transaction T_i and C_{jk}; $S_k = T_i \cap C_{jk}$

2). If $S_k = \emptyset$ then New_Sim $(T_i, C_j) = 0$.

Otherwise, $S_k = \{I_{k1}, I_{k2}, \ldots, I_{kn}\}$ with the corresponding weights $\{w_{k1}, w_{k2}, \ldots, w_{kn}\}$ in cluster C_j. Therefore,

$$New\_Sim(T_i, C_j) = w_{k1} + w_{k2} + \ldots + w_{kn}$$

The ***Algorithm2*** uses the same cluster construction and method as the ***Algorithm1***, but the similarity function is replaced by New_Sim() function.

Algorithm 2 (similarity function based on the minimum support threshold):

Input: Transaction database: D; min_sup - minimum support threshold; k – number of output clusters
Output: k clusters for Partition approach.

```
Begin:
  1. Assign the first k transactions to all k clusters, and
     initialize the all cluster centroids: $\{C_1, C_2, \ldots, C_k\}$
  2. Consider the next k transactions. These k
     transactions are assigned to k different clusters.
     Let $C^{run} = \{C_1, C_2, \ldots, C_k\}$ is a set of all k clusters;
     $T^{run} = \{T_1, T_2, \ldots, T_k\}$ is a set of all k transactions
     For each transaction $T_i$ in $T^{run}$ : $T_1$ to $T_k$
     Begin
       a) Calculate the new similarity function between $T_i$
          and all the clusters in $C^{run}$ ; determine the
          minimum similarity function value, denoted
          New_Sim($T_i$, $C_j$)
       b) Assign $T_i$ to cluster $C_j$ which has the minimum
          New_Sim($T_i$, $C_j$) value. Update the cluster
          centroid $C_j$ (maintenance of $C_j$ as the Algorithm1
          for further computation).
       c) Remove $C_j$ from the set of all the suitable
          clusters in order to keep the same size constraint.
          $C^{run} = C^{run} - \{ C_j \}$;
     End
  3. Repeat step 2 till all transactions in D are clustered
End
```

Remarks: This ***Algorithm2*** can reduce dramatically the computation of similarity function, because cluster centroid C_j contains many items (after steps, it contains almost items in database D), but only some of them have the corresponding weights which are no less than $(sup_k - 1)$. Therefore, the cost of intersection step for similarity computation is reduced.

4.4 Cluster construction based on flexible sizes

The cluster sizes should be well balanced in order to have the same distribution of items among fragments. However, the other cluster construction can be used to get the flexible size of clusters, we want to compare the output result of flexible clusters with those of the other methods and study how much we can gain with this method. Based on the ***Algorithm1***, this method uses the same similarity function Sim(T_i, C_j). We extend to the new pre-processing algorithm as follows.

Algorithm 3 (based on the flexible sizes of clusters):

Input: Transaction database: D; k – number of clusters
Output: k clusters for Partition approach.

```
Begin
1. Assign the first k transactions to all k clusters, and
   initialize the all cluster centroids: $\{C_1, C_2, \ldots, C_k\}$
2. Consider the next transaction. This transaction is
   assigned to cluster which has the minimum similarity
   between the new transaction and the clusters.
   Let $C^{run} = \{C_1, C_2, \ldots, C_k\}$ is a set of all k clusters;
   the new transaction $T_i$
   Begin
   a) Calculate the similarity functions between $T_i$ and
      all the clusters in $C^{run}$ ; determine the minimum
      similarity function value, denoted Sim($T_i$, $C_j$)
   b) Assign $T_i$ to cluster $C_j$ which has the minimum
      Sim($T_i$, $C_j$) value. Update the cluster centroid $C_j$
   End
3. Repeat step 2 till all transactions in D are clustered
End
```

5 Experimental results

In this section, we conducted experiments on three data sets: one synthetic data set (T10I4D100K) generated by Agrawal, Srikant (1994), one small real data set and one big real data set (BMS-WebView-1 and BMS-POS) from Ron et al. (2000). These data sets are converted to format as the above definitions.

Data sets	Transactions	Items	DB Size (~MB)
T10I4D100K	100K	870	4
WebView-1	26K	492	0.7
BMS-POS	435K	1657	10

Table 5.1: The characteristics of data sets

Our goal is to compare the cardinality of the outputs; at the *Local level* and the *Global level*, before and after application of our pre-processing methods. The expectation is that the final computation cost of Partition algorithm reduces dramatically after pre-processing.

Firstly, data set is partitioned into fragments; secondly the Apriori algorithm implemented by Zhu T. (2004) is applied to find *Local frequent itemsets (LP$_i$)* for each fragment. Subsequently, union of these LPi generates the *Global candidates*.

Resulting figures for each data set are represented in following template table 5.2. The 2nd, 3rd, 4th and 5th columns' names indicate four methods for data preparation: ***Sequent*** fragments correspond to loading clusters with original data (no pre-processing), ***Clustering*** fragments are constructed as described in section 4.2 (***Algorithm1***); the ***New similarity function*** fragments are the other pre-processed data as presented by our new method described in section 4.3 (***Algorithm2***); the ***New cluster construction*** fragments are the other new method described in section 4.4 (***Algorithm3***).

The data sets used are indicated on the top of each table segment. We present three different scenarios; each data set is partitioned into 1, 2 and 5 fragments. Each column represents the numbers of the *Local level* (LP$_1$, LP$_2$, …, LP$_n$), the number of *Global candidates*. Note that this figure is presented by showing its two components; for example; it, ***16 + (378),*** indicates that there are **16** candidates to be checked and **378** common candidates don't need additional check.

Further, to discuss the impact of different fragmentations and threshold level, let us denote the cardinality of checked *Global candidates* set as $|C_n^G|$, where n is the number of fragments. As can be seen from table 5.2 and table 5.3, there are big gains from the careful data pre-processing methods.

Firstly, $|C_n^G|$ is reduced for all data sets for all minimum support thresholds. For example, if T10I4D100K is partitioned into 2 fragments, $|C_2^G|$ decreases from **16** for ***Sequent*** to 4 for ***Algorithm3*** with the support threshold **0.01**. This reduction is also present when considering other real data sets that are partitioned into 2 fragments. Its value reduces from **152** for ***Sequent*** to **46** for ***Algorithm2*** with the threshold **0.01** for real data set WebView-1, and its value reduces from **1,820** for ***Sequent*** to **756** for ***Algorithm2*** with the threshold **0.005** for very large data set BMS-POS.

Moreover, if data sets are partitioned into 5 fragments, the gap is even greater. For example, if T10I4D100K is partitioned into 5 fragments, $|C_5^G|$ decreases from **48** for ***Sequent*** to **24** for ***Algorithm1*** with the threshold **0.01**. $|C_5^G|$ decreases from **698** for ***Sequent*** to **434** for ***Algorithm3*** with the threshold **0.005**. When WebView-1 is partitioned into 5 fragments, $|C_5^G|$ decreases from **425** for ***Sequent*** to **93** for ***Algorithm2*** with the threshold **0.01.** Exceptional performance for BMS-POS when data set is partitioned into 5 fragments for ***Algorithm2***: with the threshold **0.01** the reduction is from **2,263** to only **222** as well as with the threshold **0.005** it reduces significantly from **10,718** to only **1,192**.

Secondly, another interesting and encouraging trend can be found in the growth of the number of common candidates between LP$_i$ for fragmented data sets. For example, if data sets are partitioned into 2 fragments, this common number increases from **152** to **203** for ***Algorithm1*** with WebView-1 and the threshold **0.01**.

In addition, if data sets are partitioned into 5 fragments, this common number increases dramatically from **689** to **1,404** for ***Algorithm2*** with BMS-POS and the threshold **0.01** as well as from **2,346** to **4,563** for ***Algorithm3*** with the same data set and the threshold **0.005**.

For comparison, we can compare these three pre-processing methods (***Agorithm1, Agorithm2, Agorithm3***) based upon several metrics. Firstly, the number of *Global candidates* of ***Algorithm1*** is less than those of the others for almost data sets and minimum support thresholds. In contrast, ***Agorithm3*** has the highest number of *Global candidates* for all data sets and all minimum support thresholds. Because there are different sizes of output clusters of ***Agorithm3***, so the *Local frequent itemsets* of clusters are more different and the number of them is higher. As a result, the number of *Global candidates* is higher. For ***Agorithm2***, with very big real data set (BMS-POS) and the higher number of fragments (5 fragments), the output result is more benefit than those of the other methods. For example, the numbers of *Global candidates* are **222** with threshold **0.01** and **1,192** with threshold **0.005** in comparison with those of ***Agorithm1*** are **894** and **4,075**, respectively. Secondly, as mentioned in section 4.3, ***Agorithm2*** can save much more computation of the similarity function that the other methods. This is very efficient when the data sets have many different items. Consequently, the better methods can be selected depending on the characteristics of data sets such as the number of all items, the number of all transactions and the given minimum support threshold.

In summary, the figures from two tables show that the all data pre-processing methods can significantly improve the Partitioning approach. It is delivered in form of two strongly related benefits; reduction of the number of *Global candidates* requiring the final check and increase of the common candidates numbers that don't require any additional checks.

	Sequent (no processing)	**Clustering** (Algorithm1)	**New similar Function** (Algorithm2)	**New Construction** (Algorithm3)
T10I4D100K				
1-fragment: **385** Frequent Itemsets				
2 fragments				
LP1	385	385	383	385
LP2	387	386	391	389
C_2^G	***16+ (378)***	***3 + (384)***	***12 + (381)***	***4 + (385)***
5 fragments				
LP1	392	387	391	383
LP2	381	387	394	385
LP3	393	384	390	388
LP4	386	388	395	391
LP5	390	388	391	393
C_5^G	***48+ (366)***	***24+ (375)***	***31 + (379)***	***33 + (381)***
WebView-1				
1-fragment: **208** Frequent Itemsets				
LP1	227	210	196	184
LP2	229	213	232	262
C_2^G	***152+ (152)***	***17+ (203)***	***46 + (191)***	***78 + (184)***
5 fragments				
LP1	284	226	220	154
LP2	197	221	212	186
LP3	241	213	228	231
LP4	255	207	228	270
LP5	266	205	216	353
C_5^G	***425+ (92)***	***74+ (181)***	***93 + (182)***	***311 + (154)***
BMS-POS				
1-fragment: **1,503** Frequent Itemsets				
LP1	1,400	1,512	1,348	1,378
LP2	1,662	1,498	1,688	1,655
C_2^G	***390+ (1,336)***	***60+ (1,475)***	***342 + (1,347)***	***277 + (1,378)***
5 fragments				
LP1	1,996	1,150	1536	1,192
LP2	1,334	1,471	1505	1,359
LP3	744	1,864	1514	1,576
LP4	1,348	1,822	1474	1,700
LP5	2,885	1,364	1521	1,858
C_5^G	***2,263+ (689)***	***894+ (1,121)***	***222 + (1,404)***	***1,356 + (1,192)***

Table 5.2: The figures with a support threshold 0.01

	Sequent (no processing)	**Clustering** (Algorithm1)	**New Similar Function** (Algorithm2)	**New Construction** (Algorithm3)
T10I4D100K				
1-fragment: **1,073** Frequent Itemsets				
2 fragments				
LP1	1,079	1,068	1099	1091
LP2	1,101	1,092	1084	1079
C_2^G	***158 + (1,011)***	***70 + (1,045)***	***213 + (985)***	***88 + (1041)***
5 fragments				
LP1	1,150	1,089	1087	1040
LP2	1,141	1,110	1088	1032
LP3	1,248	1,059	1092	1111
LP4	1,110	1,135	1094	1161
LP5	1,120	1,098	1136	1150
C_5^G	***698 + (893)***	***373 + (941)***	***268 + (976)***	***434 + (960)***
WebView-1				
1-fragment: **633** Frequent Itemsets				
LP1	644	659	581	529
LP2	755	641	744	859
C_2^G	***503 + (448)***	***94 + (603)***	***191 + (567)***	***330 + (529)***
5 fragments				
LP1	1,107	779	766	413
LP2	489	733	722	552
LP3	839	676	751	770
LP4	894	663	648	898
LP5	977	597	615	1601
C_5^G	***1,806 + (271)***	***497 + (493)***	***538 + (485)***	***1643 + (404)***
BMS-POS				
1-fragment: **6,017** Frequent Itemsets				
LP1	5,419	6,024	5,699	5,502
LP2	6,709	5,972	6,385	6,559
C_2^G	***1,820+ (5,154)***	***348+ (5,824)***	***756 + (5,664)***	***1,061 + (5,500)***
5 fragments				
LP1	8,480	4,339	6,023	4,586
LP2	4,975	5,932	5,894	5,395
LP3	2,541	7,530	6,163	6,313
LP4	5,177	7,443	6,017	6,835
LP5	12,755	5,289	6,109	7,473
C_5^G	***10,718+ (2,346)***	***4,075+ (4,191)***	***1,192 + (5,467)***	***6,029 + (4,563)***

Table 5.3: The figures with a support threshold 0.005

6 Conclusion

As an extension of our previous work, this paper considers the extension of data pre-processing approach for further performance improvements in frequent itemsets computation. We show that the composition of fragments and the number of fragments generated, impact on the size of the data used by the original Partition algorithm.

We proposed a series of data pre-processing methods, mainly to demonstrate that the performance of computing frequent itemsets can be improved by data partitioning. The new algorithms are proposed based on different definitions of similarity function and different cluster constructions which are applied for data clustering to get more efficient. Figures from the experiments show that these data pre-processing methods offer good benefits already.

In addition, the comparison among these methods has been conducted. In particular, the better method can be applied depending on the characteristics of input data sets and the given minimum support threshold.

There is an interesting question for future work which is how to identify the methods that will deliver an even better partition for the original data sets.

7 References

Nguyen S., Orlowska M. (2005): Improvements in the Data Partitioning Approach for Frequent Itemsets Mining. *Proc. 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 05),* Springer.

Agrawal R., Imielinski T., Swami A. (1993): Mining association rules between sets of items in lagre database. *Proc. 1993 ACM SIGMOD Int. Conf. on Management of Data*, Washington DC, USA, 22:207-216, ACM Press.

Agrawal R., Srikant R. (1994): Fast algorithms for mining association rules. *Proc. 20th Int. Conf. Very Large Data Bases,* Morgan Kaufmann, (487 - 499)

Brin S., Motwani R., Ullman D.J., Tsur S. (1997): Dynamic Itemset Counting and implication rules for masket basket data. *Proc. ACM SIGMOD 1997 Int. Conf. on Management of Data,* (255 - 264)

Houtsma M., Swami A. (1995): Set-oriented mining for association rules in relational database. *Proc. 11th IEEE Int. Conf. on Data Engineering,* Taipei - Taiwan, (25 - 34)

Lin J.L., Dunham M.H. (1998): Mining association rules: Anti-skew algorithms. *Proc. 14th IEEE Int. Conf. on Data Engineering,* Florida

Pudi V., Haritsa J. (2003): ARMOR: Association rule mining based on Oracle. *Workshop on Frequent Itemset Mining Implementations (FIMI'03 in conjunction with ICDM'03)*

Savasere A., Omiecinski E., Navathe S. (1995): An efficiant algorithms for mining association rules in large database. *Proc. 21th Int. Conf. Very Large Data Bases,* Swizerland

Toivoven H. (1996): Sampling Large Databases for association rules. *Proc. 22th Int. Conf. Very Large Data Bases*, Mumbai, India

Ron K., Carla B., Brian F., Llew M., Zijian Z. (2000) KDD-Cup 2000 organizers' report: Peeling the onion. *SIGKDD Explorations*, 2(2):86-98

Goethals B. (2002): Survey on frequent pattern mining. *University of Helsinki*

Mueller A. (1995): Fast sequential and parallel algorithm for association rules mining: A comparision. *Technical Report CS-TR-3515,* Uni. of Maryland

Zhu T. (2004): The Apriori algorithm implementation, *http://www.cs.ualberta.ca/~tszhu/,* Accessed 2004

Han J., Pei J., Yin Y., Mao R. (2004): Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery,* Kluwer Academic Publishers, (8): 53-87

Zaki M.J. (2000): Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3): 372-390

Zhang S., Wu X. (2001): Large scale data mining based on data partitioning. *Applied Artificial Intelligence* 15:129-139

Ganesan P., Garcia-Monila H., Widom J. (2003): Exploiting hierarchical domain structure to compute similarity. *ACM Transactions on Information systems*. Vol. 21, No. 1, January 2003.

A Two-Phase Rule Generation and Optimization Approach for Wrapper Generation

Yanan Hao **Yanchun Zhang**

School of Computer Science and Mathematics
Victoria University
Melbourne, VIC, Australia

haoyn@csm.vu.edu.au

yzhang@csm.vu.edu.au

Abstract

Web information extraction is a fundamental issue for web information management and integrations. A common approach is to use wrappers to extract data from web pages or documents. However, a critical issue for wrapper development is how to generate extraction rules. In this paper, we propose a novel two-phase rule generation and optimization (2P-RULE) approach for wrapper generation. 2P-RULE consists of internal rule optimization (IRO) process and external rule optimization (ERO) process. In IRO, a user, through a GUI interface, firstly creates a mapping from useful values in web page to a schema specified by the users according to target web information. Based on the mapping, the system automatically generates a rule list for the schema. Whereas in ERO, the user can create multiple mappings to generate further rule lists. All the acquired rule lists are merged and refined into one optimized rule list, which is expressed with XQuery as the final extraction rules. Experiments show that our 2P-RULE approach is suitable for extracting information from web pages with complex nested structure, and can also achieve better precision and recall ratio.

Keywords: Web, extraction, wrapper, rule optimization, XQuery.

1 Introduction

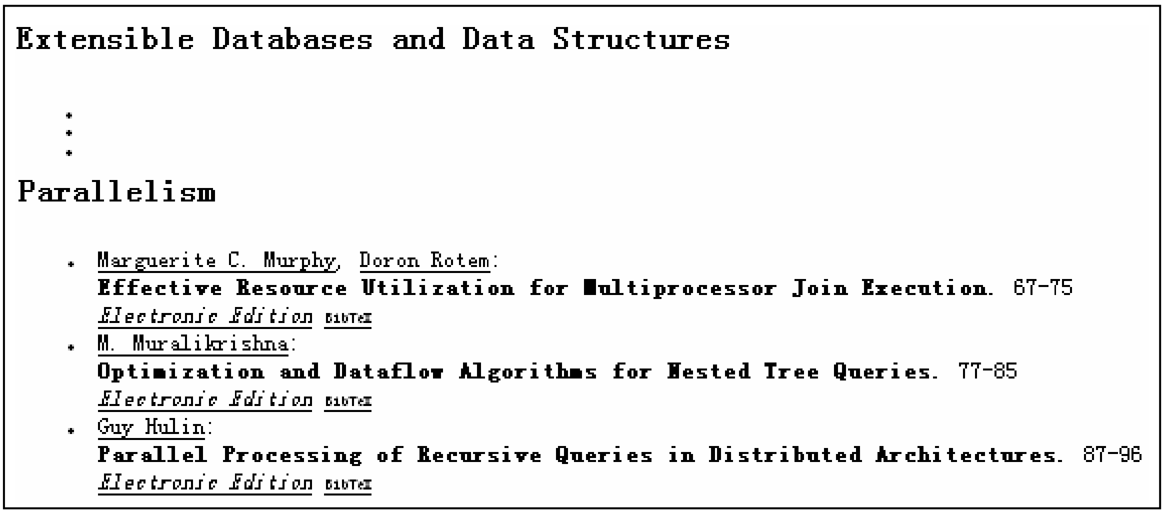
With the rapid development of Internet, World Wide Web has already become the most important and potential information resources (Lawrence S. and Giles L. 1999). HTML language aims at the visual presentation of data in web browsers, while it lacks of schema and semantic information for efficient management and retrieval web information. Most of valuable web information is in HTML form even though XML has been more and more popular today. So researchers propose wrappers technology to extract data from web pages and convert the information into a structured format. However, a critical issue for wrapper development is how to generate extraction rules for extracting data from web pages having similar structures.

Many information extraction tools (Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva, Juliana S. Teixeira., 2002) have been developed to extract data on the web. These works can be classified into three categories: manual approach, automatic approach (Soderland S. 1999, Arasu, A., Garcia-Molina, H. 2003, Hu D., Meng X. 2005, Ma L., Shepherd J. 2004) and semi-automatic approach (Liu L., Pu C., Han W. 2000, Han W., Buttler D., Pu C. 2001, Arnaud S. and Fabien A. 1999, Sahuguet A. and Azavant F. 1999, Baumgartner R., Flesca S., Gottlob G. 2001, Baumgartner R., Ceresna M., Gottlob G., Herzog M., Zigo V. 2003, Meng X., Wang H., Hu D., Chen L. 2003). In the first category, extraction rules are programmed manually which can be very hard for common users; in the second category, (Soderland S. 1999) introduces a machine learning approach. It utilizes the structures of sentences and relationships between idioms and words to create rules automatically; (Arasu A., Garcia-Molina, H. 2003) and (Hu D., Meng X. 2005) propose item identification techniques via HTML path and templates for automatic data extraction from web pages; (Ma L., Shepherd J. 2004) discovers the semantic pattern for an identified region of a document via inference, apposition and analogy. The drawbacks of these four systems are their limited expressive power of extraction rules and only suitable for simple record schema. Semi-automatic approach requires user interactions to build mappings between schema and content in web pages, after that extraction rules are derived for extracting web pages having similar structures. In the third category, XWrap (Liu L., Pu C., Han W. 2000, Han W., Buttler D., Pu C. 2001) only have good performance on web pages with distinct region features; In W4F (Arnaud S. and Fabien A. 1999, Sahuguet A. and Azavant F. 1999), expertise is required to program part of extraction rules manually; In Lixto (Baumgartner R., Flesca S., Gottlob G. 2001, Baumgartner R., Ceresna M., Gottlob G., Herzog M., Zigo V. 2003), the extraction rules are expressed in Elog language, which is difficult for the optimization and refinement of extraction rules; secondly, it require users to specify some external and internal conditions for extraction rules, thus the effectiveness and robustness of rules relies on user's action. SG-Wrapper (Meng X., Wang H., Hu D., Chen L. 2003) makes some improvements in rule generation and expression. But its extraction rules may be invalid when

Copyright (c) 2006, Australian Computer Society, Inc. This paper appeared at the *Seventeenth Australasian Database Conference (ADC2006)*, Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

the nested structures of web page do not match the pre-defined user schema.

All the tools above ignore the problem of usable features of web page and their performance in constructing extraction rules. It is very important to the robustness of extraction rules. In this paper, based on the analysis of usable features of web pages and their performance in constructing extraction rules, we propose



Extensible Databases and Data Structures

-
-
-

Parallelism

- Marguerite C. Murphy, Doron Rotem:
  Effective Resource Utilization for Multiprocessor Join Execution. 67-75
  Electronic Edition BibTeX
- M. Muralikrishna:
  Optimization and Dataflow Algorithms for Nested Tree Queries. 77-85
  Electronic Edition BibTeX
- Guy Hulin:
  Parallel Processing of Recursive Queries in Distributed Architectures. 87-96
  Electronic Edition BibTeX

Figure 1: A sample HTML fragment from VLDB

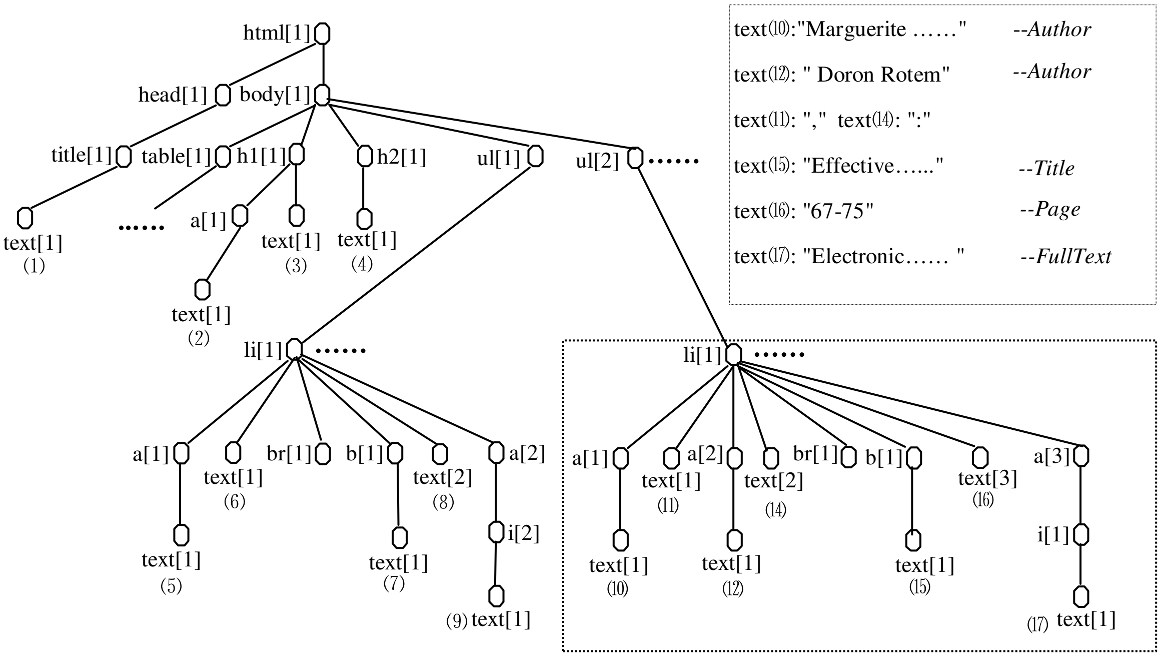
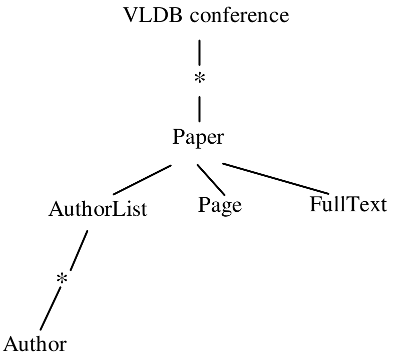


Figure 2: DOM tree



```
<!DOCTYPE  VLDBConference [
<!ELEMENT  VLDBConference  ((Paper)*)>
<!ELEMENT  Paper  (AuthorList, Page, FullText)>
<!ELEMENT  AuthorList ((Author)*) >
<!ELEMENT  Author  (#PCDATA)>
<!ELEMENT  Page  (#PCDATA)>
<!ELEMENT  FullText  (#PCDATA)> ]>
```

Figure 3: The schema tree and DTD definition

a novel two-phase rule generation and optimization (2P-RULE) approach. 2P-RULE consists of internal rule optimization (IRO) process and external rule optimization (ERO) process. In IRO, a user, through a GUI interface, firstly creates a mapping from useful values in web page to a schema specified by the users according to target web information. Based on the mapping, the system automatically generates a rule list for the schema. Whereas in ERO, the user can create multiple mappings to generate further rule lists. All the acquired rule lists are merged and refined into one optimized rule list, which is expressed with XQuery as the final extraction rules. Then the information extraction is simply a process of executing XQuery statements in any XQuery engine. The query result can be utilized by common users and further by applications. Experiments show that our 2P-RULE approach can extract information from web pages with complex nested structure and can also achieve better precision and recall ratio.

This rest of this paper is organized as follows. Section 2 introduces the presentation and semantic models for web information extraction. In section 3 we present the basis of extraction rules generation. Section 4 discusses the extraction rules and their optimization steps. Section 5 reports the experimental results. Finally conclusion and future work are discussed in section 6.

2 Representation and semantic models

2.1 Representation model of web documents

An HTML document is a text file containing markup tags. The Document Object Model (DOM) represents a document as a tree. Every node of the tree represents a HTML tag, or a text value inside an HTML tag. The tree structure describes the whole HTML document, including the child, parent, or sibling relationship between tags and text values on the page. DOM allows us to locate elements in the tree with XPath (XML Path Language) expressions. We choose DOM as the representation model of HTML information in our system. All the operations in our system are based on DOM tree.

As an example, in Figure 1 we give an HTML fragment of the web pages at http://www.informatik.uni-trier.de/~ley/db/conf/vldb/. Figure 2 shows its DOM tree structure. In Figure 2 each node tag is followed by an *ordinal*. An ordinal is the order of a node among all its siblings of the same tag. For the convenience of reference, we also assign each text node a number. For example, the following XPath expression "html[1]/body[1]/ul[2]/li[1]/text()[3]" identifies the data value 67-75 in the HTML fragment.

2.2 Semantic model

DOM tree is only the internal expression of web documents. It can effectively process data in documents, but it may not reflect the potential semantic information in document data. In this paper we choose XML as the semantic model and its schema is defined by DTD. The defined DTD can be easily represented in the form of a tree structure, which is called a **schema tree.** In our system, through a GUI interface, a user can easily specify the schema tree according to the target web documents. An example of the schema tree and the DTD corresponding to that can be seen in Figure 3. The '*' in Figure 3 denotes zero or more occurrences. For example, the '*' between *AuthorList* and *Author* means a *AuthorList* may have zero or more authors. The schema tree can support some node types corresponding to the data types defined by DTD. Using regular expression, we define these supported node types as follows:

atomic object (AO): (#PCDATA)
set object (SO): ((atomic object)*)| ((tuple object)*)
tuple object (TO): (a_1, a_2… a_n), where a_i (1<=i<=n) is (an atomic object | a set object | a tuple object)

We call each sub element of a set object a **member object (MO).** If the member is an atomic object, then it will be called a **member atomic object (MAO***).* If the member is a tuple, then it is a **member tuple object (MTO).** In general, all nodes in the schema tree are called **semantic objects.**

For example, referring the schema tree in Figure 3, both *Page* and *FullText* are atomic objects (AO); *VLDBconference* and *AuthorList* are set objects (SO); A*uthor* is a member atomic object (MAO); and *Paper* is a member tuple object (MTO).

After creating a schema tree, the user needs to create a mapping from the contents of web pages to the schema tree. To create such a mapping, the user simply selects semantic objects in the schema tree, and then highlights the corresponding content on web pages. Based on the mapping, the system automatically generates *rule segments* (see section 4) for each semantic object of the schema tree.

3 Basis of extraction rules generation

3.1 Analysis of usable web features

In most information extraction systems, extraction rules are mainly expressed with five features of web pages, including structure, position, semantics, display and references. Feature selection determines the performance of extraction rules. *Structure feature* is the paths of DOM tree. With path expression we can navigate HTML page easily, so it can be a basic feature for constructing extraction rules. But structure feature has weak differentiating ability, and extraction rules only containing structure feature may have low precision rate. For example in Figure 2, the path expression html/body/ul/li/text() can locate many nodes. *Position feature* includes *ordinal* (section 2.1) and *boundary*. Boundary is a left or right sibling node. In Figure 2, node *text(16)* has the left boundary *b* and the right boundary *a*. *Position feature* relies on the structure of web pages, so using position will decrease the coverage of extraction rules but increase the differentiating ability at the same time. *Display feature* includes font, font-size, colour and alignment. It limits nodes by node attributes in each location step of DOM path. Normally, similar or correlative contents in web pages have same display features, so selecting display feature generating rules will increase the coverage. But extraction rules can be inaccurate when multiple instances of one semantic object (say an *author* object) have same display features. *Semantic feature* is a common conceptual or content feature of data to be extracted. For example, the value of *Price* often contains a character '$'. Semantic feature lacks in differentiating ability but makes extraction rules more flexible. *Reference feature* is the hyperlink information in web pages. It has little effect on the robustness of extraction rules and we do not discuss it in this paper.

The goal of extraction rules is to have good coverage and differentiating ability. Based the analysis above, we firstly select DOM path, semantic and display features to form extraction rules, which do not rely on the structure of web pages and have good coverage ability, then add ordinal and position information to extraction rules to gain good differentiating ability.

3.2 Mismatches between schema tree and DOM tree

To extract data from similar web pages or documents, a user first defines a schema tree of the target data. The schema tree reflects the user's view of extracted data. In order to make semantics clear, the user can create a schema tree with nested structure. For example in Figure 3, a *AuthorList* object is created to represent all the authors in one paper. Since the user creates a schema tree through a GUI interface without knowing the details of HTML documents, sometimes the nested structure of the schema tree may not match its corresponding structure in DOM tree. The mismatch happens when a complex nested object in the schema tree does not have a corresponding node in DOM tree, while its sub components are directly listed. For example in Figure 3, *AuthorList* is a complex nested object whose sub components are *Author*s. Consider the DOM tree in Figure 2. The sub-tree in rectangle corresponds to the semantic object *Paper*, and its DOM paths correspond to *author*, *author*, *title*, *page* and *FullText* respectively. No node corresponds to the semantic object *AuthorList* and all the *Author* nodes are listed directly.

Mismatches between schema tree and DOM tree are main difficulties for extracting complex nested objects. Typically, there are three sorts of mismatches:

Single set object mismatch. There is only one set object in the schema tree that does not match its corresponding DOM tree structure, for example the *AuthorList* object in Figure 3. Let us suppose *author* and *FullText* have same display and structure features firstly. Further more, since the count of authors is variable, position feature can not differentiate them either. In this case, we introduce a new position feature *big boundary*. The *left big boundary* of a set object is a sequence of nodes, which are all the left siblings of the leftmost sub-tree spanned by the set object; the *right big boundary* of a set object is a sequence of nodes, which are all the right siblings of the rightmost sub-tree spanned by the set object. For example in Figure 2, the set object *AuthorList* corresponds to nodes text (10) and text (12). The *left big boundary* of *AuthorList* is null while its right *big boundary* is {text, br, b, text, a}. By *big boundary* feature, we can differentiate between member objects (*Author*) of a set object (*AuthorList*) and sibling objects (*FullText* or Page) of the set object.

Multiple set objects mismatch. There are several set objects in the schema tree, and none of them matches the corresponding DOM tree structure. For example, a user may define another set object *AddressList* as a sibling of *AuthorList*. Using the big boundary, we can still differentiate these two set objects if member object *Author* and *Address* can be differentiated. But if *Author* and *Address* have same features, these two set objects can not be differentiated in our system.

Member tuple object mismatch. In this case, a member tuple object does not have the corresponding node in DOM tree while its sub components are listed directly. The member tuple objects may not be differentiated. For example in Figure 2, if the node *Li[1]* , which corresponds to the semantic object *Paper*, does not exist, all the authors can not be differentiated, i.e. which author belongs to which paper. In our system, we add a virtual node to each member tuple object to solve the *member tuple object mismatch.*

In this section, we analyse the usable web features and the mismatches between schema tree and DOM tree. They are the basis of rules generation in our system. In section 4, we will describe our approach to generate and optimize extraction rules.

4 Generation and optimization

4.1 Rule segments

According to section 3, we distribute all the usable web features in six sorts of rule segments. The initial extraction rule for each semantic object will be composed of several rule segments. Different semantic object has different composition of rule segments as the initial extraction rule. Figure 4 gives the BNF definition of all rule segments.

PureAttrPathExp(P): We use the first letter **P** to denote PureAttrPathExp. Abbreviation for other rule segments is similar. This rule segment is called *pure-attribute path expression*, each location step of which only contains attributes limitation. If there exists attributes in a location step, then we choose all the equations of "[attribute name= *attribute value*]" as predicates to limit nodes sequence, or we do not select any predicates in this step. For example in Figure 2, html/body[@bgcolor="#ffffff"]/ul/li/text() is a pure-attribute path expression. It can locate text(11), text(16) and text(14).

AttrOrdPathExp(A): We call it *attribute-ordinal path expression*, each location step of which only contains attributes or ordinal limitation(except location steps with the node test *text()*). If it contains attributes, then we use all the equations "[attribute name= *attribute value*]" as predicates, or we use ordinals to limit nodes sequence. In Figure 2, html[1]/body[@bgcolor="#ffffff"]/ul[2]/li[1]/text() is an *attribute-ordinal path expression.*

OrdPathExp(O): This sort of rule segment is called *ordinal-path expression,* each location step of which only contains ordinal limitation (including location steps with the node test *text()*). For example,

```
PureAttrPathExp::= (NodeName("[@"AttrName"="AttrValue"]")* ) |
(NodeName("[@"AttrName"="AttrValue"]")*"/"PureAttrPathExp) | NULL
AttrOrdPathExp::=(NodeName("["num"]" | ("[@"AttrName"="AttrValue"]")*)) | NULL
| (NodeName("["num"]" | ("[@"AttrName"="AttrValue"]")*)"/"AttrOrdPathExp)
OrdPathExp::=(NodeName"["num"]") | (NodeName"["num"]""/"OrdPathExp) | NULL
TxtFeaturePredicate::=("[contains(string(.) ," TxtFeatureValue ")]") +
Big_BoundaryPredicate::=(("[count(../" NodeName "after .)=" Num "]" ) | ("[count(../" NodeName "before .)="
Num "]")) +
Small_BoundaryPredicate::=
(("[count((../* after .)[1])=0 ]") | ("[((../"Nodename " after" ".)[1])=((../*  after .)[1])]") )
(("[count((../*  before .)[1])=0 ]") | ("[((../"Nodename " before" ".)[1])=((../*  before .)[1])]"))
```

Figure 4: The BNF definition of rule segments

html[1]/body[1]/ul[2]/li[1]/text()[3] is a OrdPathExp for the semantic object *Page*(see Figure 2) .

OrdPathExp(O): This sort of rule segment is called *ordinal-path expression,* each location step of which only contains ordinal limitation (including location steps with the node test *text()*).For example, html[1]/body[1]/ul[2] /li[1]/text()[3] is a OrdPathExp for the semantic object *Page*(see Figure 2) .

TextFeaturePredicate(T): This rule segment is called *text-feature predicate.* It is a form of predicate in XPath requiring the text content of a node (for non-leaf node, it will be a concatenation of string-values of all its descendants) in DOM tree contain some fixed text value. We only use this predicate in the last location step of path expressions. For example, html/body[@bgcolor=”#ffffff”] /ul/li[contains(string(.)，”Electronic Edition”)] means the text content of nodes limited by the last location step must contain the string “Electronic Edition”.

Big _BoundaryPredicate(B): This rule segment is called *big boundary predicate.* It contains *left big boundary predicate* and *right big boundary predicate*. We introduce this rule segment for the extraction of complex nested objects, say *AuthorList* in Figure 3(section 3.2).

Small_BoundaryPredicate(S): This rule segment is called *small boundary predicate*. This predicate is to limit a node by its immediate left sibling and right sibling. In our system we only apply this segment to atomic objects and only to the last location step with the node test “text ()”, because text nodes in DOM specification are regarded as virtual nodes, and in most circumstances they do not have sibling nodes, as they do not rely on the structure of web pages.

In these six rule segments, the first three rule segments are path expressions, in which PureAttrPathExp has the best coverage ability, OrdPathExp has the worst coverage ability, and AttrOrdPathExp is middle; the rest three rule segments are all predicates. They can only be used together with the first three path expressions.

After the user creates a mapping from the contents of web pages to the schema tree, system automatically generates rule segments for each semantic object of the schema tree. As for a member object, the two segments *AttrOrdPathExp* and *OrdPathExp* do not contribute to its extraction rule, since they both contain ordinal feature and it can be invalid due to the variable count of member objects. System does not generate rule segments for a plain tuple object (i.e. it is not a member tuple object). The plain tuple object appears only once, and if its sub components can be extracted, they definitely belong to this tuple object. Thus the extraction rule for plain tuple object itself is not needed and we only need to compose rules for its sub components. But for a member tuple object, it can appear many times, so the corresponding extraction rule is needed to decide which sub component belongs to which member tuple object instance.

The rule segments of each semantic object are marked by “ √ ”in Table 1. Please see section 2.2 for the definition of semantic objects. For example, from Figure 2 we can conclude that html/body[@bgcolor=“#ffffff”]/ul/li/text() is a rule segment (P) for the semantic object *Author* (MAO).

Semantic objects **Rule segments**	**AO**	**MAO**	**MTO**	**SO**
PureAttrPathExp	√	√	√	√
AttrOrdPathExp	√			√
OrdPathExp	√			√
TxtFeaturePredicate	√	√	√	√
Big_ BoundaryPredicate				√
Small_BoundaryPredicate	√	√		

Table 1: Rule segments for each semantic object

4.2 Optimization of extraction rules

For one semantic object, its rule segments can have different combinations. Each combination is called an *initial rule* for the semantic object. Different combination of rule segments can generate different initial extraction rules. In this section, we describe our two-phase rules

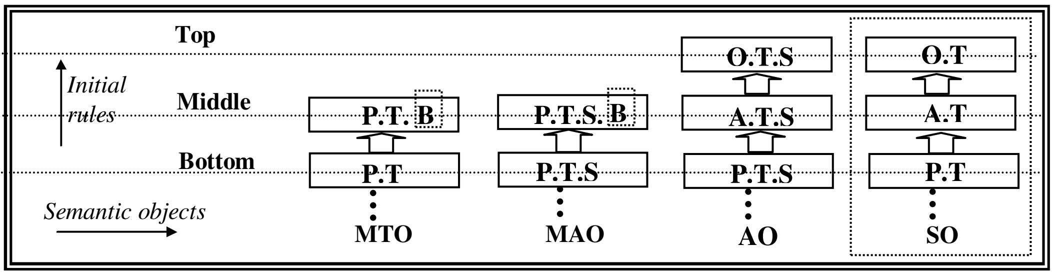


Figure 5: Initial rules for each semantic object

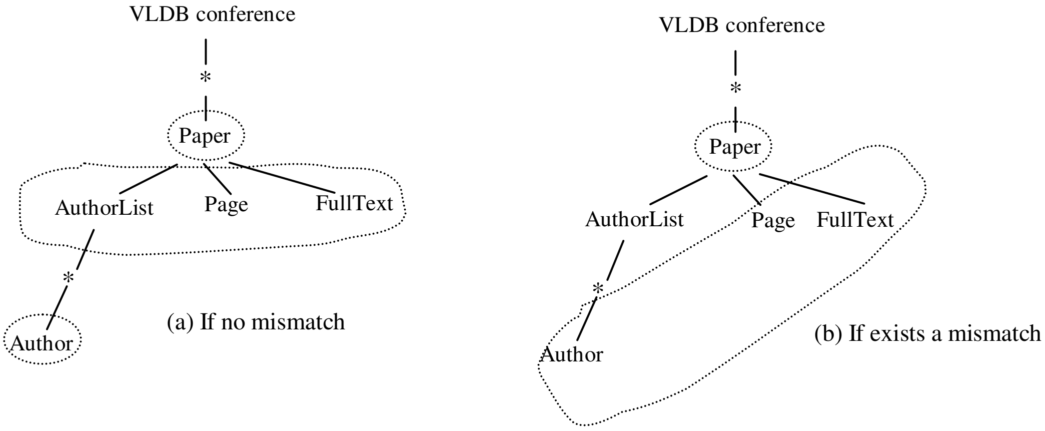


Figure 6: Grouping of semantic objects

(2P-RULE) optimization approach. 2P-RULE consists of the internal rule optimization (IRO) process and the external rule optimization (ERO) process.

4.2.1 Internal optimization (IRO)

In IRO, system automatically selects an optimized initial rule for each semantic object and generates a rule list for the schema.

(1) Initial rules

Among the six rule segments, the first three path expressions are XPath statements, and the rest three rule segments are predicates. The effective combination, initial rule, should be one and only one path expression plus several predicates. We combine both TxtFeaturePredicate and Small_BoundaryPredicate together with path expressions, for these two segments do not rely on the structure of web page. For Big_BoundaryPredicate, it is generated to deal with *set object mismatch.* When there exist set objects mismatches, we do not generate initial rules for SO and add its Big_BoundaryPredicate to member objects MTO and MAO. While when there are no mismatches, the initial rules for SO are {P.T, A.T, O.T} and the Big_BoundaryPredicate is removed. All the possible initial rules for each semantic object are listed in Figure 5.

Bottom. The initial rules at bottom have better coverage ability. They select DOM path, display feature and semantic feature. Although S rule segment is added to the atomic object, it does not rely much on the structure of web page. The shortcoming of bottom combination is that it lacks of differentiating ability.

Middle. The middle level of initial rules are acquired by adding position feature to bottom combination to get better differentiating ability. Particularly, member objects get new rule segment B; both atomic objects and set objects replace their rule segment P as A, which actually complements some ordinals for the location steps without predicates. The better differentiating ability is at the expense of decreasing of coverage.

Top. The top initial rules are only available for atomic objects and set objects. They add further position feature to the middle initial rules. Comparing with middle initial rules, their differentiating ability is further increased but coverage ability is further decreased.

(2) Optimization and selection

The goal of extraction rules is to have good coverage and differentiating ability. In Figure 5 we can see, from bottom to top, the coverage ability of initial rules is increasingly good and the differentiating ability of initial rules is increasingly poor. Our basic idea of rules optimization is to select the first “good” initial rule from bottom to top for each semantic object. Here “good” means having no collision with the selected initial rules for other semantic objects, i.e. they do not locate identical nodes.

In our system, the DOM paths forming into extraction rules are relative. Extraction rules for sub objects will locate nodes based on the extraction results of parent objects, so only the initial rules having the same base domain are possible to collide. For example in Figure 2, suppose “html/body/ul/li” is an initial rule for *Paper*, and the initial rule for *FullText* is “a/i/text()”. Obviously, these two initial rules do not collide, because “a/i/text()” is to locate nodes based on the sub tree *li*, and "html/body/ul/li" is to locate nodes based on the whole DOM tree. To detect potential collisions, we group the semantic objects according to the base domain of their initial rules. The semantic objects having sibling relationship will belong to the same group finally.

For example, Figure 6 gives the two possible groupings of schema tree in Figure 3. For the HTML fragment in Figure 1, we should choose the grouping (b).

Definition 1 [Containment relationship] Let XPathExp1 and XPathExp2 be two XPath expressions. We say XPathExp1 contains XPathExp2, if

1. They have the same DOM path, and
2. The set of predicates in each location step of XPathExp1 is a subset of the set of predicates in each corresponding location step of XPathExp2.

The containment relationship is denoted as XPathExp2 ⊂ XPathExp1. For example, suppose XPathExp1=A/B[@colour="1"]/C, and XPathExp2=A/B [@calor= "1"][@high= "6"]/C, then XPathExp2 ⊂ XPathExp1. This means the nodes set located by XPathExp2 are a subset of nodes set located by XPathExp1 and so collision occurs. Semantic objects in same group have the same DOM path, so we can use containment relationship to detect potential collisions between them.

Definition 2 [Invalid rule] Let so_i be the *i*th semantic object in a schema tree, $max(so_i)$ be the total count of initial rules for so_i and so_i-r_m($1 \leqslant m \leqslant MAX(SO_i)$) be the *i*th initial rule for so_i. We say so_i-r_m is an invalid rule，if and only if $\exists$ so_j ($j \neq i$) such that so_i-r_m contains so_j-r_n($1=<n<=max(so_j)$). Here *m* and *n* are the ordinals of initial rules for semantic objects. For each semantic object, its initial rules are numbered from bottom to top beginning with 0 (See Figure 5).

We say that an initial rule is valid if it is not an invalid rule. Based on the analysis and definitions above, the process of selection and optimization of rules for semantic objects is described as below:

Step 1: Group all the semantic objects.
Step 2: Group all the initial rules by the grouping of semantic objects.
Step 3: In each initial rules group, from bottom to top, Find the first valid initial rule as the optimized rule for each semantic object. All the optimized rules constitute an optimized rule list for the schema. We formulate this step into algorithm 1.

Algorithm 1 Internal optimization

Input: G= {g_i | i=1, 2 …}, groups obtained in step 2

Output: O= {opti | i=1, 2 …}, O is an optimized rule list for the schema, in which opt_i contains the optimized rules for semantic objects in group g_i

Description:

```
for each group g_i in G do
  Optimize (g_i);
Function optimize (g)
1: s:=0; collision:=False; // s is the ordinal of an initial rule
2: for each semantic object $so_l$ in group g do
3:   if max ($so_l$) =1 then
4:     // only one initial rule
5:       $opt_g$-$so_l$ = $so_l$ -$r_0$ ;
6:       continue;
7:   endif
8:   if exists i, j such that $so_j$-$r_i$ ⊂ $so_l$-$r_s$ then
9:     // $so_l$-$r_s$ is an invalid rule
10:    collision: =True;
11:  endif
12:  if collision then
13:      if ($so_l$ is a MO) and s=1 then
14:          exit; // Can not extract this web page.
15:      else
16:          s: =s+1; // go up
17:          goto line 8;
18:      endif // if $so_l$ is a member object
19:  endif // if collision
20:  $opt_g$-$so_l$ = $so_l$-$r_s$; // Obtain the optimized rule for $so_l$
21:  $opt_g$-$so_l$ → $opt_g$; //add $opt_g$-$so_l$ to $opt_g$
22: end // end for
```

For example, after IRO we can obtain an optimized rule list for the semantic objects in Figure 3:

- *Paper*: html/body[@bgcolor="#ffffff"]/ul/li [contains(string(.)，"Electronic Edition")]
- *Author*: a[left_big_boundary="", right_big_boundary ="text, br, b, text, a"]/text()
- *Page*: /text()
- *FullText*: a/li/text()[contains(string(.)，"Electronic Edition")]

4.2.2 External optimization (ERO)

We say a user makes a learning process, if the user creates a mapping from the web page to the schema. Based on the mapping, system generates an optimized rule list by IRO. If the user is not satisfied with the extraction results, she can make more sample learning processes. Each sample learning process will generate one optimized rule list. In ERO system merges and refines all the acquired rule lists into one optimized rule list, which is expressed with XQuery as the final extraction rules.

The whole merging and refining procedure is listed as below:

Step 1: For each semantic object.

(1) Find all its relevant optimized rules
(2) Partition these rules into several groups. In each group the rules have containment relationship with each other.
(3) Select the rule having the best coverage in each group and unit them into an optimized rule for the semantic object.

Step 2: Save all the optimized rules as the final optimized rule list for the schema.

We formulate these steps into algorithm 2.

Algorithm 2 External optimization

```
Input: Optimized rule lists: $opt_1, opt_2, \ldots, opt_k$
       $Opt_i = \{opt_i\text{-}so_1, opt_i\text{-}so_2, \ldots, opt_i\text{-}so_n\}$ $(0<i<k+1)$
Output: the final optimized rule list Opt for the schema.
Opt= $\{opt\text{-}so_1, opt\text{-}so_2, \ldots, opt\text{-}so_n\}$
Description:
for $m$=1 to n do
    partition the set $\{opt_l\text{-}so_m | 1<=l<=k\}$ into $s$ groups.
    In the $i$th ($1<=i<=s$) group, find $opt_{li}\text{-}so_m$ that contains
    all the other extraction rules for the semantic object $so_m$
    $opt\text{-}so_m = \bigcup_{i=1}^{s} (opt_{li} - so_m)$
endfor
```

4.3 XQuery expression

After ERO, we get the final optimized rule list for the schema. Each rule of the optimized rule list is expressed with an XPath expression, and each time it can only locate in DOM tree one semantic object instance of the schema. In order to locate all the semantic object instances, we translate the final rule list into a complete XQuery query statement as the extraction rule for the schema.

According to the final optimized rule list, we generate one FLWR expression for each semantic object, i.e.

- One FR expression (FOR statement and RETURN statement) for a member object.
- One LR expression (LET statement and RETURN statement) for a set object and an atomic object.

Finally, we organize all the FLWR expressions by the nested structure of the schema tree and form them into one XQuery statement. The information extraction is then a procedure of executing this XQuery statement in any XQuery engine.

As an example, let us suppose the final optimized rule list is the same as it in section 4.2.1. The XQuery statement is shown in Figure 7.

```
<vldb conference>
{FOR $paper IN (document("sample.xml")/html/body
 [@bgcolor="#ffffff"]/ul/li[contains(string(.),"Electronic
Edition")])
 RETURN
 <Paper>
   <AuthorList>
   {FOR $author IN $paper/ a[left_big_boundary="",
   right_big_boundary ="text, br, b, text, a"]/text()
   RETURN
     <Author>{$author}</Author>}
   </AuthorList>
  {LET $page:=$paper/text()
   RETURN
  <Page>{$page}</Page>}
  {LET $FullText:= $paper/ a/li/text()
   [contains(string(.),"Electronic Edition")]
   RETURN
  <FullText>{$FullText}</FullText>}
 </Paper>}
</ vldb conference>
```

Figure 7: Extraction rules expressed with XQuery

5 Experiments

Based on the optimization techniques above, we have developed a prototype system. Several experiments have been done on the three websites sites in table 2, which are already used for testing purposes by other information extraction tools.

We carry out our experiments on a Windows machine with a 2GHz Pentium IV and 512M main memory. For each website, the experiment procedure is listed as below:

Transformation. Because our prototype system uses XML as the presentation model of HTML information in web pages, and all the operations are based on DOM tree, all the web pages to be extracted to XML documents should be transformed into XML documents firstly. We use Tidy (HTML Tidy Library Project) to finish the transformation.

Creating schema tree. Select sample web pages, then require the user to create schema tree to represent the semantic information of data to be extracted.

Creating mappings. The user selects semantic objects in the schema tree firstly, and then highlights the corresponding content on web pages. Meanwhile text feature may be required.

Optimization. Execute the IRO and ERO process. System automatically generates optimized rule list expressed with XQuery.

Extraction. Execute the XQuery statement on an XQuery engine to extract data on other web pages in the website.

Analysis. We manually verify the extracted results.

5.1 Evaluation metrics

We use recall and precision rate to evaluate the effectiveness of our optimization approach. The recall and precision are defined as

- precision = A/(A+B)*100%
- recall = A/(A+C)*100%

Where A stands for the number of relevant objects, B stands for the number of irrelevant objects, C stands for the number of missing objects, A+C stands for the total number of relevant objects, and A+B stands for the total number of extracted objects.

5.2 Results analysis

Table 3 shows our 2P-RULE extraction results on the pages of websites collected in Table 2. Table 4 shows the extraction results of typical system Lixto. For the third website in Table 2, there is one webpage having 284 complex objects. Experiment results show that system extracts 285 complex objects totally with only one irrelevant object, so the precision is 99% (284/285) and the recall is 100% (no missing objects). The precision does not reach 100%, because user does not provide definite semantic information during the first learning. One object is extracted regarded as invalid by user. For the website

Name	URL	Webpage	Number of pages
Amazon	http://www.amazon.com	Top Sellers(TVs)	12
VLDB	http://www.acm.org/sigmod/dblp/db/conf/vldb	VLDB 1989	20
Web Robot	http://www.robotstxt.org/wc/active/html	Web Robot	1

Table 2: Test websites

Name	Wrapable?	Learning times	Precision	Recall	Test pages
Amazon	Yes	1	100%	98.3%	12
		2	100%	100%	12
Web Robot	Yes	1	99%	100%	1
		2	100%	100%	1
VLDB	Yes	1	100%	100%	20

Table 3: Experiment results of 2P-RULE

Name	Wrapable?	Learning times	Precision	Recall	Test pages
Amazon	Yes	1	95%	90%	12
		2	98%	100%	12
Web Robot	Yes	1	90%	96%	1
		2	95%	98%	1
VLDB	Yes	1	80%	90%	20

Table 4: Experiment results of Lixto

VLDB, our system works well. Its web pages contain complex semantic schema structure and the set objects do not have corresponding nodes in DOM tree. But after learning once, our system still has 100% precision when extracting 20 pages. For the Amazon website, there is an average recall of 98.3% on 12 web pages. The missing objects are due to the design of web page. We find that the text feature selected by user does not appear in some pages containing relevant objects and these objects are missed when applying extraction rules into their host pages. After learning once again, system automatically adds new text feature into extraction rules. So the missed objects are back and the recall becomes 100%. On this website Lixto achieves precision of 95% after learning once, and 100% after learning three times. Obviously, our system has better performance in recall, precision and learning times.

6 Conclusion and future work

In this paper, we propose a novel two-phase rule generation and optimization (2P-RULE) approach. 2P-RULE consists of internal rule optimization (IRO) process and external rule optimization (ERO) process. In IRO, based on the mapping created by a user, system automatically generates an optimized rule list for the schema. Whereas in ERO, the user can create multiple mappings to generate further rule lists. All the acquired rule lists are merged and refined into one optimized rule list, which is expressed with XQuery as the final extraction rules. Experiments show that our 2P-RULE approach is suitable for extracting information from web pages with complex nested structure, and can also achieve better precision and recall ratio. Our future work includes the automatic verification of extraction rules, the efficient organization, storage and management of obtained extraction rules.

References

Lawrence S., Giles L.(1999):Accessibility and distribution of information on the Web. Nature, 1999, 400(8): 107-109.

Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva, Juliana S. Teixeira. (2002): A Brief Survey of Web Data Extraction Tools. SIGMOD Record, 2002, 31(2): 84 - 93.

Soderland S. (1999): Learning Information Extraction Rules for Semi-structured and Free Text. Machine Learning, 1999, 34(1-3):233-272.

Liu L., Pu C., Han W. (2000): XWRAP: An XML-enabled Wrapper Construction System for Web Information Sources. In Proc. of the 16th ICDE Conf., San Diego, California, USA, 2000.

Han W., Buttler D., Pu C. (2001): Wrapping Web Data into XML. SIGMOD Record, 2001, 30(3):33-39.

Arnaud S. and Fabien A. (1999): Building Light-Weight Wrappers for Legacy Web Data-Sources Using W4F.In Proceedings of 25th VLDB Conference, Edinburgh, Scotland, UK, 1999.

Sahuguet A. and Azavant F. (1999): Web Ecology: Recycling HTML Pages as XML Documents Using W4F. In Second International. Workshop on the Web and Databases, Philadelphia, Pennsylvania, USA, 1999.

Baumgartner R., Flesca S., Gottlob G. (2001): Visual Web Information Extraction with Lixto. Proceedings of 27th International Conference on Very Large Database. Roma, Italy, 2001.

Baumgartner R., Ceresna M., Gottlob G., Herzog M., Zigo V. (2003): Web Information Acquisition with Lixto Suite. In Proceedings of the 19th ICDE Conference, Bangalore, India, 2003.

Meng X., Wang H., Hu D., Chen L. (2003): A Supervised Visual Wrapper Generator for Web-Data Extraction. COMPSAC 2003: 657-662

Arasu A., Garcia-Molina, H. (2003): Extracting structure data from web pages. In: Proceedings of SIGMOD. (2003) 337–348.

Hu D., Meng X.(2005): Automatically extracting data from data-rich web pages. DASFAA 2005, Beijing, 2005,4

Ma L., Shepherd J. (2004): Information Extraction via Automatic Pattern Discovery in Identified Region. DEXA 2004: 232-242

DOM.http://www.w3c.org/TR/REC-DOM-Level-1.Xpath

XML Path Language Version 2.0. http://www.w3.org/TR/xpath20

XQuery. http://www.w3.org/TR/xquery

HTML Tidy Library Project. http://tidy.sourceforge.net/

A Reconstruction-based Algorithm for Classification Rules Hiding

Juggapong Natwichai **Xue Li** **Maria E. Orlowska**

School of Information Technology and Electrical Engineering
The University of Queensland, Brisbane, Australia
Email: {jpn, xueli, maria}@itee.uq.edu.au

Abstract

Data sharing between two organizations is common in many application areas e.g. business planing or marketing. Useful global patterns can be discovered from the integrated dataset. However, some sensitive patterns that should have been kept private could also be discovered. In general, disclosure of sensitive patterns could decrease the competitive ability of the data owner. Therefore, sensitive patterns should be hidden before data sharing starts. To address this problem, released datasets must be modified unavoidably. However, if the overall characteristics of the dataset can be maintained, the dataset is still usable perfectly. Therefore, not only the privacy should be concerned, but also the usability. In this paper, we propose a new algorithm to preserve the privacy of the classification rules by using reconstruction technique for categorical datasets. Firstly, all discovered classification rules in the released dataset are presented to the data owner to identify sensitive rules that should be hidden. Subsequently, remained non-sensitive rules along with extracted characteristics information of the dataset are used to build a decision tree. Finally, the new dataset which contains only non-sensitive classification rules is reconstructed from the tree. From empirical studies, our algorithm can preserve the privacy effectively. Additionally, the usability of the datasets can also be preserved.

1 Introduction

The advancement in information technology leads to effective and efficient ways to collect, store or analyze data. To analyze data, besides statistical methods to determine important values from stored data, data mining techniques are always used to extract useful patterns in databases. Many data mining techniques can be applied to obtain the patterns e.g. classification mining, association rules mining or clustering.

Currently, many efficient data mining algorithms have been proposed. On one hand, with these algorithms, data owners can use them to extract useful patterns from collected data. On the other hand, the algorithms can become a threat to privacy. They can be used in combination with other techniques to disclose sensitive private data. For example, mining on the medical dataset can help re-identifying of individual person, although it seems to be anonymous.

Besides the data privacy, another problem is sensitive private patterns. Sharing of the dataset is common in many applications. Collaborating companies could share their dataset for global patterns discovering. For this purpose, the datasets could be released from an organization to the other organizations. However, there can be some sensitive patterns within shared datasets that should be kept private. Because, in general, disclosure of sensitive patterns could decrease the competitive ability of the data owner.

In this paper, we focus on the problem of classification rules privacy preservation or classification rules hiding. The motivating example was discussed in (Natwichai, Li & Orlowska 2005) as following: a credit card approval dataset is released by a credit card company for a new home loan company. Each record in the dataset is individual applicant. The collected attributes of each applicant in this dataset can be financial status, number of working years at the current company, gender, salary level, living area and range of age. While the class is the approval result. With the released dataset the home loan company is able to build an initial classification model to classify their home loan applicants. In this case, the dataset must be provided because two companies have different views on each attribute. However, some sensitive patterns can be discovered from the given dataset. More specifically, the patterns that are able to give competitive ability to the others more than the data owner's expectation can be discovered. For example, it can be used to identify appropriate groups of customers, or even individual person to send advertising mails. It can be done simply e.g. by changing the class label to be the post code of living area. Although it might not be able to do accurately, only narrowing down the scope can be considered as a threat. Therefore, the modified dataset should be given to the home loan company instead of the original one by some agreement between these two companies.

As discussed above, the released datasets should be modified in order to preserve the privacy of sensitive patterns. Obviously, the correctness of a modified dataset is decreased after modification process. Despite that fact, patterns are not generally discovered from the individual record in the dataset, but overall characteristics. If the characteristics can still be maintained, the dataset is usable practically. Therefore, the privacy preservation process should also preserve the overall characteristics, or usability of the dataset.

Many works have been proposed for sensitive association rules hiding (Atallah, Elmagarmid, Ibrahim, Bertino & Verykios 1999, Verykios, Elmagarmid, Bertino, Saygin & Dasseni 2004, Oliveira & Zaiane 2003). Heuristic modification approaches are applied to this problem. It can be done by modifying some transactions such that the support and/or con-

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

fident values of sensitive rules are fall below the specified thresholds. The modification should be done by avoiding side effect as much as possible. The side effect listed in their works are false drop rules and ghost rules. False drop rules are non-sensitive rules contained in the original dataset, but lost in the modified dataset. While ghost rules are non-sensitive rules contained in the modified dataset, but not in the original dataset. However, classification rules hiding is different problem. Unlikely to association rules that present the association between items in transactions, classification rules are built from set of attribute values which can classify records into distinct predefined classes. In classification mining, distribution of classes with regards to attributes are important. Moreover, most classification algorithms make an assumption that there is no dependence between attributes. Therefore, the heuristic approach that suites for association rules hiding may be an inappropriate approach for classification rules hiding. As we will demonstrate it later.

In (Natwichai et al. 2005), the classification rules hiding framework was proposed for categorical datasets by using reconstruction technique. Instead of arbitrary dataset modification, the framework reconstructs a new dataset that only non-sensitive rules can be discovered from it. Additionally, the usability of the new dataset is also addressed. The method is proceeded as following: a rule-based classification algorithm is used on the given dataset to obtain all classification rules. Subsequently, only non-sensitive classification rules which are approved by the data owner are used to build a data generator, a decision tree. Finally, a new dataset which contains only non-sensitive classification rules is reconstructed from the decision tree. Even the difference in representation between the reconstructed and original datasets can be found, but this approach can maintain high usability.

In this paper, we propose a new algorithm to hide sensitive classification rules for categorical datasets. Our algorithm is based on the above method. However, we propose to use more extracted characteristics information from the dataset with regard to classification issue to improve decision tree building, and make the reconstruction process better. The experimental results show that our proposed algorithm can improve the usability and still maintain the privacy of the input datasets effectively.

The rest of this paper is organized as following. Section 2 provides a review of related work. The basic concepts and a problem statement are presented in Section 3. Subsequently, our proposed approach is shown in Section 4. The experimental results on real and synthetic datasets are brought up in Section 5. Finally, Section 6 gives the conclusion.

2 Related Work

Generally, individual privacy preservation problem can be addressed by access control method that have been proposed substantially among database research community (Jajodia, Samarati, Subrahmanian & Bertino 1997, Benferhat, Baida & Cuppens 2003). While, view mechanism can be also a simple, however, effective approach for this problem. Apart from above approaches, statistical security-control can also be used (Domingo-Ferrer & Torra 2004). In this approach, perturbed datasets are used instead of original datasets. While some statistical values are still preserved as the same as the original dataset.

In the other way, generalization and/or suppression techniques can be used to protect the privacy of individual. Usually, the released dataset will be generalized or suppressed until the privacy constraint is reached. One privacy constraint is k-Anonymity (Sweeney 2002*b*). A k-Anonymized dataset is the dataset that every record can not be distinguished from k-1 other records. There are many works addressed individual privacy problem by using this constraint (Sweeney 2002*a*, Meyerson & Williams 2004, Jr. & Agrawal 2005).

However, not only the privacy of the individual should be considered, but also the patterns with regard to data mining algorithms. Obviously, the significant of this problem becomes the important issue because of the progress in data mining techniques (Clifton & Marks 1996, Estivill-Castro, Brankovic & Dowe 1999, Clifton & Estivill-Castro 2002, Thuraisingham 2002). With regard to data mining algorithm, there are many available approaches to address privacy problems. They can be categorized into a few different groups based on their approach such as cryptography-based, reconstruction-based and heuristic-based techniques (Verykios, Bertino, Fovino, Provenza, Saygin & Theodoridis 2004). There are many proposed work for association rules privacy preserving. Heuristic approaches are used to tackle the problem (Verykios, Elmagarmid, Bertino, Saygin & Dasseni 2004, Oliveira & Zaiane 2003). In this approach, selected values in the dataset are changed to decrease support and/or confident values of sensitive rules. The rules will be hidden successfully if their support and/or confident values are less than specified thresholds. Instead of values changing, there is also another heuristic way to replace selected values with unknown values (Saygin, Verykios & Clifton 2001). In generally, this approach preserves the privacy by introducing margins of support and confident values to some extent of uncertainty.

In classification mining context, most of the research works focus on preserving privacy of individuality (Agrawal & Srikant 2000). In (Chang & Moskowitz 1998), a privacy preserving method of some data for data downgrading scenarios is proposed. In the method, a classification result of unprotected data is analyzed to consider its impact on protected data. While in the viewpoint of cryptography, the problem of computing a global decision tree from multi-party datasets by avoiding a party to know any other's data is addressed in (Lindell & Pinkas 2000).

For reconstruction-based approach, there are many works have been done to preserve the privacy of association rules (Rizvi & Haritsa 2002, Chen, Orlowska & Li 2004). Generally, this approach, firstly, extracts selected characteristics of the datasets. After any privacy preserving process is done, a new dataset will be reconstructed. The approach of dataset reconstruction has advantage over the heuristic data modification approaches since it hardly introduces side-effect (Verykios, Bertino, Fovino, Provenza, Saygin & Theodoridis 2004).

In privacy preservation context, there are many proposed usability metrics. Generally, they can be categorized into two main groups: general metrics and data mining task-specified metrics. For general usability metrics, the usability can be measured by how much the dataset has been modified. For example, in (Sweeney 2002*a*), generalization process is used to preserve the individual privacy, the more generalization means the less of usability. While the data mining task-specified metrics take proposed data mining task into measurement, the more inaccurate mining result means the less of usability. In general, it is really difficult to measure the usability if there is no specified-target mining task. Because there are various type of data mining algorithms, manipulation of dataset only a little could impact some algorithms very much. Therefore, the data owner and recipient should make the agreement about what to be mined before the privacy preservation process begins, so the released dataset will be able to be used fully.

3 Basic Concepts

3.1 Datasets

Let a dataset D be a 4-tuple $\{U, A, V, f\}$ where

U is a nonempty finite set of records,

A is a nonempty finite set of attributes such that for any $a \in A$, we can associate set of values V_a that is domain of attribute a,

V is a nonempty finite set of values of all attributes such that $V_a \subseteq V$ and $\bigcup_{V_a} = V$,

and, f is function such that $f : U \times A \rightarrow V_a$, the value of an attribute for a record.

3.2 Classification Rules

Classification problem over D is a tuple $z = (mp, f_1, \ldots, f_n)$ where $mp : V^n \rightarrow C$, in which C is a nonempty finite set of class labels. Integer n is the number of attributes to be used for classification. So, problem z can be defined as a problem of searching for value $z(u, a) = mp(f_1(u, a), \ldots, f_n(u, a))$ for an $a \in A$.

Let a set of classification rules R over D be expressed as:

$$f_1(a_1) \wedge f_2(a_2) \wedge f_3(a_3) \wedge \ldots \wedge f_m(a_m) \rightarrow c,$$

where $f_1 \ldots, f_m \in F$, $a_1, \ldots, a_m \in A$ and $c \in C$. Integer m can be called *length* of a classification rule R, or $|R| = m$.

3.3 Problem statement

Given a dataset D, a set of classes C, a set of classification rules R over D, and also $R' \subset R$, R' is a set of sensitive rules, find a dataset D' such that there exists only a set of rules $R - R'$ can be derived.

4 Classification Rules Hiding Algorithms

In this section, we firstly review the heuristic modification method to be used for comparison. Subsequently, the framework for preserving the privacy of classification rules by using reconstruction technique is presented. Finally, our new algorithm is introduced.

4.1 Heuristic modification method

For the sake of clarity, we demonstrate the heuristic modification approach by example. A sample dataset for credit card approval in Table 1 will be used through this section. Every record represents a single person who applied for credit card. The categorical dataset consists of four attributes : “# years” the number of years at current work, “marriage status” and ”gender” , the marriage status and gender of applicants, “listed” an attribute of whether the applicant is on a “black list”. Finally, each class label is an approval result. For rule hiding demonstration, firstly, we use a classification algorithm (C4.5 (Quinlan 1993)) to obtain a whole set of classification rules. The set of rules is shown in Table 2.

Suppose that the owner of the dataset wants to hide the rule: **“# years = medium $\wedge$ listed = yes → class = NO”**. The easiest heuristic method (in terms of association rule mining) is to decrease the confidence of the rule. This can be done by alternating values in the right hand side, the class, of the corresponding records. In classification context, it is decreasing of ability to classify datasets. In this case, corresponding records are record number 6 and 14. Suppose that the record No. 6 is chosen. Subsequently, its class label is changed to **YES** as shown in Table 3. For checking whether the hiding successes, the dataset has to be classified again. The set of rules on modified dataset is listed in Table 4.

Table 1: Original credit card approval dataset

No.	# years	marriage status	gender	listed	class
1	short	married	female	no	NO
2	short	married	female	yes	NO
3	long	married	female	no	YES
4	medium	divorce	female	no	YES
5	medium	single	male	no	YES
6	medium	single	male	yes	NO
7	long	single	male	yes	YES
8	short	divorce	female	no	NO
9	short	single	male	no	YES
10	medium	divorce	male	no	YES
11	short	divorce	male	yes	YES
12	long	divorce	female	yes	YES
13	long	married	male	no	YES
14	medium	divorce	female	yes	NO

Table 2: Original credit card classification rules

Antecedence	Class
# years = short ∧ gender = female	NO
# years = short ∧ gender = male	YES
# years = long	YES
# years = medium ∧ black list = yes	YES
# years = medium ∧ black list = no	NO

Table 3: Modified credit card approval dataset (The record No. 6 has been modified)

No.	# years	marriage status	gender	listed	class
..	...	...	...	...	...
5	medium	single	male	no	YES
6	medium	single	male	yes	**YES**
7	long	single	male	yes	YES
..	...	...	...	...	...

From the result, it seems that the sensitive rule has been hidden successfully. However, there are some differences between the original and the modified set of rules. Some non-sensitive rules are lost e.g. the second, the third and the fifth rules of original dataset. Moreover, some insignificant patterns also become significant. For example, in the original set of rules, there is no rule likes the second, the third and the fourth rules in the new set of rules.

However, if the selected rules has much more ability to classify records, the problem becomes more complicated. Suppose that, the rule: **“# years at current work = long → class = YES”** is selected. The number of classified records in the original dataset of this rule is 4, while the previous rule has only 2. Obviously, more corresponding records, the record No. 3, 7, 12 and 13 need to be considered.

If the heuristic approach is used, we may start from modifying only one record. Suppose that the record No. 3 is chosen as in Table 5, and its class label is changed from **YES** to **NO**.

Table 4: Modified credit card classification rules (The record No. 6 has been modified)

Antecedence	Class
# years = short ∧ gender = female	NO
# years = long ∧ gender = female	YES
# years = medium ∧ gender = female	YES
gender = male	YES

Table 5: Modified credit card approval dataset (The record No. 3 has been modified)

No.	# years	marriage status	gender	listed	class
1	short	married	female	yes	NO
2	short	married	female	no	NO
3	long	married	female	yes	**NO**
4	medium	divorce	female	yes	YES
..	...	...	...	...	...

Table 6 shows the result after we subsequently classified the modified dataset. The substantial difference between results can be seen. On the other hand, if the record No. 7 is chosen as shown in Table 7, the result shown in Table 8 is also different from the original one.

As it can be expected, the set of rules is substantially different from the original, even only one record is changed. Obviously, rules hiding by arbitrary modification causes too much side effect.

4.2 Reconstruction-based Algorithms

As we have discussed, the reconstruction-based approach can provide output datasets with less side effect. Rather than dataset modification to obtain target patterns, the approach focuses on patterns and dataset characteristics controlling to obtain target dataset. In this way, the usability of the datasets will also be preserved, although the presentation of the datasets can be different.

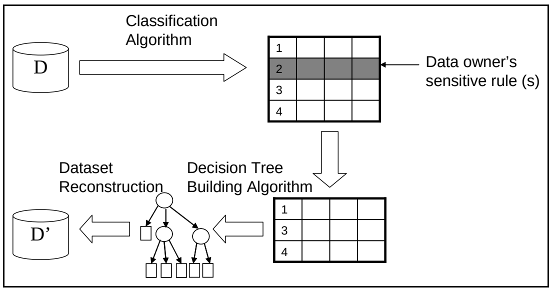

Classification Algorithm
D
1
2
3
4
Data owner's sensitive rule (s)
Dataset Reconstruction
Decision Tree Building Algorithm
D'
1
3
4

Figure 1: The Reconstruction-based Framework.

Table 6: Modified credit card classification rules (The record No. 3 has been modified)

Antecedence	Class
gender = female	NO
gender = male	YES

Table 7: Modified credit card approval dataset (The record No. 7 has been modified)

No.	# years	marriage status	gender	listed	class
..	...	...	...	...	...
6	medium	single	male	no	NO
7	long	single	male	no	**NO**
8	short	divorce	female	yes	NO
...	...	...	...	...	...

The reconstruction framework is shown in Figure 1. It starts with classifying original dataset by rule-based classification algorithms e.g. RIPPER (Cohen 1995) or C4.5 (Quinlan 1993). After a set of classification rules is extracted, the data owner can identify the sensitive rules. The remaining non-sensitive rules are considered as characteristics of the dataset. Therefore, they are used to build a dataset generator, a decision tree, by decision tree building algorithm. Because the sensitive rules are excluded in the algorithm, there is no such directly derivable rules in the reconstructed dataset. Obviously, a number of used rules effects amount of the characteristics to be preserved. Therefore, unpruned classification rules, less significant rules, are used in the algorithm. Finally, the decision tree is used to generate a new dataset with the same amount of records as the original dataset. For simplicity, each record is built and assigned each attribute value with uniform probability. Then, it is induced through the respected path in the decision tree. Finally, a class label is assigned to the record as the terminal node of the tree.

Using uniform probability data generator provides advantages to the approach. Obviously, the number of reconstructed records in each path of the trees can be estimated. For example, if a binary attribute **"gender"** is chosen as the root of a tree, approximated half of reconstructed records will have **"male"** attribute values, while another half will have **"female"**.

The decision tree building algorithm is shown in Table 9. In the algorithm, each non-sensitive rule is put in a decision tree one by one. The ordering of rules selection is based on their ability to classify original dataset. The ability to classify records of each rule, denoted as $cab(r)$, can be determined by the number of records classified by rule r in the original dataset. When any rule is put earlier, it will be in the higher level of the tree. With a uniform probability characteristic of the dataset generator, a rule that appears in the higher level will be used to generate more records. This can help maintaining similarity between original and reconstructed datasets.

In the algorithm, a selected rule is allowed to be reflected by many paths of the tree. In this way, the similarity between the reconstructed and the original dataset can be maintain by a number of its paths and levels in the tree. The function $approx(r)$ is used to determine a number of approximated records generate from the rule r. As seen in the algorithm, paths from any rule will be generated repeatedly until the number of approximated records gets close to the number of actual records classified by the rule in the original dataset.

4.3 Usability Improvement

In our proposed algorithm, not only the extracted rules are considered as the characteristics of datasets, but the gain ratio of each attribute. While a rule is induced in the tree, an attribute is selected as a node in the tree. We select the order of attributes in each rule by gain ratio, instead of least common attribute in the previous work. Therefore, the higher gain attribute is put in the higher level of the tree. This means the high gain ratio attribute in the original dataset will also has high gain ratio in the reconstructed dataset. With aiding of information gain, the usability of released datasets can be improved. In the algorithm, gain ratio for attribute a is denoted as $gain(a)$.

With regards to computational complexity, the

Table 8: Modified credit card classification rules (The record No. 7 has been modified)

Antecedence	Class
black list = yes	NO
black list = no	YES

Table 9: A new decision tree building algorithm

Inputs: R is set of classification rules.
R' is set of sensitive rules.
Outputs: DT is a decision tree.

```
While $|R| > 0$ do
    select $r_i$ to be induced such that $r_i \in R - R'$,
    and $\forall r_k \in R - R', cab(r_i) \geq cab(r_k)$.
    While $approx(r_i) \leq cab(r_i)$
        While $r_i$ is not induced completely do
            select $a_j$ in $r_i$ such that
            $\forall a_l$ in $r_i$, $gain(a_j) \geq gain(a_l)$,
            put $a_j$ as non-terminal node of $DT$.
        End while.
        Assign a class for $r_i$.
    End while.
End while.
```

time complexity of this algorithm is $O(mn)$, where m is the number of non-sensitive rules and n is the number of attributes of a given dataset.

4.4 A Decision Tree Building Example

For the sake of clarity, we demonstrate how the algorithm proceeds by example. Suppose that, a given dataset has six binary attributes: A, B, C, D, E, F and a class label (+,-). The total number of records in the dataset is 800. Its gain ratio information is shown in Table 11. After classification, a set of rules in Table 10 is obtained. Suppose that the owner selects the rule **"A=$\mathbf{a_1}\wedge$ F=$\mathbf{f_0}$ $\rightarrow$ +"** to be hidden.

The first classification rule to be induced is **"A=$\mathbf{a_1}\wedge$ B=$\mathbf{b_2}$ $\rightarrow$ +"** since it can classify the highest number of records. The attribute B is selected before attribute A because its gain ratio is higher. Subsequently, class + is assigned as shown in Figure 2. The approximated number of reconstructed records for this rule is 200.

"A=$\mathbf{a_1}\wedge$ C=$\mathbf{c_1}$ $\wedge$ D=$\mathbf{d_2}$ $\rightarrow$ +" is the second rule to be induced. The rule is placed at the left half of the tree. Attributes are selected ordered by their gain ratios, A, C and D respectively. The approximated number of records for the rule is 20. Figure 3 shows the result of the second rule induction.

In Figure 4, the rule **"B=$\mathbf{b_1}\wedge$ E=$\mathbf{e_2}$ $\rightarrow$ +"** is induced. Noticeably, there are many options to induce it. Firstly, it can be induced next to B=$b_1\wedge$ A=a_2. The second option is at B=$b_1\wedge$ A=$a_1\wedge$ C=c_2. Thirdly, it can also be at B=$b_1\wedge$ A=$a_1\wedge$ C=$c_1\wedge$ D=d_1 While, many combinations of them can be also selected. As shown in Figure 4, the algorithm induces

Table 10: Example set of rules

Antecedence	Class	# Classified Records
A=$a_1\wedge$ B=b_2	+	143
A=$a_1\wedge$ C=c_1 $\wedge$ D=d_2	+	47
B=$b_1\wedge$ E=e_2	+	21
E=$\mathbf{e_1}\wedge$ F=$\mathbf{f_1}$	+	**82**
default	–	407

Table 11: Example gain ratios of attributes

Attribute	Gain Ratio
A	0.150
B	0.153
C	0.080
D	0.070
E	0.098
F	0.072

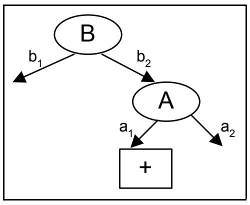


Figure 2: Decision tree generation example (the first rule is induced)

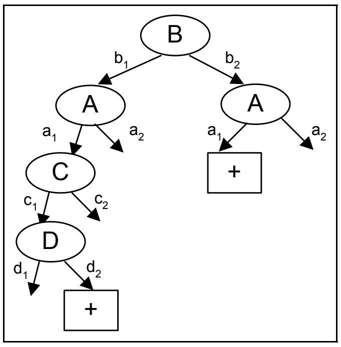


Figure 3: Decision tree generation example (the second rule is induced)

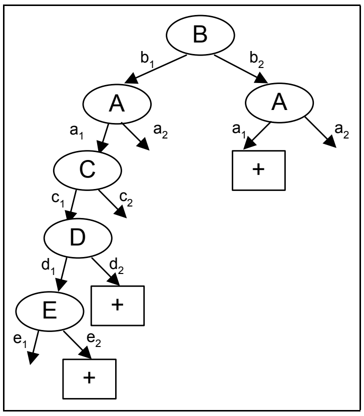


Figure 4: Decision tree generation example (the third rule is induced)

the rule by the third option. Because it makes the number of approximate reconstructed records close to number of classified records in the original dataset by this rule (25 and 21). Finally, the completed decision tree is shown in Figure 5.

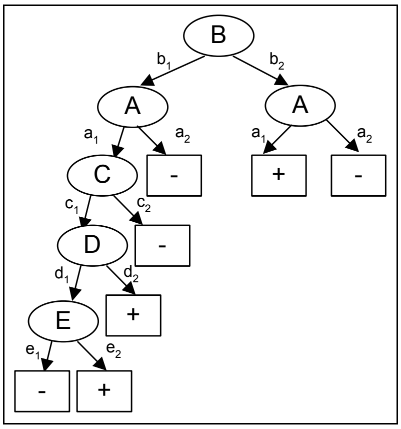


Figure 5: Final decision tree generation example

5 Experiments and results

Three real-life datasets, Credit Screening, Voting Records and Mushrooms from UCI Repository (Blake & Merz 1998) were used in our experiments. For the first dataset, the continuous attributes were transformed to categorical attributes. The detail of each dataset is shown in Table 12

The experiments consist of five parts. Firstly, a sensitive rule is randomly selected to be hidden. The issue about privacy and side effect of the reconstructed datasets are considered. The second experiment is similar to the first one, but multi-rule hiding. The third experiment considers usability issue. In this experiment, the usability of the datasets with respect to the number of non-sensitive rules that involve in the framework is considered. Another usability of the datasets with regard to the ability to classify records of each single rule is considered in the forth experiment. The last experiment shows the usability of the reconstructed datasets respect to the number of remained non-sensitive rules when multi-rule is hidden.

The procedure of the experiments is shown in Figure 6. In the experiments, two classification algorithms: RIPPER and C4.5 Rule were selected. For each experiment, two classification algorithms are used. After a set of classification rules is generated by the first algorithm, a random rule is selected as the sensitive rule. The set of remaining non-sensitive rules are used to build a decision tree by our algorithm. Subsequently, the tree is used to generate a new non-sensitive reconstructed dataset. Finally, the second classification algorithm is used to evaluate the reconstructed dataset. In the experiments, the first and second classification algorithms can be both the same or different algorithm.

Table 12: Experimental Datasets detail

Detail	Dataset		
	Credit Screening	Voting Records	Mushroom
#Records	690	435	8124
#Attributes	15	16	22
#RIPPER Rules	5	4	9
#RIPPER Unpruned Rules	26	9	12
#C4.5 Rules	5	4	7
#C4.5 Unpruned Rules	21	9	19

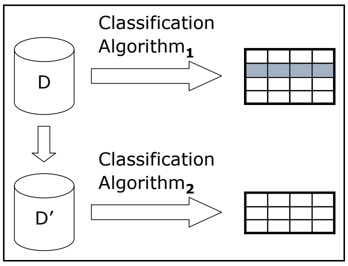


Figure 6: Experiment Procedure

Another experiment is multi-rule hiding. The algorithm will be checked whether it can perform hiding more than one sensitive rule. The same procedure as single rule hiding is used. In this experiment, all unpruned classification rules are used to build the decision tree.

5.1 Evaluation Metrics

There are three metrics for evaluation. Firstly, the privacy issue must be considered. More specifically, the existing of sensitive rules is considered from the entire set of rules discovered by the second algorithm. Secondly, other than the sensitive rules that should be hidden, the side effect from the hiding algorithm is also considered. There are two main metrics to evaluate the side effect: a number of ghost rules and false-drop rules. Both numbers can also be seen when the second classification algorithm is used. Obviously, side effect should be kept minimal.

The last metric is the usability of the reconstructed dataset. Because the released dataset are usually intended to build the classification model. Therefore, the ability to classify datasets of each attribute is measured as the usability metric. In the experiments, the gain ratio (Quinlan 1993) is used to served our propose. We measure the percentages of gain ratio variations between the original and reconstructed dataset with Equation 1.

$$V = \sqrt{\frac{\sum_{i=1}^{n} (\frac{o_i - r_i}{o_i})^2}{n}} \times 100 \qquad (1)$$

where o_i and r_i are gain ratios for the ith attribute on the original and reconstructed datasets. While n is the number of entire attributes.

In order to evaluate the usability improvement of our decision tree building algorithm, the algorithm in (Natwichai et al. 2005) was used to compare. In the experiment, the our algorithm is called ”GAIN” algorithm, While the compared algorithm is called ”NOGAIN”.

5.2 Experimental results and discussions

Table 13: Experimental result when RIPPER is used as both the 1st and the 2nd classification algorithm

Dataset	Used rules	# Discovered sensitive rules	False drop rules	Ghost rules
Credit Screening	4	0	0	0
	5	0	0	0
	7	0	0	0
	10	0	0	0
	15	0	0	0
	20	0	0	0
	25	4	0	0
Voting Records	3	0	0	0
	4	0	0	0
	6	0	0	0
	8	0	0	0
Mushroom	8	0	0	0
	9	0	0	0
	10	0	0	0
	11	0	0	0

Table 14: Experimental result when C4.5 Rule is used as both the 1st and the 2nd classification algorithm

Dataset	Used rules	# Discovered sensitive rules	False drop rules	Ghost rules
Credit Screening	4	0	0	0
	5	0	0	0
	7	0	0	0
	10	0	0	0
	15	0	0	0
	20	0	0	0
Voting Records	3	0	0	0
	4	0	0	0
	6	0	0	0
	8	0	0	0
Mushroom	6	0	0	0
	7	0	0	0
	9	0	0	0
	12	0	0	0
	15	0	0	0
	18	0	0	0

Table 15: Experimental result when C4.5 Rule is used as the 1st while RIPPER is used as the 2nd classification algorithm

Dataset	Used rules	# Discovered sensitive rules	False drop rules	Ghost rules
Credit Screening	4	0	1	0
	5	0	0	0
	7	0	0	0
	10	0	0	0
	15	0	0	0
	20	0	0	0
Voting Records	3	0	1	0
	4	0	1	0
	6	0	0	0
	8	0	0	0
Mushroom	6	0	1	0
	7	0	1	0
	9	0	0	0
	12	0	0	0
	15	0	0	0
	18	0	0	0

Table 16: Experimental result when RIPPER is used as the 1st while C4.5 Rule is used as the 2nd classification algorithm

Dataset	Used rules	# Discovered sensitive rules	False drop rules	Ghost rules
Credit Screening	4	0	0	1
	5	0	0	1
	7	0	0	0
	10	0	0	0
	15	0	0	0
	20	0	0	0
	25	4	0	0
Voting Records	3	0	0	1
	4	0	0	1
	6	0	0	1
	8	0	0	0
Mushroom	8	0	1	1
	9	0	0	1
	10	0	0	0
	11	0	0	0

It can be seen from Table 13 and 14 that our decision tree building algorithm can be used to reconstruct the datasets without any side effect, while it is also safe for sensitive rule. Even the different classification algorithm were used, there were only small amount of side effects as shown in Table 15 and 16. Additionally, when all unpruned rules were used, there was no any side effect at all.

Table 17, 18, 19, and 20 show the experimental results when multi-sensitive rules were selected to be hidden. In this experiment, all unpruned rules were used in the reconstruction process. From this experiment, no sensitive rule is discovered, even different classification algorithms were used. However, one ghost rule was found in Mushroom dataset when RIPPER was used as the first algorithm and the second algorithm was C4.5 Rule as shown in Table 19. This can be explain that the Mushroom dataset has small difference between number of pruned rules (9) and unpruned rules (12) when it is classified by RIPPER. Therefore, the number of remained non-sensitive rules as the dataset characteristic was not enough to reconstruct the new dataset. By the way, this case can be considered as the extreme case because, in this experiment, 7 classification rules were hidden from all available 9 rules.

The Figure 7 shows the usability on reconstructed datasets respect to the number of participated non-sensitive rules when a single rule is selected to be hidden. In this experiment, C4.5 Rule was used as both the first and the second classification algorithms. Obviously, our new proposed decision tree building

Table 17: Experimental result of multi-rule hiding when RIPPER is used as both the 1^{st} and the 2^{nd} classification algorithm

Dataset	Hidden rule	Discovered sensitive rules	False drop rules	Ghost rules
Credit Screening	1	0	0	0
	2	0	0	0
	3	0	0	0
Voting Records	1	0	0	0
	2	0	0	0
Mushroom	1	0	0	0
	2	0	0	0
	4	0	0	0
	7	0	0	0

Table 18: Experimental result of multi-rule hiding when C4.5 Rule is used as both the 1^{st} and the 2^{nd} classification algorithm

Dataset	Hidden rule	Discovered sensitive rules	False drop rules	Ghost rules
Credit Screening	1	0	0	0
	2	0	0	0
	3	0	0	0
Voting Records	1	0	0	0
	2	0	0	0
Mushroom	1	0	0	0
	2	0	0	0
	3	0	0	0
	5	0	0	0

Table 19: Experimental result of multi-rule hiding when RIPPER is used as the 1^{st} while C4.5 Rule is used as the 2^{nd} classification algorithm

Dataset	Hidden rule	Discovered sensitive rules	False drop rules	Ghost rules
Credit Screening	1	0	0	0
	2	0	0	0
	3	0	0	0
Voting Records	1	0	0	0
	2	0	0	0
Mushroom	1	0	0	0
	2	0	0	0
	4	0	0	0
	7	0	0	1

Table 20: Experimental result of multi-rule hiding when C4.5 Rule is used as the 1^{st} while RIPPER is used as the 2^{nd} classification algorithm

Dataset	Hidden rule	Discovered sensitive rules	False drop rules	Ghost rules
Credit Screening	1	0	0	0
	2	0	0	0
	3	0	0	0
Voting Records	1	0	0	0
	2	0	0	0
Mushroom	1	0	0	0
	2	0	0	0
	3	0	0	0
	5	0	0	0

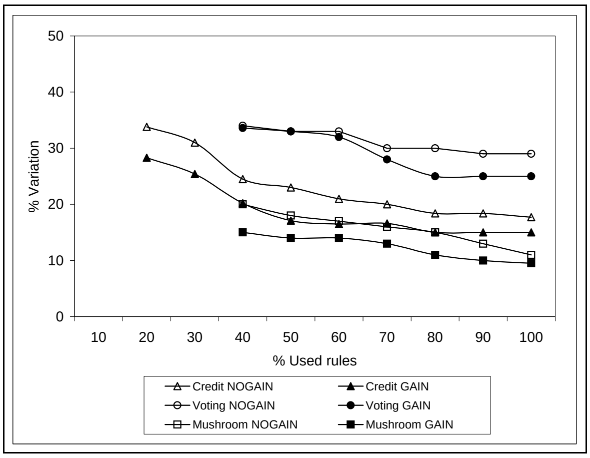


Figure 7: The usability with regard to the percentage of used rules.

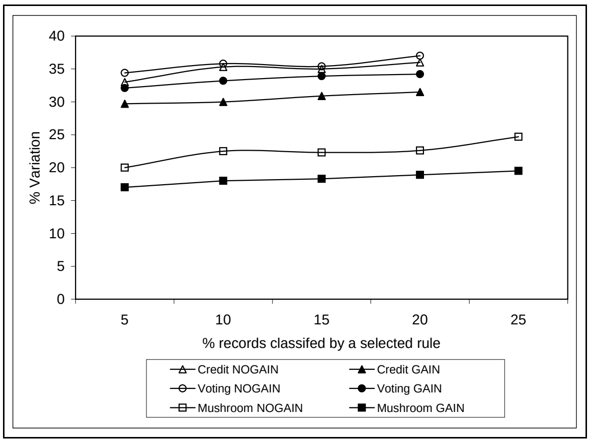


Figure 8: The usability with regard to classifying ability of rules.

algorithm, GAIN, performed better than NOGAIN algorithm. It means that GAIN algorithm can benefit from aiding of gain ratio information of the original datasets.

In Figure 8, the usability of reconstructed datasets respect to the classifying ability of sensitive rules are shown. A single rule with various classifying abilities was selected for each experiment. C4.5 Rule was used as the first and the second classification algorithms. When the rule with high ability to classify records was selected, the usability produced by NOGAIN and GAIN algorithms were different apparently. It can be seen that the NOGAIN algorithm did not seem to be consistent in this experiment. While GAIN algorithm could still maintain the usability very well.

The last experiment was usability measurement of the datasets respect to the number of sensitive rules to be hidden. Again, C4.5 Rule was used as the first and the second classification algorithms. All unpruned rules were used to build the decision tree. From Figure 9, it can be seen obviously that when the number of sensitive rules were increased, the usability of all algorithms dropped. However, our algorithm still performed very well.

6 Conclusions

In this paper, we proposed a new algorithm for privacy preserving of classification rules for categorical datasets. By using reconstruction technique, new re-

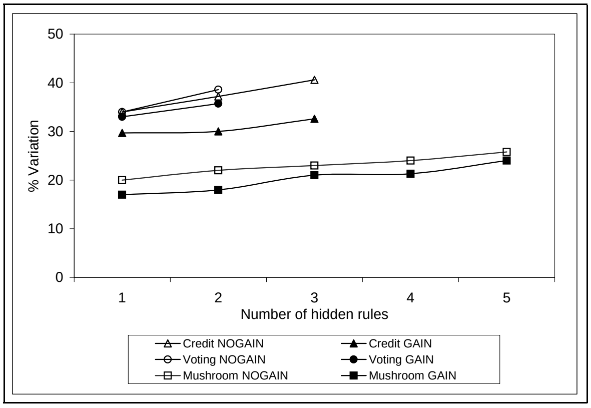


Figure 9: The usability of multi-rule hiding with regard to the number of hidden rules.

constructed datasets do not contain sensitive rules. Meanwhile, the side effect of the hiding process is kept minimal. Moreover, the higher usability of reconstructed datasets can be achieved when more characteristics of the original datasets are used. The experimental results have showed that when more rules or even the rules with high ability to classify records are hidden, our proposed algorithm can still maintain the usability satisfactorily. In the future, the other usability metrics will be studied e.g. general usability. Additionally, we will extend the algorithm to work on numerical attributes. The incremental privacy preservation processing will also be addressed.

References

Agrawal, R. & Srikant, R. (2000), Privacy-preserving data mining, *in* 'Proceedings of the 2000 ACM SIGMOD international conference on Management of data', ACM Press, pp. 439–450.

Atallah, M., Elmagarmid, A., Ibrahim, M., Bertino, E. & Verykios, V. (1999), Disclosure limitation of sensitive rules, *in* 'KDEX '99: Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange', IEEE Computer Society, Washington, DC, USA, pp. 45–52.

Benferhat, S., Baida, R. E. & Cuppens, F. (2003), A stratification-based approach for handling conflicts in access control, *in* 'SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies', ACM Press, pp. 189–195.

Blake, C. & Merz, C. (1998), 'UCI repository of machine learning databases'.
URL: *http://www.ics.uci.edu/∼mlearn/MLRepository.html*

Chang, L. & Moskowitz, I. S. (1998), Parsimonious downgrading and decision trees applied to the inference problem, *in* 'Workshop on New Security Paradigms', pp. 82–89.

Chen, X., Orlowska, M. & Li, X. (2004), A new framework of privacy preserving data sharing, *in* 'Proceedings of 4th IEEE International Workshop on Privacy and Security Aspects of Data Mining', IEEE Press, pp. 47–56.

Clifton, C. & Estivill-Castro, V., eds (2002), *IEEE ICDM Workshop on Privacy, Security and Data Mining*, Vol. 14 of *Conferences in Research and Practice in Information Technology*, ACS.

Clifton, C. & Marks, D. (1996), Security and privacy implications of data mining, *in* 'Workshop on Data Mining and Knowledge Discovery', University of British Columbia Department of Computer Science, Montreal, Canada, pp. 15–19.

Cohen, W. W. (1995), Fast effective rule induction, *in* A. Prieditis & S. Russell, eds, 'Proc. of the 12th International Conference on Machine Learning', Morgan Kaufmann, Tahoe City, CA, United States, pp. 115–123.

Domingo-Ferrer, J. & Torra, V., eds (2004), *Privacy in Statistical Databases*, Vol. 3050 of *LNCS*, Springer, Berlin Heidelberg.

Estivill-Castro, V., Brankovic, L. & Dowe, D. L. (1999), 'Privacy in data mining', *Privacy - Law and Policy Reporter* **6**(3), 33–35.

Jajodia, S., Samarati, P., Subrahmanian, V. S. & Bertino, E. (1997), A unified framework for enforcing multiple access control policies, *in* 'SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data', ACM Press, pp. 474–485.

Jr., R. J. B. & Agrawal, R. (2005), Data privacy through optimal k-anonymization., *in* 'Proceedings of the 21st International Conference on Data Engineering', IEEE Computer Society, pp. 217–228.

Lindell, Y. & Pinkas, B. (2000), Privacy preserving data mining, *in* 'Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology', Springer-Verlag, pp. 36–54.

Meyerson, A. & Williams, R. (2004), On the complexity of optimal k-anonymity, *in* 'PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems', ACM Press, New York, NY, USA, pp. 223–228.

Natwichai, J., Li, X. & Orlowska, M. (2005), Hiding classification rules for data sharing with privacy preservation, *in* 'Proceedings of 7th International Conference on Data Warehousing and Knowledge Discovery', Lecture Notes in Computer Science, Springer, pp. 468–467.

Oliveira, S. R. M. & Zaiane, O. R. (2003), Algorithms for balancing privacy and knowledge discovery in association rule mining, *in* '7th International Database Engineering and Applications Symposium', IEEE Computer Society, pp. 54–65.

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, USA.

Rizvi, S. & Haritsa, J. (2002), Maintaining data privacy in association rule mining, *in* 'Proceedings of the 28th Conference on Very Large Data Base', pp. 682–693.

Saygin, Y., Verykios, V. S. & Clifton, C. (2001), 'Using unknowns to prevent discovery of association rules', *SIGMOD Rec.* **30**(4), 45–54.

Sweeney, L. (2002*a*), 'Achieving k-anonymity privacy protection using generalization and suppression.', *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **10**(5), 571–588.

Sweeney, L. (2002*b*), 'k-anonymity: A model for protecting privacy.', *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **10**(5), 557–570.

Thuraisingham, B. (2002), 'Data mining, national security, privacy and civil liberties', *SIGKDD Explor. Newsl.* **4**(2), 1–5.

Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y. & Theodoridis, Y. (2004), 'State-of-the-art in privacy preserving data mining', *SIGMOD Rec.* **33**(1), 50–57.

Verykios, V. S., Elmagarmid, A. K., Bertino, E., Saygin, Y. & Dasseni, E. (2004), 'Association rule hiding', *IEEE Transactions on Data and Knowledge Engineering* **16**(4), 434–447.

Dynamic Labeling Schemes for Ordered XML Based on Type Information

Damien K. Fisher **Franky Lam** **William M. Shui** **Raymond K. Wong**

National ICT Australia
and School of Computer Science & Engineering
University of New South Wales
NSW 2052, Australia
Email: {damienf, flam, wshui, wong}@cse.unsw.edu.au

Abstract

With the increasing popularity of XML, there arises the need for managing and querying information in this form. Several query languages, such as XQuery, have been proposed which return their results in document order. However, most recent efforts focused on query optimization have either disregarded order or proposed static schemes in which updates are not handled efficiently. Some dynamic labelling schemes have been proposed but they do not consider type information that is usually available with the XML documents or maintained by the database system. This paper presents theoretically sound algorithms for maintaining the well-known region algebra labeling schemes, which can take advantage of type information, often present in XML documents.

Keywords: Order Maintenance, XML.

1 Introduction

Storing and querying XML [4] are both significantly more challenging problems than for relational data, which has a considerably simpler data model. The popularity of XML has led to a significant amount of research on both of these problems in recent years.

One of the most promising means of evaluating XML queries in query languages such as XPath [24] and XQuery [25] is through the use of *structural joins* [2]. These join algorithms rely heavily on a solution to the *ancestor-descendant problem*, that is, given two nodes in the database, determine whether one is the ancestor of the other. The importance of a fast ancestor-descendant operator to XML databases extends beyond structural join algorithms to touch almost all XML indexing techniques, and hence there has been a large amount of research on labeling schemes which answer this question in the literature, which we will discuss in Section 2. Until recently, however, the critical real-world issue of accommodating database insertions and deletions in such schemes has only been addressed [12, 23, 26].

A second important problem in XML databases is the *order maintenance problem*. XML has an intrinsic order, corresponding to the order of elements in the XML file. The corresponding traversal of the logical data tree is a preorder traversal. This order is important in many domains: for instance, in an XML document, the relative order of two paragraphs is most likely important semantically. Similarly, in biological database repositories order is of critical importance. As a result of this, standard XML query languages (e.g., XPath [24] and XQuery [25]) require the output of queries to be in document order by default. Maintaining enough information to implement a sort operator, whilst also allowing efficient updates, is a non-trivial task.

In some cases, maintaining XML order is unnecessary. For instance, while the order of the authors of a book is important, the order between books is generally not. However, there are still many large XML repositories where order throughout the document is of critical importance. Of particular interest are biological data repositories, which are frequently updated, and in which the relative position of nodes is important.

These two problems are closely related: Dietz [10] observed that any solution to the order maintenance problem can automatically be applied to produce a solution to the ancestor-descendant problem. The converse is not true, mainly because a solution to the ancestor-descendant problem need not distinguish between siblings, while the order maintenance problem does make a distinction in this case. Hence, the order maintenance problem is the more general of the two, and thus is the focus of this paper. While we give our results in terms of the order maintenance problem, it should be kept in mind that our results apply equally well to the ancestor-descendant problem.

In this paper, we demonstrate that the use of type information can substantially decrease the overhead of document ordering maintenance algorithms, which appear to have quite a high overhead for disk-bound structures in practice. We verify our results experimentally, proving their superiority to existing results in the literature. As described above, our results apply both to the order maintenance problem and the ancestor-descendant problem. Thus, this work is the first to give theoretically sound bounds on the update times for XML labeling schemes while utilizing type information from the schema.

The rest of this paper is organized as follows. Section 2 provides a brief survey of work related to document ordering. Section 3 defines the data model we will adopt in this paper, and defines a simple, but expensive, algorithm to determine document ordering, which is made use of in later sections. Section 4 describes a system which utilizes type information to speed up ordering indices. Section 5 presents empirical tests of our algorithms, and Section 6 concludes the paper.

2 Related Work

The most closely related work to this paper is that of ordered labeling schemes. The most thorough work in this area is that of Tatarinov et al [23], who study three techniques: global ordering, local ordering, and Dewey ordering. The global ordering scheme assigns a single integral value to each node in the database, and hence is very similar to the list labeling problem described above. The local ordering again assigns a single value to each node, which denotes its relative order amongst its siblings in the XML document. The Dewey ordering is similar to the Dewey decimal system [5], and stores with each node the concatenation of the local ordering identifiers of the node and

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology, Vol. 49. Gillian Dobbie and James Bailey, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

its ancestors. In each case, Tatarinov et al only considered naive, expensive, algorithms to handle updates — the work of this paper could be directly applied to update all three schemes more efficiently.

Another related problem is that of ancestor-descendant labeling. In brief, an ancestor-descendant labeling scheme allows one to quickly determine whether one node is an ancestor of another in the XML document. Given a solution to the order maintenance problem, one can construct a solution to the ancestor-descendant problem (the converse is not strictly true, since an ancestor-descendant labeling scheme does not need to maintain ordering information for siblings). The work of this paper can be adapted to address the ancestor-descendant problem. While many ancestor-descendant schemes have been described in the literature [1, 6, 8, 15–19, 26, 27], only a few of these schemes also provide a solution to the order maintenance problem, and even fewer handle the issue of updates in a non-trivial way.

Deschler et al [8] have recently devised a modified Dewey ordering scheme, which uses strings instead of numbers as values. The use of strings allows one to insert new nodes anywhere in the database, without having to relabel any other nodes. Unfortunately, this scheme suffers from a lower bound on the label length which is linear in the number of nodes in the database — this bound is impossible to circumvent in schemes which do not relabel other nodes, as shown by Cohen et al [6]. As our work does allow nodes to be relabeled, it is not affected by Cohen's result.

A novel recent work, by Wu et al [26], utilizes properties of prime numbers to provide an efficient ordered labeling scheme. In particular, they use the Chinese Remainder Theorem to find a mapping between the prime number labels of the nodes in the database, and their relative ordering. While the scheme is interesting, and only relabels nodes infrequently, the length of the identifiers is $O(\log n + \log\log n)$ bits, where n is the size of the database, as opposed to our schemes here which have identifiers of size $O(\log n)$ bits. Also, their scheme involves an indirection through a potentially large array in order to answer queries, which is an expensive bottleneck for large databases.

Since XML documents are represented as ordered trees, there is a close relation between this problem and the *order maintenance* problem addressed by Dietz and Sleator [3, 9]. Most XML storage schemes, such as [13–15, 19], make use of interval and preorder/postorder labeling schemes to support constant time order lookup, but fail to address the issue of maintenance of these labels during updates. Recently, Silberstein et al [21] proposed a data structure to handle ordered XML which guarantees both update and lookup costs. The primary difference between this paper and Silberstein el al [21] is that we also attempt to minimize space usage (and in fact keep the space requirement near the information theoretic minimum).

Finally, Fisher et al [12] addressed the order maintenance problem in the context of XML database systems. They experimentally demonstrated the poor performance of the $O(1)$ variant of Bender's algorithm for disk-bound structures with heavy read loads (a common database scenario), and instead proposed a simple one pass randomized algorithm, which has good performance when insertions are distributed across the database. However, their algorithm has worst case performance linear in the size of the database, which occurs when insertions are clustered together, a common occurrence in practice.

3 Formal Definitions

We will follow a common convention in the literature and model an XML document by a labeled, ordered, unranked tree. The order of the attributes of an element is undefined in XML (see, for example, the XQuery specification [25]); we will adopt the convention that the attributes come before the other children of the element, in some arbitrary, but fixed, order. This will have no impact on the results of this paper, as the document ordering between attribute nodes of an element is by definition arbitrary anyway. As we do not need to distinguish between elements, attributes, processing instructions, and other kinds of XML nodes in this paper, this model is suitable for our purposes; in fact, for most of our results we do not even need to make the tree labeled, as labels are irrelevant when considering document order (the only time we make use of this information is in Section 4).

Accessor	Description
PARENT(x)	Parent of x
NEXT-SIBLING(x)	Next sibling of x
PREV-SIBLING(x)	Previous sibling of x
FIRST-CHILD(x)	First child of x
PREORDER-PREVIOUS(x)	Node before x in document order
PREORDER-NEXT(x)	Node after x in document order

Table 1: Constant time accessor functions

The document ordering on an XML document is the total ordering defined by a preorder traversal of the corresponding tree [25]. In this paper, we will denote the document ordering by $<$. As the document ordering between attribute nodes of an element is implementation defined, for our purposes we can simply choose an arbitrary ordering amongst the attributes in our ordered tree representation, and use this as the document ordering.

Throughout this paper, we impose a specific physical data model on our XML database, which gives a set of accessor functions which we assume take constant time. We have carefully chosen this set of accessors so that it is likely that any reasonable native or relational XML database which handles ordered data would implement these accessors in constant time. The accessors needed are summarized in Table 1. Of these accessors, PREORDER-PREVIOUS and PREORDER-NEXT can easily be implemented in terms of the others, although in worst cast time linear in the depth of the database. In practice, however, the depth of an XML database is extremely small, and we can assume that these accessors will essentially run in constant time. In our implementation, we do not maintain these accessors explicitly, instead relying on the observed properties of real XML documents.

We assign to each node a unique identifier, the *object identifier*, or *oid*. Throughout this document, object identifiers will be represented by integers of word size (32 bits on many modern machines). We stress that the order of the object identifiers of two nodes x and y does not necessarily correspond to the document ordering on x and y.

This paper deals with document ordering in dynamic XML databases. For simplicity, we assume that each insertion or deletion only adds or removes a single leaf node. The insertion or deletion of entire subtrees can be modelled as a sequence of these atomic operations (with further optimisations possible).

4 Utilizing Schema Information

As described in the Related Work, most recent XML labelling proposals work on on any XML document. Certainly, a significant portion of XML documents in real life come with type information, typically in the form of a DTD [4], XML Schema [11], or a schema written in some other standard schema language. This type information is used for verification purposes and provides a primitive form of constraint checking. Even for untyped documents, there is generally a high degree of regularity, and a large body of research on schema inference can be used in these cases to generate a schema.

In this section, we provide an automated system which can utilize type information to optimize the use of a document ordering algorithm. There are several interesting aspects to our system. Firstly, the system should work with any of the numerous schema languages available, even though we frame our discussion in terms of the theoretically simplest, DTDs. Secondly, the algorithm can be used to improve the practical performance of *any* algorithm for the document ordering problem, and hence has wide applicability.

4.1 Problem Statement

We first give a formal definition of our problem. Firstly, we define a useful function $\mathcal{L}$:

Definition 1 *The function $\mathcal{L}$, from the set of XML trees over an element set Σ to the set of strings over Σ, maps a tree t to the string which results from listing the nodes of t in document order.*

It is clear from the definition that $\mathcal{L}$ is surjective, but not injective. The function $\mathcal{L}$ can be implemented extremely efficiently, in a single preorder traversal over an XML tree. If T is a set of trees, then we will use the convenient shorthand $\mathcal{L}(T) = \{\mathcal{L}(t) \mid \forall t \in T\}$.

Using this definition, we can now state the linearization problem precisely:

Definition 2 (The Linearization Problem) *The linearization problem has two parts:*

1. *Let S be a schema for XML trees defined over an element set Σ, so that S defines some tree language L. The first part of the linearization problem is to convert S into a schema S' over Σ^*, such that the language defined by S' is exactly $\mathcal{L}(L)$.*
2. *Given a language L of strings over an alphabet Σ defined by some schema S, construct the linearization graph corresponding to L.*

In the above definition, we have been deliberately vague about what a "schema" is. In fact, it is conceivable that for some XML schema language, solving the first part of the problem is impossible. Similarly, it is possible that for some string schema language, the second part of the problem is impossible. However, we will show below that, for the case of DTDs, we can find solutions to both problems. In any event, a "schema" is loosely defined as a concise representation of a possibly infinite language.

In the second part of the definition above, reference was made to the linearization graph of a language. Essentially, this graph contains all the information relevant to the relative order of symbols in strings accepted by that language. For instance, in the regular language 0^*1^*, we always know that for any string accepted by this language which contains both a 0 and a 1, then the 0 must always come before the 1. As a more complicated example, Figure 4 gives the linearization graph for the fragment of the DBLP DTD of Figure 1.

We formalize the notion of this linearization information through the use of an extraction function:

Definition 3 *For an alphabet Σ, define $\mathcal{E} : \Sigma^* \to \Sigma \times \Sigma$, as:*

$$\mathcal{E}(n_1 n_2 \ldots n_m) = \{\langle n_i, n_i + 1\rangle \mid i \in \{1, \ldots, m-1\}\}$$

For a set of strings L, we write $\mathcal{E}(L) = \bigcup_{s \in L} \mathcal{E}(s)$.

Informally, the information which the function $\mathcal{E}$ extracts from a set of strings is simply all possible symbols which might follow a given symbol. Hence, by constructing a graph which represents this information, we can follow paths through the graph to determine the relative ordering of two symbols in the alphabet for a particular language. Note that if Σ is countably infinite (as is the case with XML trees), then $\mathcal{E}(L)$ may not necessarily be finite. However, we will give below a simple technique to ensure that $\mathcal{E}(L)$ is always finite in the cases we are considering.

We are now in a position to define the linearization graph for a language:

Definition 4 (The Linearization Graph) *The linearization graph of a language L is the directed graph with vertex set Σ and edge set $\mathcal{E}(L)$.*

It is interesting to note that the linearization graph represents a language itself. We say that string $n_1 n_2 \ldots n_m$ is accepted by the graph if and only if we can find a path through the graph $n_1 \to n_2 \to \ldots \to n_m$. This language can be thought of as the set of strings which is generated only by the linearization information we have extracted, and contains no additional structure. It is easy to see that this language is regular:

Lemma 1 *The language of the linearization graph defined above is regular.*

Proof: In this proof, we make use of the definition of a context free grammar (see Definition 5).

In the following, the edge set of the linearization graph is E. We define a context free grammar $\langle V, \Sigma, R, S\rangle$ as follows:

- $V = \{R_x \mid \forall x \in \Sigma\} \cup \{S\}$; that is, the set of variables is in one-to-one correspondence with the set of terminals (the alphabet of the language), with the exception of one additional variable (the start variable S);
- For each $x \in \Sigma$, let $E_x = \{y \mid \langle x, y\rangle \in E\}$ (that is, E_x is the set of successors of x in the graph). For each $y \in E_x$, define a rule $R_x \to xR_y$. Let the set of rules R consist of all rules defined in this fashion, plus a set of rules $\{S \to R_x \mid \forall x \in \Sigma\}$.

By construction, this context free grammar is a regular grammar, and hence the language it generates is also regular. It is also clear that this language is equivalent to the language of the linearization graph: for every vertex x in the graph, we have constructed a rule R_x in the graph which corresponds to the language membership test defined above. More precisely, any string which is accepted by the rule R_x corresponds to a path in the graph beginning at node x. □

4.2 Converting a DTD to a CFG

We now turn to solving the first part of the linearization problem. While our algorithm should work with any currently used schema language, we solve the problem only for DTDs, because they are one of the most commonly used schema languages, as well as the simplest.

It is well-known that DTDs correspond to regular tree grammars [20]. In fact, we do not need the power of regular tree grammars, as we only need to consider XML documents as linked lists of nodes in document order. Therefore, we will represent DTDs as context-free grammars (the definition below is taken from Sipser [22]):

Definition 5 (Context-Free Grammar (CFG)) *A context-free grammar is a 4-tuple $\langle V, \Sigma, R, S\rangle$ where:*

1. *V is a finite set called the variables;*
2. *Σ is a finite set, disjoint from V, called the terminals; and*
3. *R is a finite set of rules, with each rule, written $v \to v_1 v_2 \ldots v_n$, being a variable v and a string $v_1 v_2 \ldots v_n$ of variables and terminals; and*
4. *$S \in V$ is the start symbol of the grammar.*

When we want to model the empty string, we will follow the normal convention, and use the special symbol ϵ, which we implicitly add to the set of terminals of the language. We will ignore this technicality throughout our presentation, and simply use ϵ where convenient.

We now give the straightforward process of converting a DTD to a CFG which is, for our purposes, equivalent to the DTD. For simplicity, we only consider elements, ignoring other node types such as attributes and processing instructions. Similarly, because we are uninterested in data values, we ignore occurences of `#PCDATA` in the DTD. With these simplifications in mind, we define a DTD as follows:

Definition 6 (DTD) *Let $\mathcal{T}$ be the countably infinite set of element types. Then a DTD is a finite set $\{\langle t, r\rangle \mid \forall t \in T\}$, where T is a finite subset of $\mathcal{T}$, and each r is a regular expression over T.*

The only subtle issue remaining with this definition is the `#ANY` keyword, which can represent any combination of elements. To handle this, we adjoin to T above a special element type α, which represents all element types not appearing in the DTD, i.e., all element types in $\mathcal{T} - T$. We replace any occurrence of `#ANY` with the regular expression $(\alpha|t_1|\ldots|t_n)^*$, where $\{t_1, \ldots t_n\} = T$. Similarly, we add a rule $\langle\alpha, (\alpha|t_1|\ldots|t_n)^*\rangle$ to the DTD, so that α is a well-defined element type. It is clear that this definition preserves the semantics of `#ANY`. Herein, we assume that, if necessary, the rule for α has been added to the DTD, and α has been added to its set of types T. We note that, at this point, our DTD only refers to a finite set of elements, because we have merged the remaining (infinite) number of element types together into a single, special element "type", α. Thus, we have in essence made the alphabet for our CFG finite, and hence ensured that the linearization graph we construct is finite.

From this set of rules, it is easy to construct the desired CFG. The set of terminals Σ becomes the set of types in the DTD T. For each rule $\langle t, r\rangle$ in the DTD, we define a rule $R_t \rightarrow tS_r$, where S_r is a variable defining the regular expression r. The variable S_r is obtained from r by first converting r from a regular expression over T to a regular expression r' over $R_V \rightarrow \{R_t \mid t \in T\}$, using the obvious isomorphism. Using well-known techniques, we then convert r' to a context free grammar $\langle V_r, R_V, R_r, S_r\rangle$. Finally, we obtain our complete CFG as:

$$\langle \bigcup_{t\in T}\{R_t\} \cup \bigcup_r \Sigma_r, T, \bigcup_{t\in T}\{R_t \rightarrow tS_r\} \cup \bigcup_r R_r, S\rangle$$

The only remaining problem is to define the start symbol S in the definition above. Unfortunately, DTDs do not have a "start element" we can map to S. However, in practice, we can always find an element which appears as the root element in every document designed to conform to the DTD, and if this element is t, then we set $S = R_t$. Alternatively, we could follow the DTD semantics more precisely and define a start symbol with additional rules $\{S \rightarrow R_t \mid \forall t \in T\}$, so that all DTD types are permitted to be the root type; as it turns out, the choice makes no difference to the construction.

We can summarize the above in the following theorem, whose proof follows directly from the construction:

Theorem 1 *Given a DTD, we can obtain a CFG, such that if a document $\mathcal{D}$ is accepted by the DTD then the document, when considered as a list of nodes in document order, is accepted by the CFG. Conversely, for any string accepted by the CFG, there exists a document accepted by the DTD which, when considered as a list of nodes in document order, is equivalent to the string.*

Thus, we have found a construction algorithm which solves the first part of the linearization problem for DTDs.

```
<!ELEMENT dblp (article|inproceedings|...)*>
<!ENTITY %field "author|editor|...">
<!ELEMENT article (%field;)*>
<!ELEMENT inproceedings (%field;)*>
<!ELEMENT author (#PCDATA)>
<!ELEMENT editor (#PCDATA)>
...
```

Figure 1: An excerpt from the DBLP DTD.

$$\begin{aligned} dblp &= (article|inproceedings|\ldots)^* \\ article &= (author|editor|\ldots)^* \\ inproceedings &= (author|editor|\ldots)^* \\ author &= \epsilon \\ editor &= \epsilon \end{aligned}$$

Figure 2: The DTD fragment of Figure 1, converted into a set of regular expressions.

As an example of the translation process in action, consider the fragment of the DBLP DTD given in Figure 1. The translation of this DTD fragment into a set of regular expressions is given in Figure 2, and using our translation process plus a few simplifications, we obtain the CFG of Figure 3. We will use this as a running example throughout the remainder of this paper.

4.3 Linearizing a CFG

We now consider the second part of the linearization problem for DTDs: given a CFG, how do we construct its linearization graph? Fortunately, it is easy to give a recursive construction algorithm for this task.

Throughout this section, we will fix a CFG $\langle V, \Sigma, R, S\rangle$ for which we wish to find the linearization graph. Our first step in this task is to define a few useful functions which we will make use of in our construction algorithm.

The first function we define is `ACCEPTS-EMPTY`, which, given a CFG, determines whether it accepts the empty string or not. The pseudocode for this function can be found in Algorithm 1. Essentially, a simple dynamic programming approach is used, which determines whether each rule accepts the empty string. We note that this function, and the others we define next, only work correctly on CFGs which accept at least one finite string. As CFGS which do not accept any finite strings are pathological, and not of the kind we are interested in (CFGs constructed from DTDs are guaranteed to accept at least one finite string), we will ignore these cases in this section. The proof of correctness is easy:

Lemma 2 *Algorithm 1 returns true if and only if a CFG accepts the empty string.*

Proof: Firstly, Algorithm 1 must always terminate, because it visits each variable at most once, and the number of variables is finite. Now suppose that Algorithm 1 returns true. In this case, if we trace out its execution path, we get a valid parse tree for the empty string, and hence the grammar must accept the empty string.

Conversely, suppose the grammar accepts the empty string, so that there is a parse tree matching it. We can

$$\begin{aligned} R_1 &\rightarrow dblp\ R_2 \\ R_2 &\rightarrow \epsilon \mid R_2\ R_3 \mid R_2\ R_4 \\ R_3 &\rightarrow article\ R_5 \\ R_4 &\rightarrow inproceedings\ R_5 \\ R_5 &\rightarrow \epsilon \mid R_5\ author \mid R_5\ editor \end{aligned}$$

Figure 3: The DTD fragment of Figure 1, converted into a CFG.

Algorithm 1 Determines whether a CFG accepts the empty string.

```
ACCEPTS-EMPTY($\langle V, \Sigma, R, S\rangle$)
EMPTY[$v$] $\leftarrow$ $unknown$ $\forall v \in V$  (this is a global variable)
ACCEPTS-EMPTY-RECURSE($\langle V, \Sigma, R, S\rangle, S, \emptyset$)
return EMPTY[$S$]

ACCEPTS-EMPTY-RECURSE($\langle V, \Sigma, R, S\rangle, curr, seen$)
if EMPTY[$curr$] $\neq unknown \lor curr \in seen$ then
    return
$seen \leftarrow seen \cup \{curr\}$
for each rule $curr \rightarrow a_1 \ldots a_n \in R$ do
    if $n = 1 \land a_1 = \epsilon$ then
        EMPTY[$curr$] $= true$
        return
    for $i \in \{1, \ldots, n\}$ do
        if $a_i \in V$ then
            ACCEPTS-EMPTY-RECURSE($\langle V, \Sigma, R, S\rangle, a_i, seen$)
            if EMPTY[$a_i$] $\neq true$ then
                break
        else
            break
        EMPTY[$curr$] $\leftarrow true$
        return
EMPTY[$curr$] $\leftarrow false$
```

be guaranteed to find a parse tree such that in every root-to-leaf path, each variable occurs only once. If this were not the case, then for each such path $V_1 \rightarrow V_2 \rightarrow V \rightarrow \ldots \rightarrow V \rightarrow \ldots \rightarrow V_n$, we can delete the portion of the tree corresponding to the part of the path between the two non-unique occurrences, and still have a valid parse tree for the empty string. Hence we are left with a parse tree which corresponds to a traversal of the variables in the grammar, where every variable is visited at most once in each path. From this parse tree, we can construct an execution path through the algorithm. □

Using the function ACCEPTS-EMPTY, we now define a function, FIRST, which, given a CFG, finds the set of terminals with which all strings accepted by the CFG must begin. The pseudocode is given in Algorithm 2; as the correctness proof for this algorithm is very similar to that of ACCEPTS-EMPTY, it is omitted.

Finally, we define a function LAST, which finds the set of terminals with which all strings accepted by the CFG must end. As this algorithm is virtually identical to that of Algorithm 2 (instead of the ascending iteration in line 5, a descending iteration is performed), it is omitted.

While we have only defined FIRST and LAST to return the sets of terminals corresponding to a particular variable (in particular, the start variable), we can extend these functions to arbitrary strings over $V \cup \Sigma$ as follows: if we have such a string s, then add a rule to the grammar $T \rightarrow s$, where T is some terminal not currently in V. Then, run FIRST and LAST on the context free grammar $\langle V \cup \{\mathtt{T}\}, \Sigma, R \cup \{T \rightarrow s\}, T\rangle$, and use these sets as the result for this string. Thus, given a string of terminals and variables, FIRST and LAST return the sets of terminals with which strings accepted by the rule defined by that string must begin or end.

We are now in a position to give the construction algorithm for obtaining the linearization graph from a CFG. In essence, we recursively build the edge set for the graph from the edge sets for the linearization of subexpressions in the CFG. The entire process is given in Algorithm 3. Note that we only need to construct the edge set incrementally, as we can leave the vertex set of the graph to be the entire alphabet Σ throughout.

We summarize the correctness of this construction in the following theorem:

Theorem 2 *Algorithm 3 constructs the linearization graph for a CFG.*

The proof of this theorem follows directly from the fact that running the extraction function $\mathcal{E}$ on the language defined by the grammar is equivalent to the process which happens in line 15 of the algorithm.

4.4 Using a Linearization

Once we have a linearization, it is a simple matter to extract document ordering information from it. We construct the graph of strongly connected components G' for the linearization G, using standard techniques [7]. It is clear that G' is a directed acyclic graph. The following lemma easily follows from the definition of the linearization graph (Definition 4):

Lemma 3 *Let $G = \langle V, E\rangle$ be the linearization of a DTD, and $G' = \langle V', E'\rangle$ be the graph of strongly connected components of G. Then for any document $\mathcal{D}$ validated by the DTD, for any nodes $x, y \in \mathcal{D}$, if $[t(y)]$ is reachable from $[t(x)]$ in G', then $x < y$ in document order, where $t(x)$ is the type of x and $[v] \in V'$ is the strongly connected component of vertex $v \in V$ in G.*

Thus, ordering information can be easily computed at database creation time by traversing G', and storing the relative ordering between the strongly connected components in an array of size $O(|V'|^2)$. More sophisticated techniques can be used if $|V'|$ is large, although in practice it generally is not. We do not know the relative order of nodes lying within the same strongly connected component. In this case, we can apply a document ordering algorithm such as Bender's algorithm; to see why this is so, it is sufficient to note that each strongly connected component is a contiguous block in document order, because the document ordering on nodes carries through to an ordering on the strongly connected components (by Lemma 3).

Note that we can apply a *different* instance of the algorithm to each strongly connected component, which can result in reducing both the size of the tag and decreasing the cost of updates. As an extension, if we have selectivity information indicating the approximate size of each strongly connected component, then we can use this information to choose the optimal algorithm for each component. For instance, for very large components it may be

Algorithm 2 Determines with which terminals a string accepted by a CFG must begin.

```
FIRST($\langle V, \Sigma, R, S \rangle$)
  ACCEPTS-EMPTY($\langle V, \Sigma, R, S \rangle$)
  (we make use of the EMPTY global variable)
  FIRST[$v$] $\leftarrow \emptyset \; \forall v \in V$
  FIRST-RECURSE($\langle V, \Sigma, R, S \rangle, S, \emptyset$)
  return FIRST[$S$]

FIRST-RECURSE($\langle V, \Sigma, R, S \rangle, curr, seen$)
  if $curr \in seen$ then
      return
  $seen \leftarrow seen \cup \{curr\}$
  for each rule $curr \rightarrow a_1 \ldots a_n$ do
      for $i \in \{1, \ldots, n\}$ in ascending order do
          if $a_i \in V$ then
              FIRST-RECURSE($\langle V, \Sigma, R, S \rangle, a_i, seen$)
              FIRST[$curr$] $=$ FIRST[$curr$] $\cup$ FIRST[$a_i$]
              if EMPTY[$a_i$] $\neq true$ then
                  break
          else
              FIRST[$curr$] $=$ FIRST[$curr$] $\cup \{a_i\}$
              break
```

Algorithm 3 Given a CFG, return the edge set for the corresponding linearization.

```
LINEARIZE($\langle V, \Sigma, R, S \rangle$)
  return LINEARIZE-RECURSE($\langle V, \Sigma, R, S \rangle, S, \emptyset$)

LINEARIZE-RECURSE($\langle V, \Sigma, R, S \rangle, a_1 a_2 \ldots a_n, seen$)
  if $n = 1 \wedge a_1 \in V$ then
      if $a_1 \in seen$ then
          return
      $seen \leftarrow seen \cup \{a_1\}$
      $E \leftarrow \emptyset$
      for each rule $curr \rightarrow a'_1 \ldots a'_{n'}$ do
          $E \leftarrow E \cup$
              LINEARIZE-RECURSE($\langle V, \Sigma, R, S \rangle, a'_1 \ldots a'_{n'}$)
      return $E$
  elif $n \leq 1$ then
      return $\emptyset$
  else
      $m \leftarrow \lfloor \frac{n}{2} \rfloor$
      $E_1 \leftarrow$ LINEARIZE-RECURSE($\langle V, \Sigma, R, S \rangle, a_1 \ldots a_m$)
      $E_2 \leftarrow$ LINEARIZE-RECURSE($\langle V, \Sigma, R, S \rangle, a_{m+1} \ldots a_n$)
      return $E_1 \cup E_2 \cup \{\langle x, y \rangle \mid \forall x \in$ LAST$(a_1 \ldots a_m)$,
          $\forall y \in$ FIRST$(a_{m+1} \ldots a_n)\}$
```

suitable to choose the $O(1)$ variant of Bender's algorithm, but for smaller components we can choose an algorithm with less space overhead.

4.5 Annotating a Linearization

While the scheme described so far has been very general, it has some limitations in practice. We have found that many schemas used in the real world are significantly looser than they might ideally be, and that this inexactness propogates itself into the analysis above. For instance, consider the linearization in Figure 4. In this case, it is clear that there are only two strongly connected components, one consisting of the single node type dblp. Clearly, this is not particularly useful information, because, as dblp elements are the root elements in practice, we have only been able to deduce the relative order of a single node.

However, we can still use type information in a heuristic manner which gives good practical performance in many cases. In Figure 4, we have annotated the linearization with dotted edges, which represent that, in our particular database system, from any node we can access its parent node. In fact, we also have an accessor which allows any parent node to access its first child. This accessor does not appear in Figure 4, because the DTD does not give us enough information to determine what the first child is. For instance, consider a node of type article. From the DTD, we are unable to infer whether this node will even *have* a first child. On the other hand, we always know that any node (except dblp) will have a parent. We will now formalize the notion of an acceptable accessor:

Definition 7 *If $G = \langle V, E \rangle$ is the linearization graph, then an* accessor *is a map from $\phi : V_1 \rightarrow V_2$, where $V_1, V_2 \subseteq V$, such that $\forall x \in V_1$, $\phi(x)$ can be retrieved from x.*

Not all accessors are particularly interesting in terms of document ordering. The interesting accessors are those that allow us to infer some information about the relative ordering between nodes. For instance, the parent accessor is useful, because if two nodes have different parents, then clearly their relative order is related to the relative order of

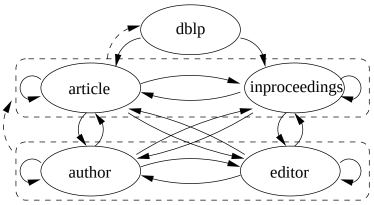


Figure 4: Linearization of the DTD fragment of Figure 1.

their parents. Thus, we are interested in *order preserving* accessors:

Definition 8 *An order preserving accessor is an accessor* $\phi : V_1 \to V_2$ *such that* $\forall x, y \in V_1$, *if* $x < y$ *then* $\phi(x) \leq \phi(y)$.

The accessors available to us differ depending upon the underlying implementation. As an example, we define the parent accessor for a particular element type t. Let V_1 be the set of all element types that appear alongside t in regular expressions in the DTD (that is, V_1 is the set of all possible sibling types of t), and and V_2 be the element types which have defining regular expressions containing an element type in V_1. Then in our implementation there is a map $\phi : V_1 \to V_2$ which yields the parent of any $v \in V_1$. This accessor is illustrated in Figure 4 — the regions surrounded by dotted areas are the domain and codomain of this accessor, with the dotted lines representing the movement through the accessor.

While we are interested in order preserving accessors, we are not interested in all of them. For instance, suppose it is possible to access the root node of the database. Then it would be trivial to define an accessor from the set of all element types V which accessed the root. This would not be a terribly useful accessor in terms of document ordering, however, because $\forall x, y \in V$, the image under the map would be equal, and hence traversing the accessor would never yield additional information about the relative order of x and y. In this case, the cost of using the accessor outweighs the benefits.

Therefore, we introduce a threshold parameter, controlled by the database creator, which specifies the maximum acceptable cost of an accessor. This cost could be measured in one of several ways, and is mainly dependent upon the underlying implementation. For the rest of our paper, we will use the cost function that was best suited to our implementation — the probability of incurring a disk access by using an accessor. Under this cost measure, an accessor which frequently accessed objects lying in the same page would score more highly than an accessor which accessed objects in other pages. For a given set of accessors, we assume we have the approximate probability of a disk access incurred by following that accessor. This information could come from one of several sources:

- The clustering subsystem of the database;
- The database creator; or
- An analysis of a small representative sample of the data to be stored in the database.

We do not believe that in practice it is difficult to produce these cost estimates. For instance, if DBLP is stored in document order, then it is fairly obvious that each record (where a record is a complete subtree rooted at `article`, `inproceedings`, etc.) will generally lie on a single page, and hence for intra-record accesses the likelihood of a disk access is quite low.

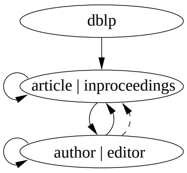


Figure 5: Linearization of the DTD fragment of Figure 1 after coalescing.

4.6 Coalescing

Once we have a set of accessors, along with their corresponding costs, we *coalesce* portions of the linearization. Informally, we merge nodes which have no accessors between them, and which lie in the same strongly connected component. The motivation is that as we are unable to infer any information about the relative ordering of nodes in the database corresponding to these types, there is no point in maintaining them as separate nodes in the linearization.

Given a set of accessors $\mathcal{A}$, we coalesce using a simple greedy heuristic. Firstly, we remove any accessor $\phi : V_1 \to V_2 \in \mathcal{A}$ such that V_1 or V_2 are not strictly contained in a strongly connected component of the linearization. We then repeat the following steps until $\mathcal{A}$ is empty:

1. Remove the accessor $\phi : V_1 \to V_2 \in \mathcal{A}$ which has the smallest cost, and annotate the graph with ϕ.
2. Merge the nodes in V_1 into a single node, and similarly merge the nodes in V_2 into another node in the graph.
3. Remove any $\phi' : V_1' \to V_2' \in \mathcal{A}$ where either V_1' or V_2' have a non-empty intersection with either V_1 or V_2.

The process of running this procedure on the linearization of Figure 4 is shown in Figure 5. In this example, we assigned a high cost to the accessor from `article` and `inproceedings` to `dblp`, because this almost always incurred a disk read. Similarly, a low cost was assigned to the accessor from `author` and `editor` to `article` and `inproceedings`, because these frequently lay on the same disk page, as we clustered the nodes in the database into document order.

4.7 Using an Annotated Linearization

Using an annotated linearization is similar to using a linearization. We use the techniques described in previous sections when comparing nodes belonging to two different strongly connected components. When the nodes lie in the same strongly connected component, however, we now see if they lie in the same node in the linearization (which may be the result of merging several types). If they are, and this node has an available accessor, we first traverse the accessor and determine the relative ordering under the image of the accessor. Only in the event that this does not resolve the relative order of the two nodes do we compare them using their own tag values.

This scheme works because we have split portions of the strongly connected components, in such a way that we can again apply a different instance of the document ordering algorithm. For instance, in Figure 5, we can apply a separate instance of the document ordering algorithm to each node in the graph. In fact, for the `author|editor` node, we apply a separate instance of the ordering algorithm to each set of nodes S such that $\forall x, y \in S$, $\phi(x) = \phi(y)$. It is easy to see that such a set of nodes must be contiguous (in document order), from the definition of an order preserving accessor. More precisely, for any node in the graph with an outgoing accessor (the heuristic above ensures there is at most one outgoing accessor), we apply a separate instance of an ordering algorithm to each set of nodes S such that $\forall x, y \in S, \phi(x) = \phi(y)$. Thus, we have refined the sets upon which we maintain ordering information, at the cost of a more complicated query mechanism.

4.8 Issues with Other Schema Languages

While we have only discussed the linearization of DTDs, it is not difficult to conceive of similar strategies for dealing with other popular schema languages, such as XML Schemas. However, there are a few issues that are worth raising:

- It is possible, and perhaps desirable, that a schema language might offer the ability for the user to specify that certain parts of an XML document are set oriented, instead of list oriented, and hence that ordering information should not be maintained for these portions. For example, in the DBLP database, while ordering might be important within a record, it is unlikely that it is important outside the record. It would be easy to accommodate such a feature within our framework, but it should be noted that sometimes, it may be easier to maintain ordering information in unordered regions anyway. For instance, suppose we have a linked list of elements $O_1 \rightarrow U_1 \rightarrow U_2 \rightarrow O_2$, where ordering information must be maintained for O_1 and O_2, but not for U_1 and U_2. Unless there is a way to navigate directly from O_1 to O_2, it will probably be faster to maintain ordering information for U_1 and U_2 too, since they must be traversed when moving between O_1 and O_2.
- Some schema languages, notably XML Schema, allow the use of *local types*, that is, one element might have distinctly different types, depending on its local context. This feature can be easily incorporated into our overall framework, by changing the vertex set of the linearization graph from the set of all elements to the set of all unique local types.

5 Experimental Results

5.1 Experimental Setup

We performed several experiments to determine the effectiveness of the various algorithms covered in this paper. All experiments were run on a dual processor 750 MHz Pentium III machine with 512 MB RAM and a 30 GB, 10,000 rpm SCSI hard drive. We used as our data set the DBLP database. Our experiments were performed on a disk-bound database, without transactions or concurrency enabled.

Our experiments were designed to investigate the worst case behavior of each of the document maintenance algorithms implemented. We chose to focus on the worst case for several reasons. Firstly, previous work [12] developed an algorithm which has very good average case performance (for a particular definition of average), but very poor worst case performance. As our new work gives algorithms with guaranteed worst case performance, we emphasize this in our experiments. As we also demonstrate that this worst case performance is extremely good, it follows that the average case performance is also good. More practically, however, at the time of writing it is very difficult to determine a suitable "average" case for usage of document ordering indices, simply because XML is still an immature technology.

For each experiment, we investigated the performance of four different algorithms:

1. The randomized algorithm of Fisher et al [12];
2. The $O(1)$ and $O(\log n)$ variants of Bender et al [3];
3. The randomized variant of Bender's algorithm by extending it to take an additional parameter, c, which gives the number of nodes that should share the same tag value. It is straightforward to generalize the $O(\log n)$ algorithm to this case: instead of defining the density in a range of tag values as $\frac{n}{2^i}$, where n is the number of nodes in that range, we use $\frac{n}{2^i c}$. It is easy to generalize the proof for Bender's algorithm to this case. For this algorithm, we used values of c from the set $\{5, 10, 50, 100, 200, 1000\}$; and
4. The $O(\log n)$ variant of Bender's algorithm, but augmented with the linearization graph of the DBLP DTD, using the methods of Section 4.

5.2 Experiment 1: Performance on a Bulk Insert

In this experiment, we evaluated the performance of the algorithms under a uniform query distribution. The experiment began with an empty database, which was then gradually initialized with the DBLP database. After every insertion, on average r reads were performed, where r was a fixed parameter taken from the set $\{0.01, 0.10, 1.00, 10.0\}$. Each read operation picked two nodes at random from the underlying database, using a uniform probability distribution, and compared their document order. In all of our experiments, we measured the total time of the combined read and write operations, the number of read and write operations, and the number of relabelings. However, due to space considerations, and the fact that the other results were fairly predictable, we only include the graphs for total time.

This experiment was designed to test what is a worst case scenario for all the algorithms except for the randomized algorithm. We included this experiment because the extremely heavy paging incurred by the uniform query distribution demonstrates some of the practical problems with Bender's $O(1)$ algorithm.

As can be seen from the results in Figure 6, the randomized algorithm is easily the best performer. The reason for this excellent performance is because on every insertion the randomized algorithm simply added one to the identifier of the last node in the database. While we could have added a special case to the other algorithms to handle insertions at the end of the database in a similar fashion, we instead opted to treat them no differently from insertions anywhere else in the database, to indicate the worst case performance of the algorithms. We included the randomized algorithm as a useful baseline to compare the other algorithms with, because it represents the fastest possible implementation for this experiment. Nevertheless, the fact that this is a special case for the randomized algorithm (and could similarly be made a special case for the other algorithms) should be kept in mind.

We note that, as the ratio of reads increases, the performance of both Bender's $O(1)$ algorithm and the schema based algorithm degrades relative to the other algorithms. We attribute this in both cases to the extra indirection involved in reading from the index. As values of $r > 100$ are common in practice, we expect that the behavior of the $O(1)$ algorithm will be even worse than even the $O(\log n)$ variant for many real-life situations. This was demonstrated more emphatically by Fisher et al [12]. Also, because of the extremely heavy paging, even the small paging overhead incurred by an algorithm such as the schema based algorithm, which only infrequently loads in an additional page in due to a read from the index, has a massive effect on the performance. Thus, although this experiment is slightly contrived, it does demonstrate that in some circumstances the indirection involved becomes unacceptable, given that values of r in real life will often be 100 or 1000. We note that one advantage of the schema based algorithm is that we can remove the indirection if necessary, whereas with the $O(1)$ algorithm, we cannot.

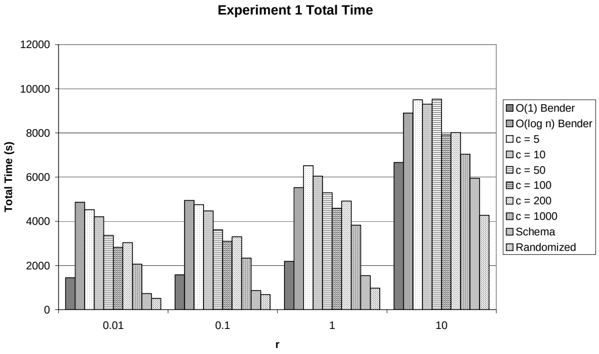


Figure 6: Experiment 1 Results

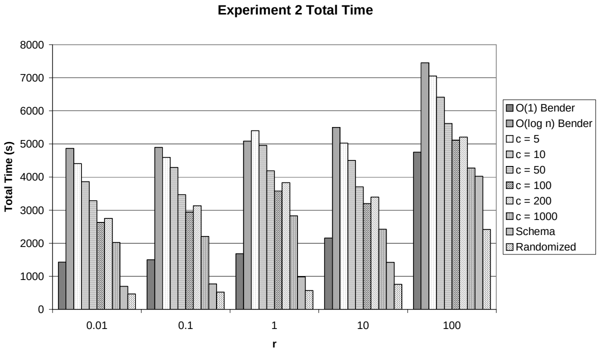


Figure 7: Experiment 2 Results

5.3 Experiment 2: Performance on a Non-Uniform Query Distribution

This experiment was identical to the first experiment, except that the reads were sampled from a normal distribution with mean $|D|/2$, and variance $|D|/10$, and we took $r \in \{0.01, 0.10, 1.00, 100.0\}$. The idea was to reduce the heavy paging of the first experiment, and instead simulate a database "hot-spot", a phenomenom which occurs in practice.

As can be seen from the results of Figure 7, this experiment took substantially less time to complete than the first experiment. It can be seen that, apart from the randomized algorithm (which again performed the minimal possible work), the schema based algorithm is clearly the best algorithm. Indeed, it is impressive that it came so close to the optimal performance.

5.4 Experiment 3: Worst-Case Performance for the Randomized Algorithm

The previous two experiments showed that the randomized algorithm had very good performance for the special case of appending to the end of the datbase. We demonstrate in this experiment that, in some cases, it has very bad performance, far worse than the other algorithms. This experiment was identical to the first experiment, except that instead of inserting DBLP records at the end of the database, we inserted them at the beginning. As the experiment would take an unreasonable amount of time to run on the full DBLP, and because the worst case performance of the randomized algorithm is so pronounced in this case, we only ran the experiment on a small subset of DBLP.

As can be seen from the results in Figure 8, all the other algorithms easily beat the randomized algorithm's performance. Hence, in situations where worst case bounds must be guaranteed, the randomized algorithm is not a good choice.

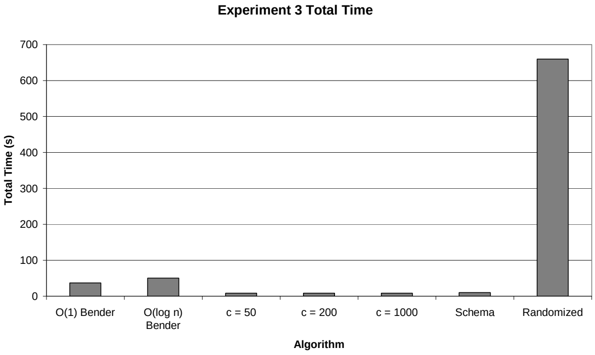


Figure 8: Experiment 3 Results

6 Conclusions

We developed a general scheme which utilizes type information to improve the speed of document ordering indices. This work is especially significant due to its wide applicability, as it is not tied to any particular ordering algorithm. We have found that in practice many large XML repositories have a DTD or some other schema for constraint purposes, and hence we expect that this work will have great practical impact.

References

[1] Serge Abiteboul, Haim Kaplan, and Tova Milo. Compact labeling schemes for ancestor queries. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 547–556. Society for Industrial and Applied Mathematics, 2001.

[2] Shurug Al-Khalifa, H. V. Jagadish, Nick Koudas, and Jignesh M. Patel. Structural Joins: A Primitive for Efficient XML Query Pattern Matching. In *ICDE*. IEEE Computer Society, 2002.

[3] Michael A. Bender, Richard Cole, Erik D. Demaine, Martin Farach-Colton, and Jack Zito. Two simplified algorithms for maintaining order in a list. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA 2002)*, volume 2461 of *Lecture Notes in Computer Science*, pages 152–164, Rome, Italy, September 17–21 2002.

[4] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler. Extensible Markup Language (XML) 1.0 (second edition). http://www.w3.org/TR/2000/REC-xml-20001006, 2000.

[5] Online Computer Library Center. Introduction to the Dewey Decimal Classification. http://www.oclc.org/oclc/fp/about/about_the_ddc.htm.

[6] Edith Cohen, Haim Kaplan, and Tova Milo. Labeling Dynamic XML Trees. In *Proceedings of PODS*, pages 271–281, New York, June 3–5 2002. ACM Press.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*. MIT Press and McGraw-Hill Book Company, 6th edition, 1992.

[8] Kurt Deschler and Elke Rundenstiner. MASS: A Multi-Axis Storage Structure for Large XML Documents. In *To appear in Proceedings of the Twelfth International Conference on Information and Knowledge Management*, 2003.

[9] P. Dietz and D. Sleator. Two algorithms for maintaining order in a list. In *Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 365–372. ACM Press, 1987.

[10] Paul F. Dietz. Maintaining order in a linked list. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 122–127, 1982.

[11] D. C. Fallside (Eds). “XML Schema Part 0: Primer”. W3C Recommendation, May 2001. http://www.w3.org/TR/xmlschema-0.

[12] Damien K. Fisher, Franky Lam, William M. Shui, and Raymond K. Wong. Efficient ordering for xml data. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM 2003)*, November 2003.

[13] Torsten Grust. Accelerating XPath location steps. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 109–120. ACM Press, 2002.

[14] Alan Halverson, Josef Burger, Leonidas Galanis, Ameet Kini, Rajasekar Krishnamurthy, Ajith Nagaraja Rao, Feng Tian, Stratis Viglas, Yuan Wang, Jeffrey F. Naughton, and David J. DeWitt. Mixed Mode XML Query Processing. In *Proceedings of the 29th International Conference on Very Large Databases (VLDB)*, pages 225–236. Morgan Kaufmann, 2003.

[15] H. V. Jagadish, S. Al-Khalifa, A. Chapman, L. V. S. Lakshmanan, A. Nierman, S. Paparizos, J. M. Patel, D. Srivastava, N. Wiwatwattana, Y. Wu, and C. Yu. TIMBER: A native XML database. *VLDB Journal: Very Large Data Bases*, 11(4):274–291, 2002.

[16] Haim Kaplan, Tova Milo, and Ronen Shabo. A comparison of labeling schemes for ancestor queries. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 954–963. Society for Industrial and Applied Mathematics, 2002.

[17] W. Eliot Kimber. HyTime and SGML: Understanding the HyTime HYQ Query Language. Technical Report Version 1.1, IBM Corporation, August 1993.

[18] Yong Kyu Lee, Seong-Joon Yoo, Kyoungro Yoon, and P. Bruce Berra. Index structures for structured documents. In *Proceedings of the first ACM international conference on Digital libraries*, pages 91–99. ACM Press, 1996.

[19] Quanzhong Li and Bongki Moon. Indexing and querying XML data for regular path expressions. In *Proceedings of VLDB*, pages 361–370, 2001.

[20] M. Murata, D. Lee, and M. Mani. “Taxonomy of XML Schema Languages using Formal Language Theory”. In *Extreme Markup Languages*, Montreal, Canada, August 2001.

[21] Adam Silberstein, Hao He, Ke Yi, and Jun Yang. BOXes: Efficient maintenance of order-based labeling for dynamic XML data. In *the 21st International Conference on Data Engineering (ICDE)*, 2005.

[22] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Co., Boston, Massachusetts, 1997.

[23] Igor Tatarinov, Stratis D. Viglas, Kevin Beyer, Jayavel Shanmugasundaram, Eugene Shekita, and Chun Zhang. Storing and querying ordered XML using a relational database system. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 204–215. ACM Press, 2002.

[24] W3C Recommendation. XML Path Language (XPath) Version 1.0. *http://www.w3.org/TR/xpath*, November 1999.

[25] W3C Working Draft. XQuery 1.0: An XML Query Language. *http://www.w3.org/TR/2002/WD-xquery-20021115*, November 2002.

[26] Xiaodong Wu, Mong Li Lee, and Wynne Hsu. A Prime Number Labeling Scheme for Dynamic Ordered XML Trees. In *To appear in Proceedings of the twentieth International Conference on Data Engineering*, 2004.

[27] Masatoshi Yoshikawa, Toshiyuki Amagasa, Takeyuki Shimura, and Shunsuke Uemura. XRel: a path-based approach to storage and retrieval of xml documents using relational databases. *ACM Transactions on Internet Technology (TOIT)*, 1(1):110–141, 2001.

Establishing an XML Metadata Knowledge Base to Assist Integration of Structured and Semi-structured Databases

Fahad M. Al-Wasil W. A. Gray N. J. Fiddian

Department of Computer Science
Cardiff University
Wales, UK

{Wasil, W.A.Gray, N.J.Fiddian}@cs.cardiff.ac.uk

Abstract

This paper describes the establishment of an XML Metadata Knowledge Base (XMKB) to assist integration of distributed heterogeneous structured data residing in relational databases and semi-structured data held in well-formed XML documents (XML documents that conform to the XML syntax rules but have no referenced DTD or XML schema) produced by internet applications. We propose an approach to combine and query the data sources through a mediation layer. Such a layer is intended to establish and evolve an XMKB incrementally to assist the Query Processor to mediate between user queries posed over the master view and the distributed heterogeneous data sources. The XMKB is built in bottom-up fashion by extracting and merging incrementally the metadata of the data sources. The XMKB is introduced to maintain the data source information (names, types and locations), meta-information about relationships of paths among data sources, and function names for handling semantic and structural discrepancies. A System to Integrate Structured and Semi-structured Databases (SISSD) has been built that generates a tool for a meta-user (who does the metadata integration) to describe mappings between the master view and local data sources by assigning index numbers and specifying conversion function names. This system is flexible: users can get any master view from the same set of data sources depending on their interest. It also preserves local autonomy of the local data sources. The SISSD uses the local-as-view approach to map between the master view and the local schema structures. This approach is well-suited to supporting a dynamic environment, where data sources can be added to or removed from the system without the need to restructure the master view and to regenerate the XMKB from scratch.

Keywords: Well-formed XML document, Relational database, XML queries, semi-structured data, semantic mapping, data integration.

1 Introduction

With the growth, widespread and increasing popularity of the Internet the number of data sources available for public access is rapidly increasing both in number and size, while at the same time users and application programs increasingly need to combine data from these different autonomous and heterogeneous data sources (Segev and Chatterjee, December 1991.) (Karunaratna et al., May 1998). However, for the foreseeable future, most data will continue to be stored in relational database systems because of the reliability, scalability, tools and performance associated with these systems (Funderburk et al., 2002) (Shanmugasundaram et al., September 2000). Additionally, many web-based applications and web services publish their data using XML (Lehti and Fankhauser, 2004), therefore much interesting and useful data can be found in well-formed XML documents. Hence, building a data integration system that provides unified access to semantically and structurally diverse data sources is very desirable to link structured data held in relational databases and semi-structured data in XML documents (Gardarin et al., 1999, Lee et al., 2002).The data integration system has to find structural transformations and semantic mappings that result in correct merging of the data and allow users to query the so-called mediated schema (Kurgan et al., 2002). This linking is a challenging problem since the pre-existing databases concerned are typically autonomous and located on heterogeneous hardware and software platforms. In this context, it is necessary to resolve several conflicts caused by the heterogeneity of the data sources with respect to data model, schema or schema concepts. Therefore, the mapping between entities from different sources representing the same real-world objects has to be defined. The main difficulty is that the data at different sources may be represented in different formats and in incompatible ways. For example, the bibliographical databases of different publishers may use different formats for authors' or editors' names (e.g., full name or separated first name and last name), or different units for prices. Moreover, the same expression may have a different meaning, or the same meaning may be specified by different expressions. This implies that syntactical data and metadata can not provide enough semantics for all potential integration purposes. As a result, the data integration process is often very labour-intensive and demands more computing expertise than most application users have. Therefore, semi-automated approaches seem the most promising, where mediation

Copyright (c) 2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

engineers are given an easy tool to describe mappings between the integrated (integrated and master are used interchangeably in this paper) schema and local schemas, to produce a uniform view over the local databases (Young-Kwang et al., October 2002).

XML is becoming the standard format to exchange information over the internet. The advantages of XML as an exchange model, such as rich expressiveness, clear notation and extensibility, make it the best candidate to be a data model for the integrated schema. As the importance of XML has increased, a series of standards has grown up around it, many of which were defined by the World Wide Web Consortium (W3C). For example, the XML Schema language provides a notation for defining new types of XML elements and XML documents. XML with its self-describing hierarchical structure and the language XML Schema provide the flexibility and expressive power needed to accommodate distributed and heterogeneous data. At the conceptual level, they can be visualized as trees or hierarchical graphs.

This paper mainly refers to the problem of integrating distributed heterogenous structured data residing in relational databases with semi-structured data held in well-formed XML documents (that conform to the XML syntax rules but have no referenced DTD or XML schema) produced by internet applications. These XML documents can be XML files on local hard drives or remote documents on web servers. We propose an approach to combine and query the data sources through a mediation layer. Such a layer is intended to establish and evolve an XML Metadata Knowledge Base (XMKB) incrementally to assist the Query Processor in mediating between user queries posed over the master view and the distributed heterogeneous data sources, to translate such queries into sub-queries -called local queries- which fit each local data source, and to integrate the results. The XMKB is built in a bottom-up fashion by extracting and merging incrementally the metadata of the data sources. The XMKB is an XML document which includes the database or XML document name, type and location information, and the metadata, in which the mappings between the master view and schema structures of the data sources are defined. A System to Integrate Structured and Semi-structured Databases (SISSD) has been built that generates a tool for meta-users to do the metadata integration, producing an XML Metadata Knowledge Base (XMKB), which is then used to generate queries to local data sources from user queries posed over the master view, and to integrate the results. This tool parses the master view to generate automatically an index number for each element and parses local schema structures to generate a path for each element, and produce a convenient GUI. The mappings assign indices to match local elements to corresponding master elements and to names of conversion functions. These functions can be built-in or user-defined functions. The XMKB is then generated based on the mappings by combination over index numbers. User queries are expressed in FLWR expressions of XQuery (a powerful universal query language for XML) and processed according to the XMKB, by generating an executable query for each relevant local data source.

This system is flexible: users can get any virtual master view they want from the same set of data sources depending on their interest. It also preserves local autonomy of the local data sources, thus any data sources can be handled without rebuilding or modification. The SISSD uses the local-as-view approach to map between the master view and the local schema structures. This approach is well-suited to supporting a dynamic environment, where data sources can be added to or removed from the system without the need to restructure the master view. The XML Metadata Knowledge Base (XMKB) is evolved and modified incrementally when any data sources are added to or removed from the system without the need to regenerate it from scratch.

The rest of the paper is organized as follows. The next section presents related work. The architecture and its main components are described in section 3. Section 4 presents the structure, content and the organization of knowledge in the XMKB and how it is generated. Finally, we present conclusions in section 5.

2 Related Work

Data integration has received significant attention since the early days of databases. In recent years, there have been several projects focusing on heterogeneous information integration. Most of them are based on a common mediator architecture (Wiederhold, March 1992). In this architecture, mediators provide a uniform user interface to query integrated views of heterogeneous data sources. They resolve queries over global concepts into sub-queries over data sources. Mainly, they can be classified into structural approaches and semantic approaches.

In structural approaches, local data sources are assumed to be crucial. The integration is done by providing or automatically generating a global unified schema that characterizes the underlying data sources. On the other hand, in semantic approaches, integration is achieved by sharing a common ontology among the data sources. According to the mapping direction, the approaches are further classified into two categories: global-as-view and local-as-view (Lenzerini, 2002). In global-as-view approaches, each item in the global schema is defined as a view over the source schemas. In local-as-view approaches, each item in each source schema is defined as a view over the global schema. The local-as-view approach is well-suited to supporting a dynamic environment, where data sources can be added to or removed from the data integration system without the need to restructure the global schema.

There are several well-known research projects and prototypes such as Garlic (Carey et al., 1995), Tsimmis (Ullman, 1997), MedMaker (Papakonstantinou et al., 1996) and Mix (Baru et al., 1999) which take a structural and global-as-view approach. A common data model is used, e.g., OEM (Object Exchange Model) in Tsimmis and MedMaker. Mix uses XML as the data model; an XML query language XMAS was developed and used as

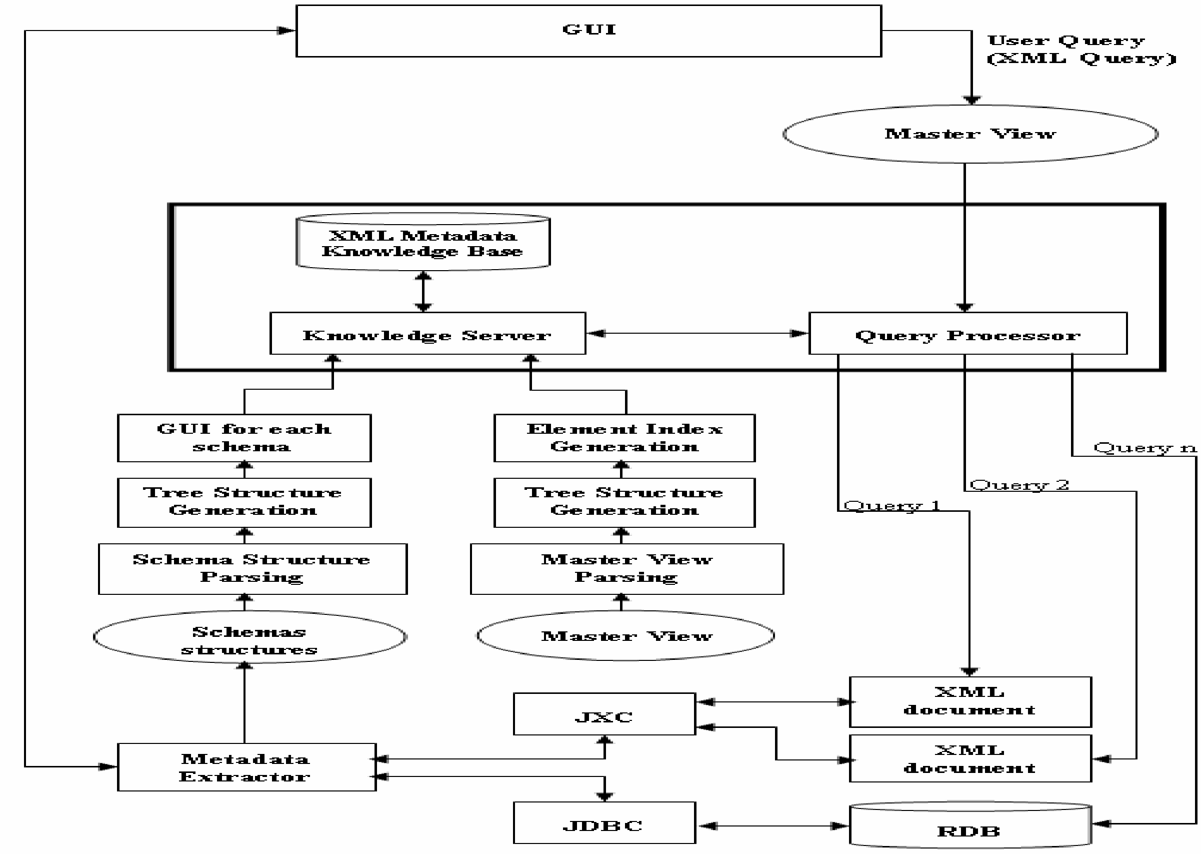


Figure 1: The Architecture of our System.

the view definition language there. DDXMI (Young-Kwang et al., October 2002, Nam et al., 2003) (for Distributed Database XML Metadata Interface) builds on XML Metadata Interchange. DDXMI is a master file including database information, XML path information (a path for each node starting from the root), and semantic information about XML elements and attributes. A system prototype has been built that generates a tool to do the metadata integration, producing a master DDXMI file, which is then used to generate queries to local databases from master queries. In this approach local sources were designed according to DTD definitions. Therefore, the integration process is based on the DTD parsing that is associated with each source. (Almarimi and Pokorny, 2004) describe an approach for mediation of heterogeneous XML data sources. Their approach is proposed as a tool for an XML data integration system to combine and query XML documents through a mediation layer. This layer is intended to describe the mappings between the global XML schema and local heterogeneous XML schemas. It produces a uniform interface over the local XML data sources and provides the required functionality to query these sources in a uniform way. It involves two important units: the XML Metadata Document (XMD) and the Query Translator. The XMD is an XML document containing metadata, in which the mappings between global and local schemas are defined. The XML Query Translator which is an integral part of the system is introduced to translate a global user query into local queries by using the mappings that are defined in the XMD. In this case the XML data sources are described by the XML Schema language.

We classify our work as being in the structural category but we differ from the others (Ullman, 1997, Papakonstantinou et al., 1996, Baru et al., 1999) by following the local-as-view approach. The XML documents that we are interested in are well-formed XML documents which have no referenced DTD or XML schema, while the work in (Young-Kwang et al., October 2002, Nam et al., 2003) is interested in XML documents designed according to DTD definitions, and that in (Almarimi and Pokorny, 2004) is interested in XML documents satisfying different XML schemas. Also our work differs from the others (Young-Kwang et al., October 2002; Nam et al., 2003; Almarimi and Pokorny, 2004) by using an incremental tool to build the XML Metadata Knowledge Base (XMKB). This tool would start from the previous XMKB file and slightly modify it in light of slight modifications to data sources schema structure or when any data sources are added to or removed from the system, instead of regenerating it from scratch.

3 The SISSD Architecture and Components

In this section, we present an overview of the SISSD architecture and summarize the functions of the main components. The architecture we adopt is depicted in Figure 1. Its main components are the Metadata Extractor (MDE), the Knowledge Server (KS) and the Query Processor (QP).

3.1 Metadata Extractor (MDE)

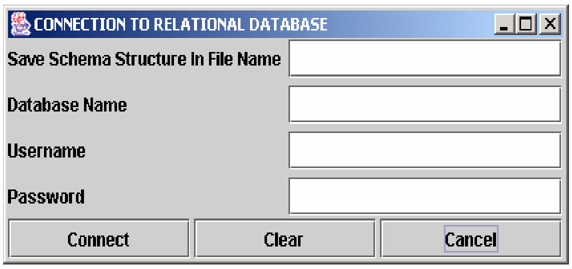
The MDE needs to deal with heterogeneity at the hardware, software and data model levels without violating the local autonomy of the data sources. It interacts with the data sources via JDBC (Java Database Connectivity) if the data source is a relational database or via JXC (Java XML Connectivity) if the data source is an XML document. The MDE extracts the metadata of all data sources and builds a schema structure in XML form for each data source.

We have developed JXC using a JDOM (Java Document Object Model) interface to detect and extract the schema structure of a well-formed XML document (that conforms to the XML syntax rules but has no referenced DTD or XML schema) where the metadata are buried inside the data.

3.1.1 Schema Structures

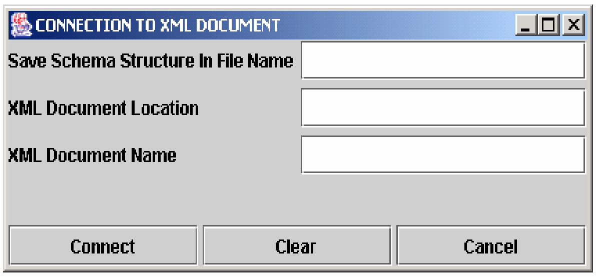
Typically, the heterogeneous data sources use different data models to store the data (e.g. relational model and XML model). This type of heterogeneity is referred to as syntactic heterogeneity. The solution commonly adopted to overcome syntactic heterogeneity is to use a common data model and to map all schemas to this common model. The advantages of XML as an exchange model, make it a good candidate to be the common data model and for supporting the integrated data model. The metadata extracts generated on top of the data sources by using this data model are referred to as schema structures. We define a simple XML Data Source Definition Language (XDSDL) for describing and defining the relevant identifying information and the data structure of a data source. The XDSDL is represented in XML and is composed of two parts. The first part provides a description of the data source name, location and type (relational database or XML document). The second part provides a definition and description of the data source structure and content. The emphasis is on making these descriptions readable by automated processors such as parsers and other XML-based tools. This language can be used for describing the structure and content of relational databases and well-formed XML documents which have no referenced DTD or XML schema.

For relational databases the MDE employs JDBC to access the DB without making any changes to it. The MDE accepts the information necessary to establish a connection to a DB to retrieve the metadata of its schema and uses the XDSDL to build the target schema structure for that DB, together with necessary information such as the DB location (URL), where to save the schema structure, the User ID and Password.


CONNECTION TO RELATIONAL DATABASE
Save Schema Structure In File Name
Database Name
Username
Password
Connect
Clear
Cancel

It opens a connection to that DB through a JDBC driver. Opening this connection enables SQL queries to be issued to and results to be retrieved from the DB. Once the connection is established, the MDE retrieves the names of all the tables defined in the accessed DB schema and then uses the XDSDL to define these tables as elements in the target schema structure. Furthermore, for each table the MDE extracts and analyses the attribute names, then defines these attributes as child elements for that table element in the target schema structure using the XDSDL.

For XML documents the MDE employs JXC to access the document without making any changes to it. The MDE accepts the information necessary to establish a connection to a well-formed XML document to retrieve the metadata of its schema where the metadata are buried inside the data. It then uses the XDSDL to build the target schema structure for that XML document, together with necessary information such as the document location (URL), where to save the schema structure, and the document name.


CONNECTION TO XML DOCUMENT
Save Schema Structure In File Name
XML Document Location
XML Document Name
Connect
Clear
Cancel

It opens a connection to that XML document through a JDOM interface. Once the connection is established, the JXC automatically tracks the structure of the XML document, viz. each element found in the document, which elements are child elements and the order of child elements. The JXC reads the XML document and detects the start tag for the elements. For each start tag, the JXC checks if this element has child elements or not: if it has then this element is defined as a complex element in the target schema structure using the XDSDL, otherwise it is defined as a simple element by the MDE. The defined elements in the target schema structure take the same name as the start tags.

3.2 Knowledge Server (KS)

The Knowledge Server (KS) is the central component of the SISSD. Its function is to establish, evolve and maintain the XML Metadata Knowledge Base (XMKB), which holds information about the data sources and provides the necessary functionality for its role in assisting the Query Processor (QP). The KS generates a tool for meta-users to do metadata integration by building the XML Metadata Knowledge Base (XMKB) that comprises information about data structures and semantics. This can then be used by the Query Processor (QP) to automatically rewrite a user query over the master view into sub-queries called local queries, fitting each local data source, and to integrate the results.

3.3 Query Processor (QP)

The Query Processor receives a user query over the master view to process it and returns the query result to the user in integrated form. User queries are expressed in XQuery (a powerful universal query language for XML) using FLWR expressions. XQuery offers seven types of expression, but FLWR expressions are among the most interesting types of expression it offers. Using these expressions for queries over the master view makes it easy to translate the sub-queries directed at relational databases into SQL queries since syntactically, FLWR expressions look similar to SQL select statements and have similar capabilities, only they use path expressions instead of table and column names.

The Query Processor is composed of several components in charge of:

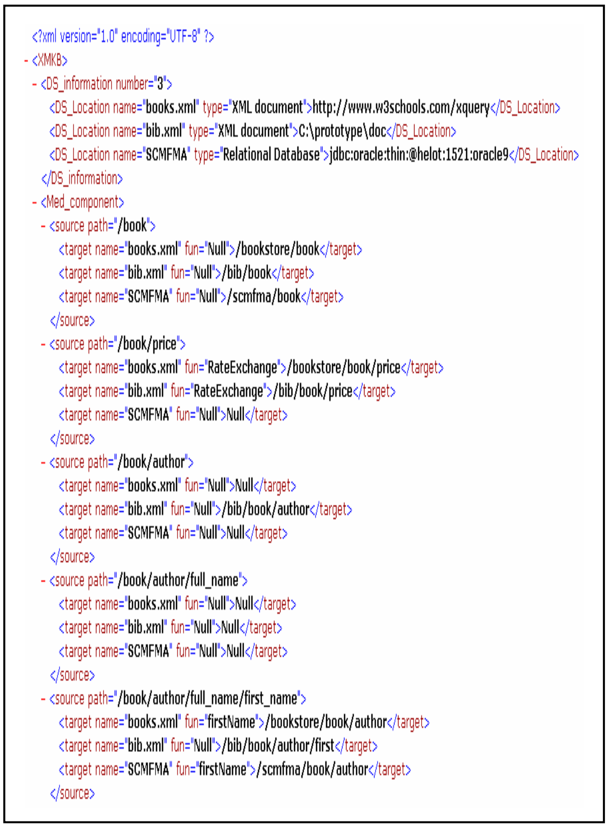
- Rewriting the user query into sub-queries -called local queries- which fit each local data source, by using the mapping information stored in the XMKB.
- Converting the XQuery sub-queries addressed to the relational databases into SQL queries to execute them, then converting the results into XML format.
- Sending local queries to their corresponding local data source engine, to process the query and return the results.
- Merging the results of the sub-queries.

4 The XML Metadata Knowledge Base (XMKB)

The building of the XML Metadata Knowledge Base (XMKB) is performed through a semi-automatic process. The XMKB is generated based on mappings between the master view and the local schemas, and includes the data source information (names, types and locations), XML path information (a path for each node starting from the root), and semantic information about XML elements (function names to resolve structural and semantic conflicts).

4.1 The Structure of XMKB

The XML Metadata Knowledge Base (XMKB) is an XML document composed of two parts. The first part contains information about data source name, type and location. The second part contains meta-information about relationships of paths among data sources, and function names for handling semantic and structural discrepancies. The XMKB structure with its schema is shown in Figures 2 and 3, respectively. The *DS_information* element in Figure 2 contains data source names, types and locations. The *DS_information* element has one attribute called *number* which holds the number of data sources present in the integration system (3 in the example shown). Also the *DS_information* element has child elements called *DS_Location* elements. Each *DS_Location* element contains the data source name, its type (relational database or XML document) as an attribute value and the location of the data source as an element value. This information is used by the Query Processor to specify the type of generated sub-query (SQL if the data source type is relational database, or XQuery if the data source type is XML document) and the data source location that the system will submit the generated sub-query to. The *Med_component* element in Figure 2 contains the mappings between the master view elements and the local data source elements, and the function names for handling semantic and structural discrepancies. The master view elements are called *source* elements, while corresponding elements in local data sources are called *target* elements. The *source* elements in the XMKB document have one attribute called *path* which contains the path of the master view elements. Also the source elements in this document have child elements called *target* which contain the corresponding path for the master view elements in each local data source, or null if there is no corresponding path. The *target* elements in the XMKB document have two attributes. The first one is called *name* and contains the name of the local data source, while the second is called *fun* and contains the function name that is needed to resolve semantic and structural discrepancies between the master view element and the local data source element concerned.



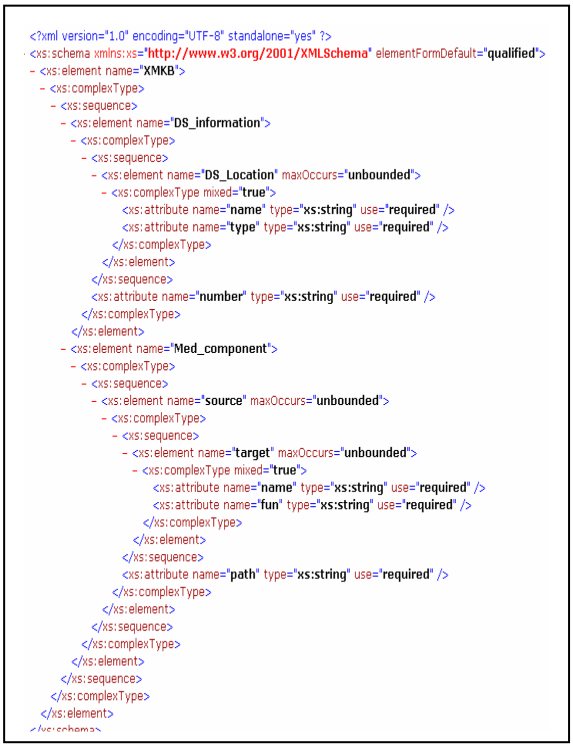
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <XMKB>
  - <DS_information number="3">
      <DS_Location name="books.xml" type="XML document">http://www.w3schools.com/xquery</DS_Location>
      <DS_Location name="bib.xml" type="XML document">C:\prototype\doc</DS_Location>
      <DS_Location name="SCMFMA" type="Relational Database">jdbc:oracle:thin:@helot:1521:oracle9</DS_Location>
    </DS_information>
  - <Med_component>
    - <source path="/book">
        <target name="books.xml" fun="Null">/bookstore/book</target>
        <target name="bib.xml" fun="Null">/bib/book</target>
        <target name="SCMFMA" fun="Null">/scmfma/book</target>
      </source>
    - <source path="/book/price">
        <target name="books.xml" fun="RateExchange">/bookstore/book/price</target>
        <target name="bib.xml" fun="RateExchange">/bib/book/price</target>
        <target name="SCMFMA" fun="Null">Null</target>
      </source>
    - <source path="/book/author">
        <target name="books.xml" fun="Null">Null</target>
        <target name="bib.xml" fun="Null">/bib/book/author</target>
        <target name="SCMFMA" fun="Null">Null</target>
      </source>
    - <source path="/book/author/full_name">
        <target name="books.xml" fun="Null">Null</target>
        <target name="bib.xml" fun="Null">Null</target>
        <target name="SCMFMA" fun="Null">Null</target>
      </source>
    - <source path="/book/author/full_name/first_name">
        <target name="books.xml" fun="firstName">/bookstore/book/author</target>
        <target name="bib.xml" fun="Null">/bib/book/author/first</target>
        <target name="SCMFMA" fun="firstName">/scmfma/book/author</target>
      </source>
```

Figure 2: A sample XMKB document.

4.2 How to Generate an XMKB

The XML Metadata Knowledge Base (XMKB) is utilized in mediation to overcome the heterogeneity of data sources. XMKB is intended to maintain the correspondence between the components of the data sources. For each component of the master view, the objective is to record the set of components having the same meaning in the local schema structures and the discrepancy resolution function if it is needed.

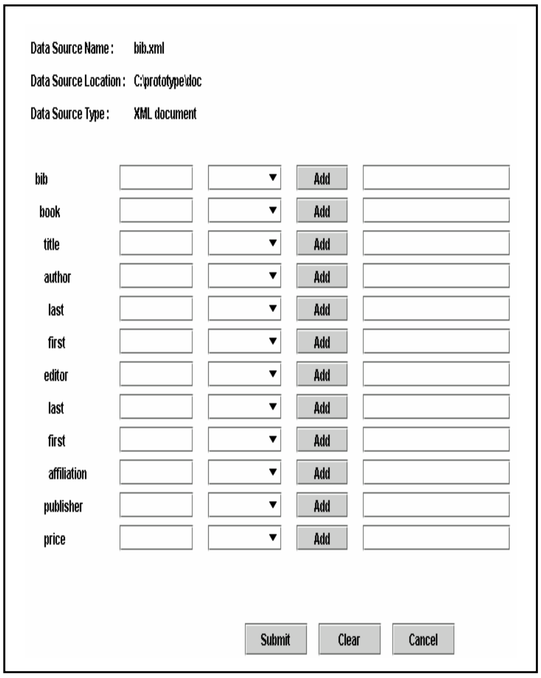
Each data source (relational database or well-formed XML document) has its own schema structure in XML format constructed by the Meta-data Extractor (MDE). We assume that elements in local data sources do not contain attributes. This implies that data source schema structures can be represented as n-ary trees. Our approach involves mapping paths in the master view to (sets of) paths in the local schema structures, though we often speak of elements instead of the paths that lead to these elements. We match an element in the master view with elements in local data source schema structures, through generating an index number for each element in the master view tree and then assigning these index numbers to the element(s) with the same meaning in the local schema structure trees. Hence elements with the same number have the same meaning. By collecting all elements with the same numbers, the source and target paths can be generated automatically, and the XMKB can be easily constructed. An especially convenient special case is where an element in the master view exactly matches one in a local schema structure, in that its field has the same meaning as the one in the master view. Elements in local schema structures should not appear in the XMKB file if their meaning does not relate to any element in the master view.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
- <xs:element name="XMKB">
  - <xs:complexType>
    - <xs:sequence>
      - <xs:element name="DS_information">
        - <xs:complexType>
          - <xs:sequence>
            - <xs:element name="DS_Location" maxOccurs="unbounded">
              - <xs:complexType mixed="true">
                  <xs:attribute name="name" type="xs:string" use="required" />
                  <xs:attribute name="type" type="xs:string" use="required" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="number" type="xs:string" use="required" />
          </xs:complexType>
        </xs:element>
      - <xs:element name="Med_component">
        - <xs:complexType>
          - <xs:sequence>
            - <xs:element name="source" maxOccurs="unbounded">
              - <xs:complexType>
                - <xs:sequence>
                  - <xs:element name="target" maxOccurs="unbounded">
                    - <xs:complexType mixed="true">
                        <xs:attribute name="name" type="xs:string" use="required" />
                        <xs:attribute name="fun" type="xs:string" use="required" />
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name="path" type="xs:string" use="required" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 3: An XML schema of an XMKB document.

Constructing an XMKB file manually is an error prone and tedious job, so that machine support is highly desirable. Hence, we have developed a system that constructs an XMKB automatically. We implement a simple form (GUI) -from parsing a local schema structure- as an assistant tool for mapping generation. For example, Figure 4 presents part of a GUI for the local schema structure shown in Figure 5.



Data Source Name : bib.xml

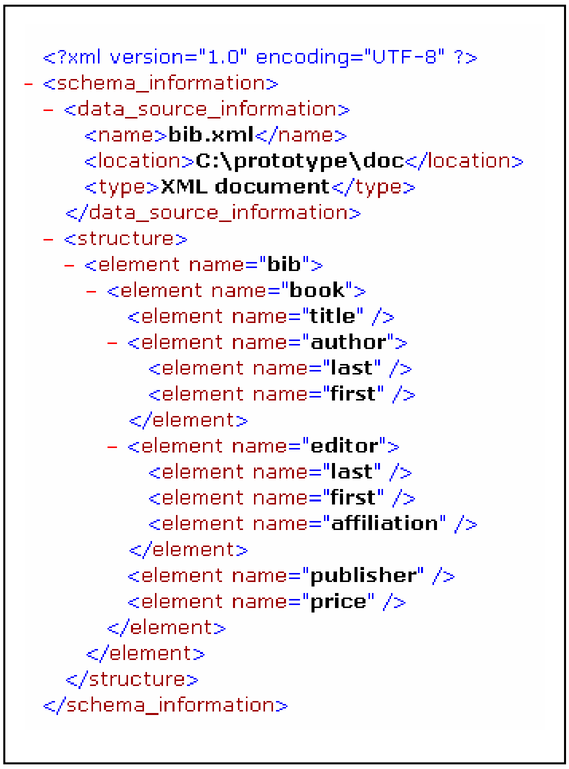
Data Source Location : C:\prototype\doc

Data Source Type : XML document

bib
book
title
author
last
first
editor
last
first
affiliation
publisher
price

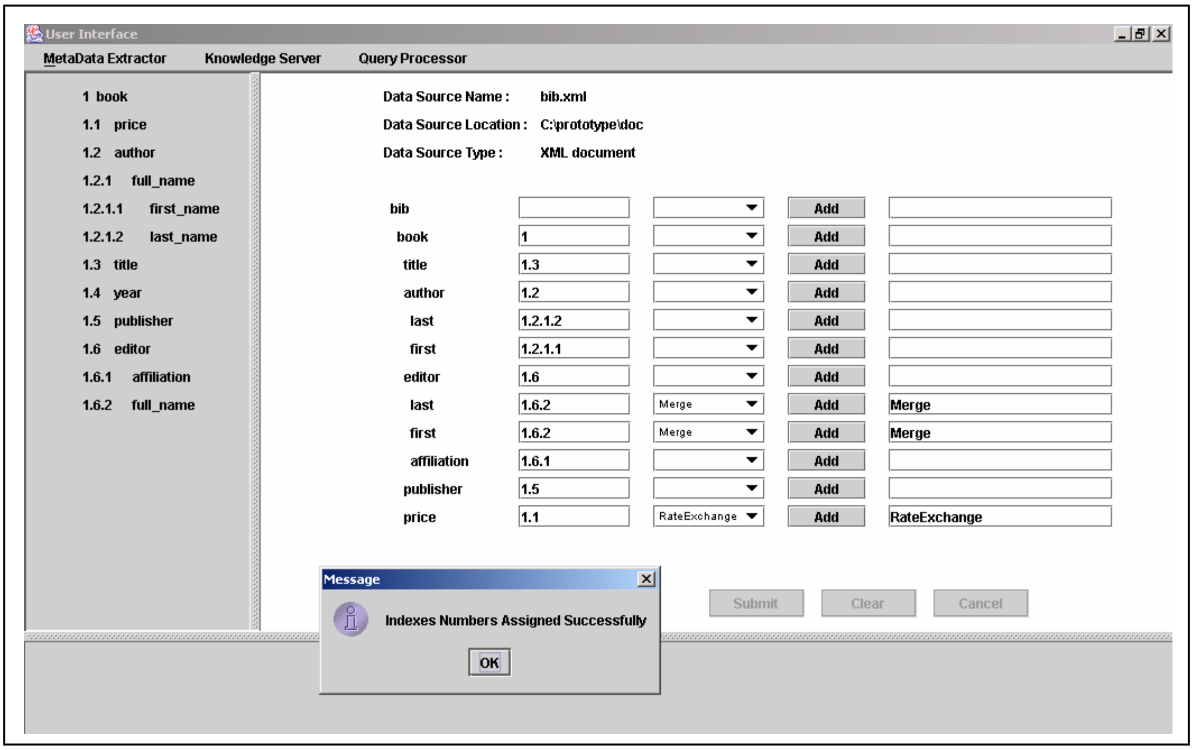
Figure 4: A GUI for local schema structure of the bib XML document.

The first column is used for assigning the unique index numbers of master view elements to the equivalent elements in the local schema structure. Elements without an equivalent index number are not included in the XMKB document. The second column is used to specify the function names which are needed to resolve heterogeneity conflicts by performing specific operations.



```
<?xml version="1.0" encoding="UTF-8" ?>
- <schema_information>
  - <data_source_information>
      <name>bib.xml</name>
      <location>C:\prototype\doc</location>
      <type>XML document</type>
    </data_source_information>
  - <structure>
    - <element name="bib">
      - <element name="book">
          <element name="title" />
        - <element name="author">
            <element name="last" />
            <element name="first" />
          </element>
        - <element name="editor">
            <element name="last" />
            <element name="first" />
            <element name="affiliation" />
          </element>
          <element name="publisher" />
          <element name="price" />
        </element>
      </element>
    </structure>
</schema_information>
```

Figure 5: Schema structure of the bib XML document.



1 book
1.1 price
1.2 author
1.2.1 full_name
1.2.1.1 first_name
1.2.1.2 last_name
1.3 title
1.4 year
1.5 publisher
1.6 editor
1.6.1 affiliation
1.6.2 full_name

Data Source Name : bib.xml
Data Source Location : C:\prototype\doc
Data Source Type : XML document

bib			Add	
book	1		Add	
title	1.3		Add	
author	1.2		Add	
last	1.2.1.2		Add	
first	1.2.1.1		Add	
editor	1.6		Add	
last	1.6.2	Merge	Add	Merge
first	1.6.2	Merge	Add	Merge
affiliation	1.6.1		Add	
publisher	1.5		Add	
price	1.1	RateExchange	Add	RateExchange

Indexes Numbers Assigned Successfully

Figure 6: Example of mapping between master view and XML document.

The process of XMKB generation comprises the following steps:

1. Automatically generate unique index numbers for the master view elements.
2. Produce a convenient GUI for each local schema structure.
3. Using the GUI for each local schema structure, the unique index numbers of the master view elements are assigned to the equivalent local schema structure elements. Figure 6 shows an example of mapping between a master view (on the left of the figure) and the schema structure of the bib XML document shown in Figure 5.
4. Use the second column of the GUI form to specify the function names which are needed to resolve any heterogeneity conflicts by performing specific operations.
5. After assigning index numbers and function names, mapping paths in the master view to (sets of) paths in the local schema structures are generated for each element starting from the root, see Figure 7.
6. Finally, data source information (name, type and location), mapping paths and function names needed to resolve semantic and structural conflicts are merged with the XML Metadata Knowledge Base (XMKB) based on the mappings by collecting elements with the same index numbers.

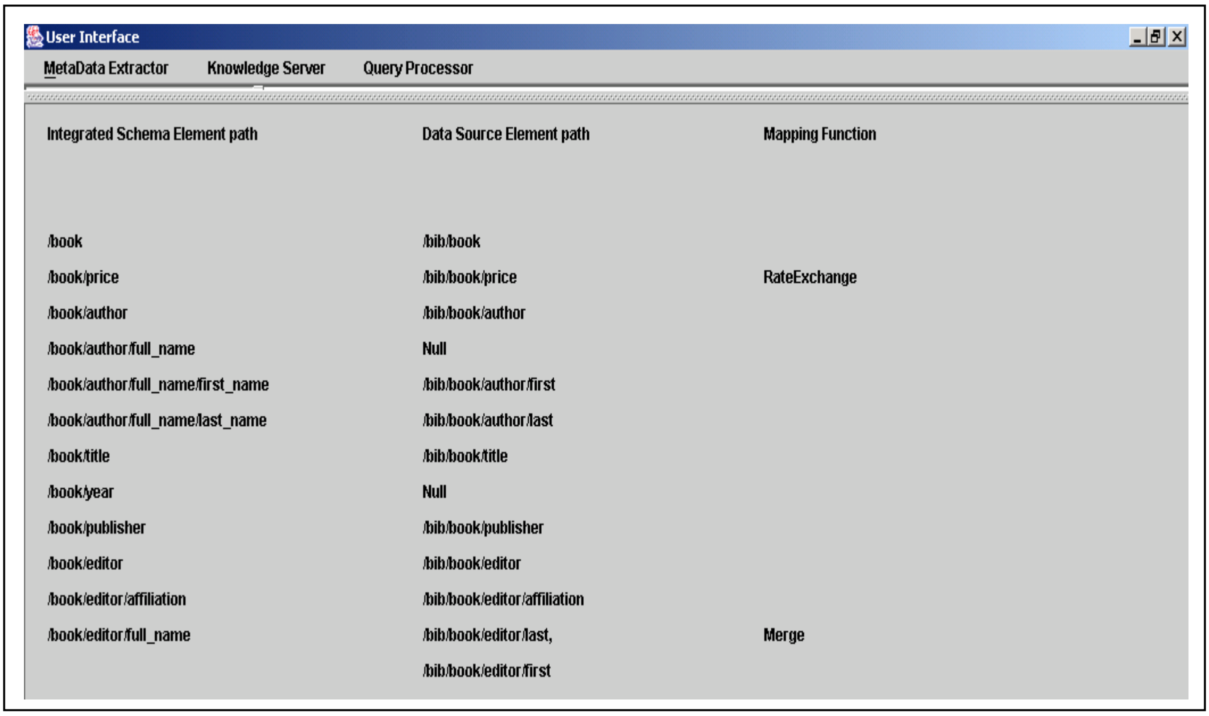
This approach provides a flexible environment able to accommodate the continual change and update of data source schemas, especially suitable for XML documents on web servers since these remote documents are not static and are often subject to frequent update. The SISSD gives the flexibility to remove any data source schema from the XMKB and then add this data source again with an updated or altered schema without any other impact on the XMKB or the need to regenerate it from scratch.

4.2.1 Index number generation for the master view elements

The generated index numbers for the master view elements are used to match local elements to corresponding master elements. We employ a mechanism to generate such index numbers using JDOM technology. By applying this mechanism, a unique index number is generated for each element in the XML document whatever the nesting complexity of the document.

The process of generating the index numbers comprises the following steps:

1. First, use JDOM technology to read and parse the XML document (master view) and map it to a tree.
2. Identify the root element of the document and assign index number 1 to it.
3. For each element in the document (including the root element), get all the children of this element.



Integrated Schema Element path	Data Source Element path	Mapping Function
/book	/bib/book	
/book/price	/bib/book/price	RateExchange
/book/author	/bib/book/author	
/book/author/full_name	Null	
/book/author/full_name/first_name	/bib/book/author/first	
/book/author/full_name/last_name	/bib/book/author/last	
/book/title	/bib/book/title	
/book/year	Null	
/book/publisher	/bib/book/publisher	
/book/editor	/bib/book/editor	
/book/editor/affiliation	/bib/book/editor/affiliation	
/book/editor/full_name	/bib/book/editor/last, /bib/book/editor/first	Merge

Figure 7: Generated mapping paths.

4. Assign a sequential number starting from 1 for each child to represent the order of children for that parent.

5. Combine this number given for the child with the index number of the parent separated by dot (.) and this will be the index number of that child. For example, if the root element has four child elements, the index number of the first child element will be 1.1, the index number of the second child element will be 1.2, and so on.

4.2.2 Mapping between elements

According to the number of elements that are involved in the master view and a local schema structure, mappings between them are classified as One-to-One, One-to-Many or Many-to-One. For example, a local data source may represent author names as full names, while the master view separates the first and last names. In this case, the answer from the local data source must be split up if a query is to retrieve the first name of the author. Several mapping cases are possible in which such conflicts may occur between elements. In the next subsections, we describe some cases.

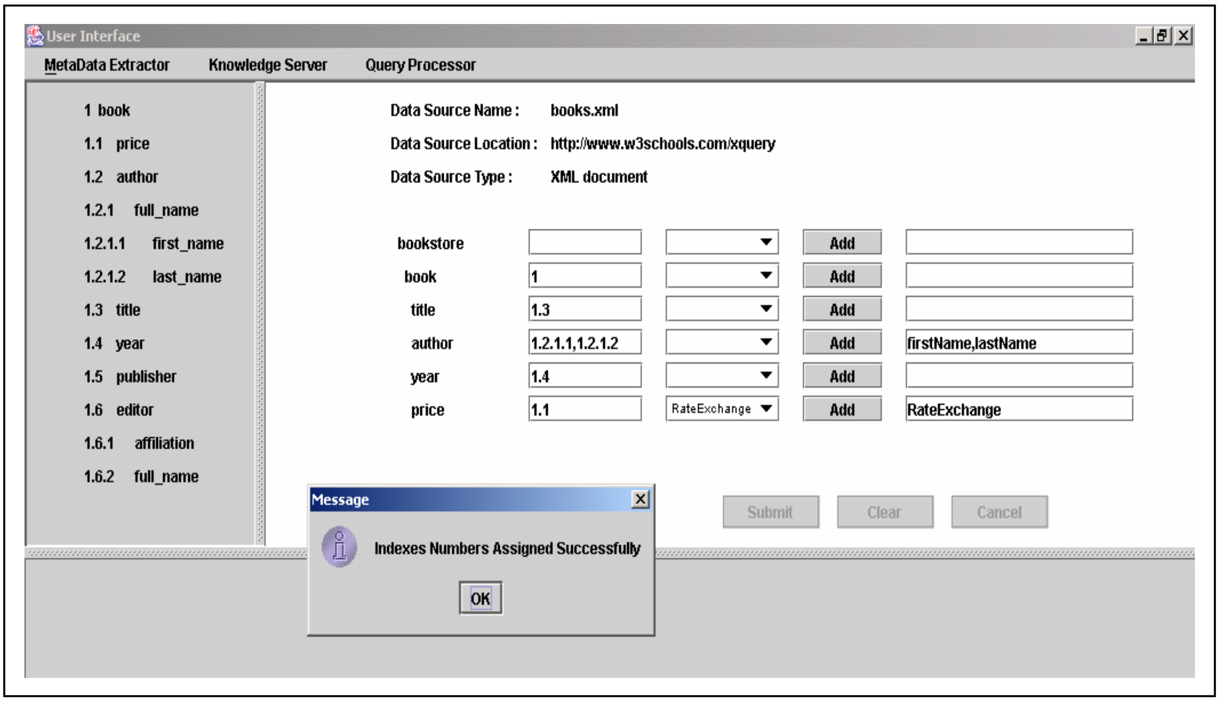
- **One to One with semantic functions:** this can occur when an element in the master view matches one element in a local schema structure but they use different reference systems. For example, the master view may represent price elements in dollar currency, while the local data source uses sterling currency or represents prices in cents. Therefore to resolve this conflict some conversion mechanism is required to translate between such representations.

- **One-element to Many-elements:** this case can occur when there is one element in the master view mapped to many elements in a local schema structure. Hence, more than one element in the local schema structure has the same index number. For example, the master view may represent an editor name as a full name, while the local data source separates an editor's first and last names. Therefore to resolve this conflict we need a function to concatenate the first and the last name elements to get the full name.

- **Many-elements to One-element:** this case can occur when there is more than one element in the master view corresponding to one element in a local schema structure. Hence, the element in the local schema structure will have more than one index number and more than one function name. For example, the master view may represent an author name as first_name and last_name, while the local data source represents it as a full name. Therefore to resolve this conflict two functions, *firstName* and *lastName*, are needed to split the author's full name into separate first name and last name. Figure 8 shows an example of Many-elements to One-element mapping and how this is done inside the GUI.

5 Conclusions

In this paper, we have described an approach for establishing an XML Metadata Knowledge Base (XMKB) to resolve structural and semantic conflicts between distributed heterogeneous structured data residing in relational databases and semi-structured data



1 book
1.1 price
1.2 author
1.2.1 full_name
1.2.1.1 first_name
1.2.1.2 last_name
1.3 title
1.4 year
1.5 publisher
1.6 editor
1.6.1 affiliation
1.6.2 full_name

Data Source Name : books.xml
Data Source Location : http://www.w3schools.com/xquery
Data Source Type : XML document

bookstore			
book	1		
title	1.3		
author	1.2.1.1,1.2.1.2		firstName,lastName
year	1.4		
price	1.1	RateExchange	RateExchange

Message
Indexes Numbers Assigned Successfully

Figure 8: Example of Many-elements to One-element mapping.

held in well-formed XML documents produced by internet applications. The XMKB is employed to maintain the data source information (names, types and locations), meta-information about relationships of paths among data sources, and function names for handling semantic and structural discrepancies. A GUI tool is generated automatically for a meta-user (who does the meta-data integration) to describe mappings between the master view and local data sources by assigning a unique index number generated automatically for master view elements to the element(s) with the same meaning in the local schema structures, and specifying any necessary conversion functions for resolving structural and semantic conflicts. The XMKB is built based on these mappings by collecting element paths with the same index numbers to contain information about data structures and semantics. It can then be used by the Query Processor (QP) to mediate between user queries posed over the master view and the distributed heterogeneous data sources, to translate such queries into sub-queries -called local queries- which fit each local data source, and to integrate the results of these sub-queries.

The SISSD has been developed using Java, JDOM, and the JavaCC compiler. We have implemented the Meta-data Extractor (MDE) using JDBC and JDOM technology. We use JDBC as the API to connect to a relational database system. As a result, our implementation works on top of most commercial database systems including DB2, Oracle and Microsoft SQL Server, and on most hardware platforms. We have developed JXC using a JDOM (Java Document Object Model) interface to detect and extract the schema structure of a well-formed XML document, where the metadata are buried inside the data. The SISSD also preserves local autonomy of the local data sources, thus any data sources can be handled without rebuilding or modifying the XMKB.

Certain issues remain to be investigated. For example, if some elements in the local data sources contain attributes and these attributes correspond to elements in the master view, how mapping between them would be achieved. This is not yet implemented, but should not be difficult. There are other matters, notably relating to the SISSD Query Processor, which have been outlined but not discussed in any great detail in this paper: it is intended that these will be elaborated in other, separate, publications.

6 References

W3C Consortium: Extensible Markup Language (XML). http://www.w3.org/TR/2000/REC-xml.

ALMARIMI, A. & POKORNY, J. (2004) A Mediation Layer for Heterogeneous XML Schemas. *Proceedings of the Sixth International Conference on Information Integration and Web Based Applications & Services (iiWAS2004).* Jakarta, Indonesia.

BARU, C., GUPTA, A., LUDÄSCHER, B., MARCIANO, R., PAPAKONSTANTINOU, Y., VELIKHOV, P. & CHU, V. (1999) XML-based information mediation with MIX. *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data.* ACM Press.

CAREY, M. J., PETKOVIC, D., THOMAS, J., WILLIAMS, J. H., WIMMERS, E. L., HAAS, L. M., SCHWARZ, P. M., ARYA, M., CODY, W. F., FAGIN, R., FLICKNER, M., LUNIEWSKI, A. W. & NIBLACK, W. (1995)

Towards heterogeneous multimedia information systems: the Garlic approach. *RIDE '95: Proceedings of the 5th International Workshop on Research Issues in Data Engineering-Distributed Object Management (RIDE-DOM'95).* IEEE Computer Society.

FUNDERBURK, J. E., KIERNAN., G., SHANMUGASUNDARAM, J., SHEKITA, E. & WEI, C. (2002) XTABLES: Bridging Relational Technology and XML. *IBM Systems Journal,* 41(4)**,** 616-641.

GARDARIN, G., SHA, F. & DANG-NGOC, T. (1999) XML-based Components for Federating Multiple Heterogeneous Data Sources. *ER '99: Proceedings of the 18th International Conference on Conceptual Modeling.* Springer-Verlag.

KARUNARATNA, D. D., GRAY, W. A. & FIDDIAN, N. J. (May 1998) Organising Knowledge of a Federated Database System to Support Multiple View Generation. *Proceedings of the 5th KRDB Workshop (Knowledge Representation meets Data Bases), pp. 12.1-12.10,* Seattle, Washington, USA.

KURGAN, L., SWIERCZ, W. & CIOS, K. (2002) Semantic Mapping of XML Tags using Inductive Machine Learning. *Proceedings of the International Conference on Machine Learning and Applications - ICMLA '02.* Las Vegas, Nevada, USA.

LEE, K., MIN, J. & PARK, K. (2002) A Design and Implementation of XML-Based Mediation Framework (XMF) for Integration of Internet Information Resources. *HICSS '02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 7.* IEEE Computer Society.

LEHTI, P. & FANKHAUSER, P. (2004) XML data integration with OWL: Experiences & challenges. *Proceedings of the International Symposium on Applications and the Internet (SAINT 2004).* Tokyo, Japan.

LENZERINI, M. (2002) Data integration: a theoretical perspective. *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of database systems.* Madison, Wisconsin.

NAM, Y.-K., GOGUEN, J. & WANG, G. (2003) A Metadata Tool for Retrieval from Heterogeneous Distributed XML Documents. *Proceedings of the International Conference on Computational Science, LNCS 2660, Springer, pp. 1020-1029.*

PAPAKONSTANTINOU, Y., GARCIA-MOLINA, H. & ULLMAN, J. D. (1996) MedMaker: A Mediation System Based on Declarative Specifications. *ICDE '96: Proceedings of the Twelfth International Conference on Data Engineering.* IEEE Computer Society.

SEGEV, A. & CHATTERJEE, A. (December 1991) Data manipulation in heterogeneous databases. *Sigmod Record,* 20(4)**,** 64-68.

SHANMUGASUNDARAM, J., SHEKITA, E. J., BARR, R., CAREY, M. J., LINDSAY, B. G., PIRAHESH, H. & REINWALD, B. (September 2000) Efficiently Publishing Relational Data as XML Documents. *Proceedings of the 26th International Conference on Very Large Databases, (VLDB2000).* Cairo, Egypt.

ULLMAN, J. D. (1997) Information Integration Using Logical Views. *ICDT '97: Proceedings of the 6th International Conference on Database Theory.* Springer-Verlag.

WIEDERHOLD, G. (March 1992) Mediators in the Architecture of Future Information System. *IEEE Computer,* 25(3)**,** 38-49.

YOUNG-KWANG, N., JOSEPH, G. & GUILIAN, W. (October 2002) A Metadata Integration Assistant Generator for Heterogeneous Distributed Databases. *Proceedings of the Confederated International Conferences DOA, CoopIS and ODBASE, Irvine CA, LNCS 2519, Springer, pp. 1332-1344.*

Peer-to-Peer Form Based Web Information Systems

Stijn Dekeyser[1] **Jan Hidders**[2] **Richard Watson**[1] **Ron Addie**[1]

[1] **University of Southern Queensland, Australia**
[2]**University of Antwerp, Belgium**

Abstract

The World Wide Web revolutionized the use of forms in everyday private and business life by allowing a move away from paper forms to easily accessible digital forms. Data captured using such HTML forms could be processed using relational databases or other applications that enforce and apply business logic. Lately XForms has been introduced, offering a logical evolution of digital data capture and dissemination using Internet and document technology.

This paper introduces two important new ideas. The first one is the main focus of the paper: a novel type of peer-to-peer web information system where forms are first-class citizens containing extended access rules of very fine granularity which govern read and update rights to data objects associated to the forms. The second idea, which we explore in a preliminary section, forms a powerful motivation for the use of such systems: the automatic and dynamic derivation of workflow processes from the access rules contained in forms.

As such, the proposed system leverages current forms and Internet technology to liberate the creation and use of forms and reports, facilitating the capture and dissemination of data, while allowing dynamic management of work flows within organizations.

1 Introduction

The background for the theory developed in this paper is a real-life problem. The Department of Mathematics & Computing at USQ would like its staff to easily capture data from colleagues and students through web-based forms, efficiently store that data, and re-use the data captured by others, without compromising security and access rights, and without staff having to script their own web pages with database functionality. In addition, while giving everyone the opportunity to create complex interactive data-driven applications, the system which allows this must be able to communicate with other such systems on different peers, and show end-users the various actions which make up the workflow represented by a form. The workflow is not defined a priori; instead, addition of new forms may add or alter individual steps.

Let us consider a brief example. Suppose the secretary to the Dean creates a Leave Application form to be filled in by staff members prior to going on leave. The Dean must approve the application, after which it is sent to human resources and cannot be altered anymore. Prior to the Dean's approval, the applicant may change the dates of the application, but he cannot do so after approval is given. Months later, it is decided that the Head of Department (HoD) must give a separate approval, prior to the Dean's. A simple change of the form's definition must add this new information and alter the original access rules, which changes the workflow graph.

Another example in which data is re-used by different peers is given by the Publications Form. Suppose staff member John makes a form available to all staff in which they can enter and alter details of publications of which they are an author, and let others use parts of the entered information. Now suppose another staff member, Jill, wants to extend this form. She might create a form definition that re-uses John's data objects, but extends them with her own. End-users may enter data in either form, after which the basic publication data as defined in John's form is available to everyone, as long as the access rights are satisfied.

These two scenarios constitute a basis for proposing a new paradigm for web based information systems in which forms are first-class citizens representing complex, distributed instances and in which workflows are dynamically built up from the access rules present in form definitions.

Motivation. To the best of our knowledge, a system that allows all these functions does not yet exist. Currently, parts of the problem can be solved using various techniques and tools. For example, capturing data can be done by a distributed database, where users create their own tables and re-use information by using views[1] defined by others. Electronic forms can be generated using HTML and special purpose scripts, or the recent XForms [14] recommendation may be used depending on available implementations. In the latter case, access rights to data elements, and a concept of workflow, still need to be coded separately. Finally, commercial workflow systems (e.g. [7, 22]) require a complex design phase and implementation performed usually by specialists outside of the organization, after which adaptations in the business actions require a new cycle of design and implementation.

Hence, the two main motivations for this research are as follows. Firstly, we want to ultimately create truly enabling software that allows individuals a fairly easy way to create electronic forms, capture data with them that will be stored efficiently, and generate reports of data captured by their own forms but also those of other, distributed, users so long as this is allowed by access rules.

Secondly, as forms defined in the system include

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at The Seventeenth Australasian Database Conference (ADC2006), Hobart, Tasmania. Conferences in Research and Practice in Information Technology, Vol. 49. Gillian Dobbie and James Bailey, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

[1] This then raises the problem of updating relational views.

access rights, a workflow process is automatically associated with them. There is no need for a complex design phase for constructing a workflow, and any updates to the access rules (or new ones in a new form) incrementally modify the associated workflow. In addition to this significant benefit, we want to give creators of forms an easy method to check if the desired end states of the form can indeed be reached, and want to inform users of a form how the data is used (if this is allowed by the creator). These are properties that can be derived once the graph representing the workflow is constructed.

Contribution. The twin motivations listed above naturally translate in two research goals. The main contribution of this paper, however, lies with the formalization of the form-based peer-to-peer web information system. Specifically we formalize schemas for forms, and define the access rules language.

A secondary contribution lies in the exploratory fifth section by describing research questions associated to the derivation of workflow processes implied by access rules in forms.

Organization. This paper is organized as follows. In Section 2 we discuss both technical and theoretical work related to ours. In Section 3 we present the formal model for form-based peer-to-peer web information systems. We present the access rule language in Section 4 which will also be used to infer workflows, as explored in Section 5. Finally, in Section 6 we give a brief conclusion, discuss implementations, and outline the next steps in our ongoing research of deciding workflow processes in electronic form systems.

2 Related Work

The examples briefly described in the introduction point to both a variety of tools and systems that can be used to implement such a system, and also a variety of fundamental topics and concepts that are being drawn upon. In the first category, we may list tools such as XForms and server-side scripting languages (e.g. PHP) which facilitate communication with database servers (e.g. PostgreSQL). On the theory side, clearly all of the following are relevant: workflow theory, data and schema integration, distributed databases, views, peer-to-peer information systems, and security.

2.1 XForms

Many practical problems associated with electronic forms as implemented by HTML have recently been solved by the introduction of the XForms recommendation by the World Wide Web consortium [14]. In our work, we use XForms as but one of the tools to solve our technical problem; indeed, we use its strengths such as its Model-View-Controller design pattern, its client-side validation, construction of XML output, and so on. Importantly, however, we add many desirable fundamental features, thus suggesting new avenues of study in the context of future versions of XForms.

Looking at some important practical differences between our form-based information system and XForms, we list the following three issues:

- *Database connection.* In XForms, it is possible to read data from and write data to a database. But the tables must already exist, requiring form designers to know the schema and provide the correct SQL expressions.
  In contrast, in our system form designers need only to focus on creating the schema of their form, possibly re-using other form schemas; read and write access to and from the database, as well as preceding data-definition statements, are handled automatically.
- *Access rules.* In XForms, there are no rules to regulate access to data stored in the XML instance or a database. It is assumed that all data in the XML source is accessible, or that the database handles access rights. In the latter case, the form designer may not have full control over these constraints.
  In our proposed system, access rules are an explicit part of the form's definition, under control of the form's designer, and enforced by the forms server.
- *Workflow modelling.* There is also no notion of a workflow process attached to XForms; fields may be entered in a random order, although some values are calculated from others, and constraints may reference other fields.
  The access rules we require in a form's definition implicitly impose an order in which fields may be assigned values. Hence it is possible to infer a workflow process corresponding to a form.

Turning to some more theoretical issues w.r.t. XForms, we note that the recommendation is very complex owing to the fact that users have a very expressive language in which to describe forms. Not only does XForms use the full power of XML Schema's type system, it also introduces a rich constraint language. This expressivity precludes finding decidable problems such as completion. In our ongoing work, we will take a different approach, limiting expressiveness to allow the study of decidability problems.

2.2 Workflow Processes

The secondary aim of this paper is to investigate under which conditions (in the form of a data model and access rules language) it is possible to automatically construct a workflow graph corresponding to a form definition.

Research in the area of workflow modelling [3] has been active since the late eighties and has led to the commercial development of various Workflow Management Systems (WfMS) [18]. The main perspectives have traditionally been (1) *control flow* (or process), (2) *resource* (or organization), (3) *data* (or information), (4) *task* (or function), and (5) *operation* (or application) perspectives [2]. Often the aim has been to extend modelling concepts to better capture various subtle details of these perspectives. Dynamic derivation of workflow processes has not yet received attention, and constitutes a very significant motivation for using form-based information systems, which are the main contribution of this paper. The most relevant perspective relating to workflow research in our setting is the data perspective, as electronic forms record data and do this progressively on availability of other data previously entered.

In contrast, in Workflow Patterns [5] control flow (constraints on order of processing, synchronization, etc) is more important than data-flow. In our case, we focus on the flow of data and the operations performed on them; control-flow more or less implicitly follows from the data-flow.

Hence, a workflow *case* in the context of this paper is an instance over a certain form's schema, an *action* corresponds to the entry of data in a part of a form, and the workflow *process* is the sequence of actions that can be executed to arrive at a correctly completed form as defined by the access rules over the data provided by the form's designer.

Workflow Mining. Another area in workflow research recently has been the mining of workflow processes from diverse information sources such as transaction and event logs [4, 8]. In this case, as in our work, workflow processes are not modelled ahead of time by experts. However, the focus is significantly different from ours, and the two methods are completely independent.

Finally, in our own previous work [16, 17], we have discussed formal methods to decide when two workflow processes are the same, and have also presented non-destructive methods to integrate form-based views in workflow systems.

2.3 Databases, Modelling, Integration, Distribution, and Views

The largest area of research relevant to this paper is most obviously that of databases. Several topics are especially relevant. Firstly, our form definitions contain a schema strongly based on *entity-relationship modelling*. Instances over the schemas correspond in fact to *nested relations*, a concept widely studied in the seventies of the previous century. We represent the instances as trees and will normally serialize them as *XML documents*, another popular, if much more recent, database research area. Likewise, the schemas will be expressed in a language based on XML Schema. In addition, our access rules language is based on a subset of XPath corresponding to first order logic restricted to two variables (FO^2).

Distributed Databases and Peer-to-Peer Information Systems. A clear design decision for our forms system has been to use the powerful notion of peer-to-peer information systems. Rather than describing one centralized forms server, we assume groups of users each have their own server which communicates with peers to access forms, obtain data stored at other locations, and add to that data. This aspect opens up many interesting topics already studied in the context of distributed databases. We shall assume solutions from that field rather than re-invent the wheel. However, two related issues deserve some additional attention: views and data & schema integration.

Data and Schema Integration. Both within a single peer as between various peers, it is possible and desirable to reuse schemas introduced by different forms. Schema integration is a very complex and widely studied topic in the context of databases systems [10, 11, 20, 21]. In our case, some of the complexities are irrelevant, while other results are very much applicable. However, the main difference, from a data integration perspective, is that we don't limit integration to read-only data. Indeed, we must allow users to read information from various peers representing an integrated schema, but we must also enable them to update this data. This further complicates integration considerably, however, the description of our form-based information system will adequately support a simple and efficient procedure for reading and writing data from various sources.

Views. Our data model stores a collection of logically related data structured in accordance to the schemas present in forms. In a real sense, each form schema plus its access rules acts as a view on the underlying data model. The `isa` constraints that we define in Section 3 are able to *project* parts from different entities. The *join* operation is present in the modelling of relations in the schema. And some form of *selection* is done when applying access rules. Hence, creators of forms have an implicit view language at their disposal, and the problem of view updatability and maintenance appears.

The data model can be implemented in various types of database systems; usually, a relational DBMS will be used because our nested relations-like instances can be translated easily to this model. Alternatively, native or XML-enabled databases can also be used. When a relational system is chosen, results from research into relational views can be used [12]. Research into updating XML views is currently underway [19].

2.4 Security and Authentication

Forms systems collect, store, retrieve and disseminate possibly sensitive information. It is essential that such systems provide secure access to data when required. The area of security has been and continues to be thoroughly studied. Formal languages for expressing security relationships have also been defined [15, 24]. In our work, we take a rather database-oriented approach to security; in Section 4 we describe a rule language that enables a forms creator to define precisely who can create, read and delete data corresponding to forms. These rules will be part of the data dictionary of the forms system.

A reliable authentication process together with data encryption, typically based on public key cryptography and digital certificates, is needed in addition to the access rules to build a truly secure system. It is assumed that these authentication and encryption procedures are available as a service to the form-based information system, and so is outside the scope of the current work.

Authentication on single server systems is a straightforward operation. Extending authentication seamlessly to systems where data is shared across multiple peers is not so simple. We certainly wish to avoid the situation were a user is required to identify herself to every peer that is involved in a forms transaction. Instead, authentication should be carried out at the primary forms server, and inter-peer negotiation should propagate appropriate access rights to data on other servers. This problem as yet requires additional research.

3 FormWIS Data Model

We proceed with describing what a form-based web information system (FormWIS) is, and define the data model that it uses.

Form-based Web Information System. A FormWIS is a cooperative information system that presents all data to the user in electronic web forms. As such it offers a view on the underlying data model that can be updated through data entry in the form. Users can perform all manipulations of the presented data if this is allowed by the access rules that are part of the definition of a form[2].

Users can add new forms that may or may not share information with previously defined forms. The underlying data model is then automatically extended with the extra information in this new form definition. The access rules associated to the new data are determined by the user who defined the form.

[2] This implies that users may have to identify themselves by supplying, for example, a password. We assume such an authentication mechanism is present.

A FormWIS will cooperate with peers over a network, making data sharing between disparate organizations possible. What is special and desirable about FormWISs, is that they allow a more natural evolution of data capturing and liberal reuse of information sources both within an organization and with third parties, while maintaining strict rules about who has which type of access to which data. In addition, as will be discussed in Section 5, they allow for automatic and dynamic modelling of a workflow process.

3.1 Formal Definitions

To foster a clear understanding of the FormWIS data model, we will first define schemas and instances over schemas, before turning to two different semantics of instances.

3.1.1 Schemas and Instances

A FormWIS stores definitions of forms and instances that belong to a form. We will defer the formal definition of a *form* to Section 4, but one important part of a form is its schema. In turn, a schema consists of several parts. We first define a frame.

Definition 1 (Frame) *A* frame *is a tuple* $F = (C, R, s, t)$ *where*

- *C is a set of class names partitioned into C^e, the entity classes, and C^v, the value classes*
- *R a set of relation names*
- *$s : R \to C^e$ and $t : R \to C$ giving the source and target classes of a relation.*

A frame is said to be a forest frame *if the corresponding multi-graph is a forest.*

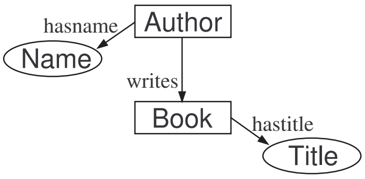


Figure 1: An example of a frame.

Example 1 *Consider the following frame F which will be used in the context of a form capturing book publication details for authors. $F = (C, R, t, s)$ where $C = C^e \cup C^v$ with $C^e = \{Author,\ Book\}$ and $C^v = \{Name,\ Title\}$. $R = \{writes,\ hasname,\ hastitle\}$, and $s(writes) = \{Author\text{: } dekeyser\ \} = s(hasname)$. Furthermore, $s(hastitle) = \{Book\} = t(writes)$, $t(hasname) = \{Name\text{: }\}$, and $t(hastitle) = \{Title\}$. A graphical representation of this frame is shown in Figure 1.*

One of the central ideas in our work is that we argue that forms are usually hierarchical in nature. It therefore seems natural to use an hierarchical data model like nested relations or XML. However, we will want to model *many-to-many* relations such as the relationship between the classes *Book* and *Author*, which is more easily done in a graph-based model such as the Entity-Relationship Model (ER Model) or the Object Exchange Model (OEM).

Definition 2 (Instance of frame) *An* instance *of a frame $F = (C, R, s, t)$ is a tuple (O, I_C, I_R) where*

- *O a set of objects partitioned into O^e, entities, and O^v, values*
- *$I_C : C \to 2^O$ the class interpretation function such that $I_C(c) \subseteq O^e$ if $c \in C^e$ and $I_C(c') \subseteq O^v$ if $c' \in C^v$*
- *$I_R : R \to 2^{O \times O}$ the relation interpretation function such that for all $(o_1, o_2) \in I_R(r)$ it holds that $o_1 \in I_C(s(r))$ and $o_2 \in I_C(t(r))$.*

Example 2 *Consider the following instance I of frame F presented in Example 1: $I = (O, I_C, I_R)$ where $O = O^e \cup O^v$ with $O^e = \{a_1, a_2, a_3, b_1, b_2\}$ and $O^v = \{knuth,\ date,\ widom,\ programming,\ databases\}$. Furthermore, $I_C(Author\text{: } dekeyser\ ) = \{a_1, a_2, a_3\}$ and $I_C(Book) = \{b_1, b_2\}$. Finally, $I_R(writes) = \{(a_1, b_1), (a_2, b_1), (a_2, b_2), (a_3, b_2)\}$, $I_R(hasname) = \{(a_1,knuth), (a_2,date), (a_3,widom)\}$, and $I_R(hastitle) = \{(b_1,programming), (b_2,databases)\}$.*

An instance of a frame can also be thought of as a labelled graph where the nodes are labelled with sets of classes (meaning that a node is an object in each of those classes) and edges are labelled with the name of a relationship.

Definition 3 (Instance graph) *The* graph of an instance (O, I_C, I_R) *of a frame* (C, R, s, t) *is the tuple* (O, E, λ) *where*

- *O is the set of nodes,*
- *$E = \{(o_1, r, o_2) | (o_1, o_2) \in I_R(r)\}$ is the set of labelled edges*
- *$\lambda : O \to 2^C$ the node labelling function such that $\lambda(o) = \{c \in C | o \in I_C(c)\}$.*

Example 3 *The instance formulated in Example 2 can be presented as the graph shown in Figure 2.*

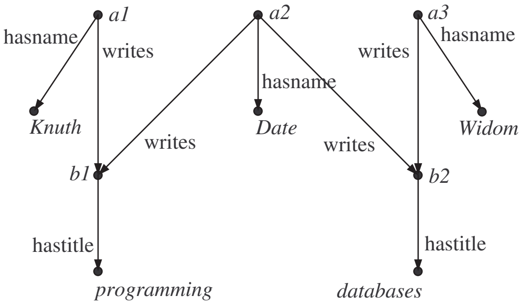


Figure 2: An instance graph for Frame F presented in Example 1, with the λ labelling function omitted for clarity.

We are now ready to define the schema of a form.

Definition 4 (Schema) *A* schema *is a tuple $S = (F, K)$ where F is a forest frame and K a set of* constraints *over F. These can be any type of constraint but we will assume here that they are of the following forms: c_1 isa c_2, r_1 $\mathsf{isa}^{\downarrow}$ r_2 and r_1 $\mathsf{isa}^{\uparrow}$ r_2 with $c_1, c_2 \in C$ and $r_1, r_2 \in R$.*

This definition allows us to create one conceptual schema from a variety of schemas each belonging to different forms. Thus, a form's schema is a forest frame consisting of one tree plus isa constraints. These constraints can refer to classes or relationships within the form's frame, but also to other forms' frames. It is only when taking the schemas of all forms together that one, purely conceptual, schema emerges; this *super schema* is a forest of trees with isa 'vines' between them. An example is given in Figure 3.

Note that it is the inclusion of the isa constraints in our theoretical data model that allows implementations to become peer-to-peer, hence providing the functionality alluded to in the title of this paper. Indeed, in Figure 3 the separate schemas that make up the conceptual super schema may be present on different peers.

3.1.2 Graph and Tree-based Semantics

Whereas a form's schema is a tree, a corresponding instance is actually a graph. Clearly the *writes* relation used in the previous examples is many-to-many, meaning that an author has written several books and a book may be written by several authors. Thus, the corresponding instance is a graph, as shown in Figure 2.

However, we would like instances to be trees, for a variety of reasons. First, because a form's schema is hierarchical, we would like to render its instances as trees on the users' screens. Exactly how the rendering should indicate that two objects shown is actually one and the same object is a GUI issue that we don't discuss here. An instance being a tree also allows data to be serialized as XML documents (perhaps using attribute references), which can then be further manipulated using languages such as XSLT and XQuery. The main reason, however, will become clear in Section 5: without hierarchical instances it is not possible to describe individual states in the state diagram corresponding to the form's workflow model.

To allow users to specify many-to-many relationships in the presence of both *hierarchical* schemas *and* instances, we will require them to use isa constraints. Consider that we have a second form whose schema $S' = (F', K)$. The frame F' is similar to F: $F' = (C', R', s', t')$, $C' = \{$Book', Author'$\}$, $R' = \{$written$\}$, $s'($written$) = \{$Book'$\}$, and $t'($written$) = \{$Author'$\}$. In addition, the set K has the following isa constraints: *Book'* isa *Book*, *Author'* isa *Author*, and *written* isa$^{\uparrow}$ *writes*. The conceptual super schema is given in Figure 3.

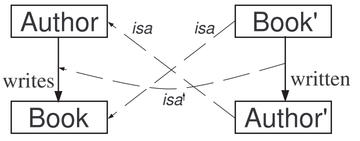


Figure 3: An example of a conceptual super schema obtained from schemas S and S'.

In the presence of such isa constraints, we must define the semantics of instances. We first turn to graph instances.

Definition 5 (Graph Instance) *A* graph instance *of a schema $S = (F, K)$ is an instance (O, I_C, I_R) of F that satisfies the constraints in K:*

1. *if c_1 isa $c_2 \in K$ then $I_C(c_1) \subseteq I_C(c_2)$*
2. *if r_1 isa$^{\downarrow}$ $r_2 \in K$ then $I_R(r_1) \subseteq I_R(r_2)$*
3. *if r_1 isa$^{\uparrow}$ $r_2 \in K$ then $I_R(r_1)^{-1} \subseteq I_R(r_2)$*

and the following two general constraints:

1. disjointness *if $o \in I_C(c_1) \cap I_C(c_2)$ then there is a $c_3 \in C$ such there are* isa *paths from c_3 to both c_1 an c_2 and $o \in I_C(c_3)$*
2. surjectivity *if $o \in I_C(t(r))$ then there is an o' such that $(o', o) \in I_R(r)$*

Note 1 *We make the following remarks about this definition:*

- *The disjointness constraint we use means that if an object belongs to two distinct classes, then these classes have a common subclass of which the object is also a member. This constraint is more liberal than requiring for two classes to be disjoint there should be no directed path of* isa *edges between them.*
- *There can be implicit* isa *constraints that are logically implied by the set K. In previous work, we have shown how they can be derived. We assume in this paper that this derivation happens automatically when a form is submitted.*
- *To have a simple notion of "location" of data we will assume that there are no cycles of* isa *edges and there is no multiple inheritance. Under these conditions there is always a unique highest class for an object, which might be considered as the true storage location of the data.*

Example 4 *The instance graph shown in Figure 2 is a graph instance since for the preceding examples the set of* isa *constraints K is empty, and the disjointness and surjectivity constraints are satisfied.*

As we will not be using graph instances, but hierarchical instances, we now turn to the tree-based semantics.

Definition 6 (Tree Instance) *A* tree instance *of a schema $S = (F, K)$ is an instance (O, I_C, I_R) of frame F plus an equivalence relation $\equiv \subseteq O \times O$ such that the graph of the instance is a forest and nodes are labelled by λ with at most one class, and moreover satisfies the constraints in K under $\equiv$:*

1. *if c_1 isa $c_2 \in K$ then $I_C^{\equiv}(c_1) \subseteq I_C^{\equiv}(c_2)$ where $I_C^{\equiv}(c) = \{[o]^{\equiv} | o \in I_C(c)\}$*
2. *if r_1 isa$^{\downarrow}$ $r_2 \in K$ then $I_R^{\equiv}(r_1) \subseteq I_R^{\equiv}(r_2)$ where $I_R^{\equiv}(r) = \{([o_1]^{\equiv}, [o_2]^{\equiv}) | (o_1, o_2) \in I_R(r)\}$*
3. *if r_1 isa$^{\uparrow}$ $r_2 \in K$ then $I_R^{\equiv}(r_1)^{-1} \subseteq I_R^{\equiv}(r_2)$*

and the following four general constraints:

1. disjointness *if $o_1 \in I_C(c_1)$, $o_2 \in I_C(c_2)$ and $o_1 \equiv o_2$ then there is a $c_3 \in C$ and $o_3 \in I_C(c_3)$ such $o_2 \equiv o_3$ and there are* isa *paths from c_3 to both c_1 an c_2 and $o \in I_C(c_3)$*
2. surjectivity *if $o \in I_C(t(r))$ then there is an o' such that $(o', o) \in I_R(r)$*
3. duplicate-free attributes[3] *if $(o_1, o_2) \in I_R(r)$ and $(o_1, o_3) \in I_R(r)$ then $o_2 \not\equiv o_3$*
4. equivalent common attributes[4] *if $o_1, o_2 \in I_C(s(r))$, $o_1 \equiv o_2$ and $(o_1, o_3) \in I_R(r)$ then there is an $o_4 \in O$ such that $o_3 \equiv o_4$ and $(o_1, o_4) \in I_R(r)$*

It is important to understand (1) why the graph semantics and the tree semantics are not equivalent, and (2) why this is not important in the context of this paper. However, the reasons for this can only be given when we have defined the access rules.

[3] Every equivalence class appears only once in attribute.

[4] Equivalent nodes have the same set of equivalence classes in common attributes.

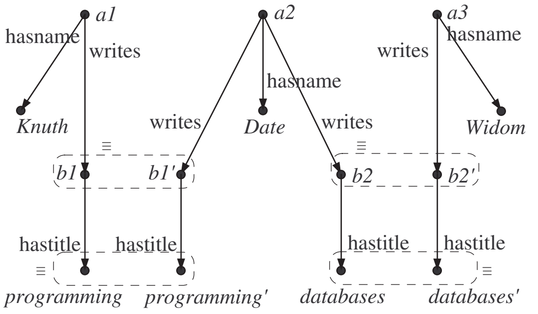


Figure 4: A tree instance corresponding to the graph instance shown in Figure 2.

4 Access Rules

We have now defined the data model for the forms system. Up to now a form has a hierarchical schema to which a number of hierarchical instances correspond. This means that users accessing a form can now be shown instances of the form, and can also make changes to the instance. However, we need to be able to specify access rules to each object in the instance. We will describe such rules in this section.

4.1 Actions on Data

Users of the forms system will want to do a number of things with it. Firstly, they want to open a form to see the instances that are associated with it. Clearly, it should be possible to restrict users to see only instances they are entitled to see, as defined by the form's designer. Secondly, they will want to update information in the form, either by changing values in an existing instance, or by creating a new instance all together.

A good example is the paper submission form present in our department. Users of the form will want to see those details of all papers that they are entitled to see. They will also want to be able to change the title of a paper, but only if it's a paper of which they are an author. Likewise, they should be able to add a new paper as long as they are an author of it.

In our system, we will allow the designer of a form to specify access rights tied to the schema of the form. The access rules in the schema will be evaluated over the corresponding instances, and only those objects in the instance where the rules are satisfied will be accessible.

CRUD. Of the usual CRUD (Create, Read, Update, Delete) rules used for accessing data objects, only the C, R, and D rights will be needed. This is because our rules will be tied to edges in the data model, and edges have no properties of their own, making an update of an edge meaningless. Updating the value of a property of a class involves removing an edge and creating a new one. Furthermore, as in previous work [13], we will only consider leaf operations: edges can be created or deleted if they appear as leafs in the tree. Larger operations on the tree (such as a move) can be simulated by a sequence of leaf operations.

Note that there is no need for propagation of updates between different peers, as each individual data item is stored at a unique location and peers that use it must obtain it from this location[1].

Handling updates on form schemas when they are used across different peers is another matter. We assume that in such instances the system will notify affected users.

4.2 Access Rule Language and Forms

As mentioned in Section 2, our access rule language is based on a limited subset of XPath corresponding to FO^2. Specifically, we use XPath's surface syntax, including conditions containing path expressions, but excluding the descendent-or-self axis (denoted as `//`). As our instances resemble nested relations more than semi-structured data, the nesting depth is always fixed in the schema, thus making this axis unnecessary.

The path expressions can be used on their own, meaning that existence of the end-node is checked, or in a comparison to a constant or one of three system variables `userid`, `date`, and `time`. In addition, path expressions can be combined using *and*, *or*, and *not*.

Definition 7 (Access Rule) *Given a frame $F = (C, R, s, t)$, an access rule is a tuple (e, o, r) with edge $e \in R$, o a* create, read, *or* delete *operation, and r an access rule expression.*

Thus, access rules are attached to edges in the frame of a schema and indicate that the operation in question may be performed when the access rule expression evaluates to true. An access rule is evaluated over each instance tree of the forest of instances belonging to a schema. Evaluation of a path expression starts from the node in the tree that corresponds to the class from which edge e departs in the schema.

Note that this definition of an access rule allow very fine-grained security provisions. Indeed, access is regulated to the level of individual attributes, giving the designer of a form full control over how data captured through her form, but also other forms that re-use part of her schema, is used.

Definition 8 (Form) *A form is a tuple (S, A) where S is a schema with a frame F that is a single tree, and A is a set of access rules.*

The notion of a form is the central idea in this paper: a form represents a tree-like data model with `isa` constraints and a set of access rules, and corresponds with a forest of tree instances over its schema. Users may access and edit the instances as long as the access rules are satisfied. Individual objects in the instances may be shared over various forms.

A Comprehensive Example. We now show how the real-life example described in the Introduction can be solved in our forms-based information system. Figure 5 gives a graphical representation of the schema of the *Leave Application* form. A non-exhaustive list of corresponding access rules is given in Figure 6. Note that we use the abbreviation U as a shorthand for both *create* and *delete* rules.

Rule (1) means that users can only create new leave applications for themselves. Rule (2) means that users can only see their own applications, except for the Dean and the Head of Department, who can see all applications. The begin and end dates for leave applications can only be changed by the user whose application it is, as stated by rules (3) and (4), and only if the Head of Department has not already approved the application. Rule (5) indicates that only

[1] Of course, for efficiency reasons, an actual implementation may choose to propagate updates instead; our model doesn't necessitate this but does allow it.

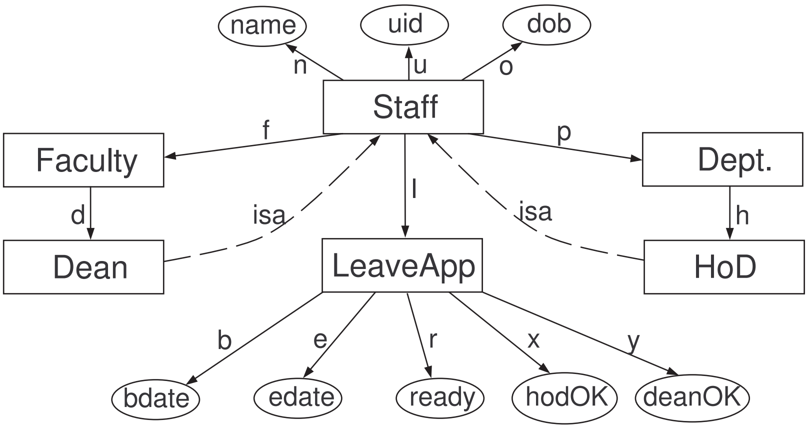


Figure 5: The Schema of the *Leave Application* Form.

$$(l,\ C,\ \texttt{./uid} = \text{userid}) \tag{1}$$
$$(l,\ R,\ \texttt{./uid} = \text{userid}\ \vee \texttt{./f/d/uid} = \text{userid}\ \vee \texttt{./p/h/uid} = \text{userid}) \tag{2}$$
$$(b,\ U,\ \texttt{../uid} = \text{userid}\ \wedge\ \text{not}(\texttt{./hodOK})) \tag{3}$$
$$(e,\ U,\ \texttt{../uid} = \text{userid}\ \wedge\ \text{not}(\texttt{./hodOK})) \tag{4}$$
$$(x,\ U,\ \texttt{./ready}\ \wedge \texttt{../p/h/uid} = \text{userid}) \tag{5}$$
$$(y,\ U,\ \texttt{./hodOK}\ \wedge \texttt{../f/d/uid} = \text{userid}) \tag{6}$$

Figure 6: Some access rules for the *Leave Application* Form.

the Head of Department can set the `hodOK` attribute to true, and only if the user has indicated that her application is ready. Hence, the system can automatically notify the Head of Department that his input is sought, when rule (5) is satisfied. This illustrates how control-flow is derived from the access rules. Finally, rule (6) says that the Dean can approve the application when the Head of Department has already done so.

The definition of a form indicates that it is the form's creator (or owner) that sets up the access rules for individual data items described in the form's schema. When another person creates a new form that re-uses all or part of the original form's schema, the original access rules still apply, in addition to any new access rules defined by the new form. To give an example, suppose the above Leave Application form was created by the Faculty, but the Head of Department wants to capture additional data if his staff are applying for leave (e.g. he wants them to supply a reason). His new form will re-use the original form's schema *and* access rules, and in addition he can add rules. All rules must be satisfied before the operation can proceed. If the HoD wants the original rules to be modified, he will need to negotiate with the owner of the form that first defined the rule. We argue that this precisely captures real-life dynamics within an organization, making significantly liberated capture and re-use of data possible while maintaining the highest level of security.

4.3 Information Leakage

While the access rules language we presented provides a very powerful yet elegant method to constrain access, the method is not water tight. Consider the following scenario: form designer *Bob* creates a class C_1 and specifies that only he can read it. Now suppose a second person *Alice* creates a form with a class C_2 and specifies that the class C_2 may be read if $C_1.a = x$ (where a is some attribute of C_1 and x is a value for a). Alice (and others) can now infer the the value of $C_1.a$ by attempting to modify C_2, which is against the access rule specified by Bob.

A simple access rule evaluator will hence allow information leakage in certain cases. A somewhat naive solution would be to decree that access rules may only be evaluated over parts of the instance tree that the user may see. However, this is circular as now visibility may become dependent on visibility.

It seems that information leakage, however, is a decidable property of a set of access rules. Hence, a practical solution to this potential security concern is to check newly submitted forms with their access rules and reject those that test positive for the information leakage property.

5 Workflow Processes

It is clear that the access rules not only regulate who can see and update which part of the instance, but also that these rules impose an order on updates. Hence, it would be very useful if our system could automatically derive the workflow process associated to this form. We investigate this in a preliminary manner in this section.

The reason why we include this rather informal discussion in this paper is twofold: firstly, automatic derivation of workflow processes is one of the main motivations for introducing form-based information systems. Secondly, the section will show that there are some highly interesting, non-trivial research problems to be found in this topic. This is the secondary contribution of this paper, and may perhaps inform the direction of continued research associated to XForms.

The leave application form detailed in the previous

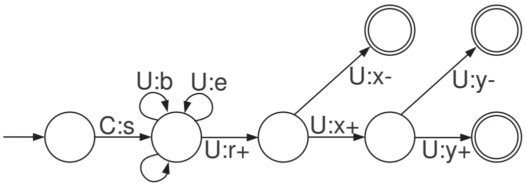


Figure 7: The *Leave Application* Workflow Process.

section illustrates that a form corresponds to a workflow process. In workflow literature, processes are usually modelled using finite state machines or more often using Petri nets [1, 23]. We will first examine finite state machines, attempting to establish whether one can be derived automatically for a form. Figure 7 shows a finite state machine (excluding some transitions for simplicity) modelling the workflow process of the leave application form shown in Figure 5.

States correspond to separate steps (or *actions*) in the workflow process. They represent an instance of the form at a certain time. A specific update of the instance represents a transition to another state.

The transitions are labelled with an abbreviation of the operation performed on the instance represented by the starting state of the transition. For example, `U:x+` means an update of edge x with value *true* (minus meaning *false*), while `C:s` means the creation of a new edge s to a new object of the LeaveApp class.

Importantly, the FSM in Figure 7 shows that different end states may exist. For this purpose, we extend the definition of a Form to include a *completed form formula*, expressed in the access rule language, which describes 'completion' end states; i.e., states that the creator of the form has indicated are acceptable completions of the form.

An interesting research question can now be posed: can we represent a form's workflow process using finite state machines such that reachability of completion states is decidable?

5.1 Canonical Instances

The first problem we need to solve is that of finding a finite representation of the infinite number of instances that may correspond to a form. The number of instances is infinite even when only structure is concerned and specific values are disregarded, since a relationship between two classes in a schema may have a many-to-many participation constraint. Figure 8 shows some instances for a simple frame involving relationships a and b.

Figure 8 also illustrates that it is possible to partition the set of infinite instances into a finite number of equivalence classes, which we call *canonical instances*. From the perspective of the access rule language (at least when considering only unary path expressions that check whether a node exists but does not compare values) each member of a canonical instance is indistinguishable from another member.

In the example, canonical instance I represents all instances in which no a edge exists (and hence no b edge under it). Canonical instance II represents all instances in which at least one a exists, but none of them have a b child. The third canonical instance represents those that have at least one a edge, and each of them has a b edge. Finally, canonical instance IV contains instances that have at least one a, and at least one of those has a b child.

Theorem 1 (Canonical Instances) *The set of all tree instances of a form can be partitioned into a finite number of canonical instances.*

Clearly it is very important that we now have a finite number of canonical instances, since finding a finite state machine representing a form's workflow process involves finding a finite set of states for the automaton.

Note that we can only create canonical instances for tree instances, not for graph instances. That is why we presented both in Section 3, and stated that we have to use the tree instances instead of the graph instances.

Unfortunately, a finite set of states for the FSM is not sufficient. Consider that transitions between instances going left to right in Figure 8's top row represent addition operations that add an a edge, while transitions in the opposite direction represent deletions of such edges. The problem is that the transition from canonical instance II to I also involves a deletion of the 'last' a remaining in II. We require these cases to be in two different canonical instances, because our XPath-based access rule language can distinguish between the two (e.g. `/a`). Hence, we require *counting* to determine if a transition stays within a canonical instance, or results in another canonical instance.

The conclusion is that we cannot use canonical instances as the states and update operations as the transitions of the FSM that is to represent the workflow process of a form. Therefore we briefly considered using Petri nets because they have the ability to do some counting using different *tokens* inside *places*. Unfortunately, use of *inhibitor* arcs [9] proved necessary to perform the counting we need, thus making reachability undecidable.

5.2 Decidability

The problems outlined above indicate that checking reachability of completion states is undecidable.

Theorem 2 (Undecidability) *Given a form with a schema, access rules, and a completed form formula, it is undecidable whether a completion state can be reached.*

The proof involves reduction to the two-counter automaton, a FSM with two registers that contain whole numbers, and that can check whether the registers contain 0. Transitions increment or decrement the registers. It is well known that two-counters are Turing-complete. We can simulate the registers using edges in instances, and check for 0 by expressing for example *not*(a).

One positive result so far, involves dead-end states, or instances of a form that cannot be changed anymore. Dead-ends are usually interpreted in a negative way: as states that should not be reached because the

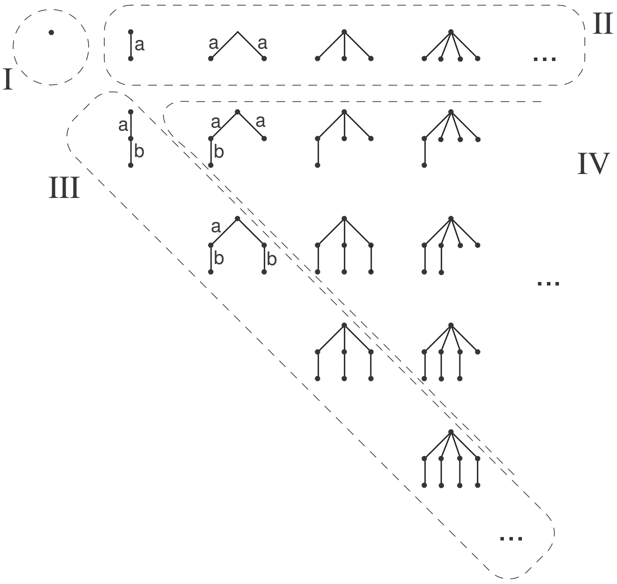


Figure 8: (Canonical) Instances over a simple schema. Some edge labels omitted for clarity.

system then enters a deadlock. However, it may be the case that a *completed form formula* describes a situation in which data in a form should not be updatable when some flag has been set. Not updatable means that it's a dead-end state, but one that represents a valid, correctly completed form. An example is where the Dean has approved a leave application, after which nothing in it can be changed anymore. In this sense, determining dead-ends is very worthwhile, helping the designer of the form to verify the rules he supplied with the form.

Theorem 3 (Dead-Ends) *Given a form including a schema and access rules, and an instance tree over the schema, it is decidable whether the instance can be updated.*

This result is due to our access rules language conforming to FO^2, which is a decidable subset of First Order logic using only two variables.

Unfortunately, because of the undecidability of general reachibility, we cannot prove that the dead-ends we found were reachable in the first place, making this result somewhat less practical.

To end our examination of the secondary aim of this work, i.e., under which circumstances would it be possible to automatically construct a workflow process from a form definition, we offer the conjecture that a positive result may be found when a slightly weaker access rules language is used.

Conjecture 1 (Decidability) *Given a form with a schema, access rules over the proposed XPath subset* without *negation, and a completed form formula, it is decidable whether a completion state can be reached.*

6 Conclusion, Implementations, and Future Work

Conclusion. We have presented a formal model for a form-based peer-ro-peer web information system. The model includes a definition for forms that incorporate a schema extended with access rules. A schema may be constructed by re-using elements from other schemas, both on the same peer and on other peers. This re-use is done via isa relationships. With a schema corresponds an instance in the form of a graph. To allow the access rules to traverse upwards to just one parent of a node, the instance is shredded into a forest of trees. The access rules impose an implicit order for data entry in the corresponding form, enabling us to check whether a workflow graph can be constructed, and to find specific states in the workflow.

Future Work

We are currently refining the data model we presented in this paper, and are investigating, given that data model, what subsets of the rule language do allow decidability while still maintaining a practical level of expressiveness.

We will also construct a rigorous proof, based on our current sketch, that the information leakage property we presented in Section 4.3 is decidable.

Implementation

On a practical level, we have already implemented a first prototype of very limited abilities [6] and are starting work on a second prototype that implements most of the ideas presented in this paper. Many practical issues, such as user interface design, will not be dealt with in this second prototype and will require additional research.

The second prototype is built around a substantially altered version of the Document Object Model as implemented in Java by the Xerces Apache project. It is currently a stand-alone forms server that accepts requests and updates and returns a form instance restricted by applying the relevant access rights. Extending it to enable peer-to-peer communication is being planned.

Derivation of workflow processes is not yet considered in the second prototype. This is the main goal of our third prototype, concurrently being planned. Here data is stored in a relational back-end, and a very restricted access rules language is offered. A key design issue is how to push the checking of access rules to the relational database.

Acknowledgement

We would like to thank Toon Calders for sharing his insights in the decidability problem described in this paper, and Hua Wang for valuable discussions about security issues.

References

[1] W. van der Aalst: Petri-net-based Workflow Management Software. In Proceedings of the NFS Workshop on Workflow and Process Automation in Information Systems, pp 114–118, May 1996.

[2] W. van de Aalst, P. Barthelmess, C. Ellis, and J. Wainer: Workflow Modeling using Proclets. In Proceedings of CoopIS'00, pp. 198–209, 2000.

[3] W. van der Aalst and K. van Hee: Workflow Management: Models, Methods and Systems. MIT Press, 2001.

[4] W. van der Aalst, A. Weijters, and L. Maruster: Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering (TKDE), volume 16(9), pages 1128-1142, 2004.

[5] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros: Workflow Patterns. Technical Report Eindhoven University of Technology. `http://is.tm.tue.nl/research/patterns/documentation.htm`, 2002.

[6] R. Addie: FormsFree – a secure but accessible system development and use of on-line forms. University of Southern Queensland Technical Report, December 2004.

[7] Advantys: `http://www.workflowgen.com/en/`, 2004.

[8] R. Agrawal, D. Gunopulos, and F. Leymann: Mining process models from workflow logs. Lecture Notes in Computer Science, 1377:469498, 1998.

[9] N. Busi: Analysis issues in Petri nets with inhibitor arcs. *Theoretical Computer Science*, 275:1-2, pp. 127–177, 2002.

[10] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati: Logical Foundations of Peer-To-Peer Data Integration. In Proceedings of PODS'04: pp. 241–251, 2004.

[11] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, G. Vetere: Hyper: A Framework for Peer-to-Peer Data Integration on Grids. ICSNW'04: pp. 144–157, 2004.

[12] S. Cosmadakis, and C. Papadimitriou: Updates on relational views. Journal of the ACM, 31(4):742–760, 1984.

[13] S. Dekeyser, J. Hidders, and J. Paredaens: A Transaction Model for XML Databases. World Wide Web Journal, 7(1): 29–57, Kluwer, 2004.

[14] M. Dubinko, L. Klotz, R. Merrick, and T.V. Raman: XForms 1.0. World Wide Web Consortium (W3C) Recommendation, October 2003.

[15] J. Glasgow, G. MacEwen, and P. Panangaden: A logic for reasoning about security. ACM Transactions on Computer Systems 10(3), pp 226–264, August 1992.

[16] J. Hidders, M. Dumas, W. van der Aalst, A. ter Hofstede, and J. Verelst: When are two Workflow Processes the same? Computing: the Australian Theory Symposium (CATS'05), 2005.

[17] J. Hidders, J. Paredaens, P. Thiran, G-J Houben, K. van Hee: Non-destructive Integration of Form-based Views. In Proceedings of ADBIS'05, 2005.

[18] D. Hollingsworth: Workflow Management Coalition: The Workflow Reference Model Ten Years On. Technical Report, February 2004.

[19] H. Kozankiewicz, J. Leszczylowski, and K. Subieta: Updatable XML Views. In Proceedings of ADBIS'03, LNCS 2798, 2003.

[20] M. Lenzerini: Data Integration Is Harder than You Thought. In Proceedings of CoopIS'01: pp. 22-26, 2001.

[21] M. Lenzerini: Data Integration: A Theoretical Perspective. In Proceedings of PODS'02: pp. 233–246, 2002.

[22] Microsoft: InfoPath 2003 Product Information: The Microsoft Office information gathering and management program. `http://www.microsoft.com/office/infopath/prodinfo/trial.mspx`, 2004.

[23] T. Murata: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, 77(4):541–580, April 1989.

[24] S. Nanchen and R. Stark: A Security Logic for Abstract State Machines. In Proceedings of the 11th International Workshop on Abstract State Machines '04. LNCS 3052, 2004.

Dimensionality Reduction in Patch-Signature Based Protein Structure Matching

Zi Huang[1] **Xiaofang Zhou**[1] **Dawei Song**[2] **Peter Bruza**[2]

[1]School of Information Technology and Electrical Engineering
University of Queensland,
St. Lucia, QLD 4072
Australia
Email: {huang,zxf}@itee.uq.edu.au

[2]Distributed Systems Technology Centre
Level 7, General Purpose South
University of Queensland,
St. Lucia, QLD 4072
Australia
Email: {dsong,bruza}@dstc.edu.au

Abstract

Searching bio-chemical structures is becoming an important application domain of information retrieval. This paper introduces a protein structure matching problem and formulates it as an information retrieval problem. We first present a novel vector representation for protein structures, in which a protein structural region, formed by the vectors within the region, is defined as a patch and indexed by its patch signature. For a k-sized patch, its patch signature consists of $7k - 10$ inter-atom distances which uniquely determine the patch's spatial structure. A patch matching function is then defined. As structures for proteins are large and complex, it is computationally expensive to identify possible matching patches for a given protein against a large protein database. We propose to apply dimensionality reduction to the patch signatures and show how the two problems are adapted to fit each other. The Locality Preservation Projection (LPP) and Singular Value Decomposition (SVD) are chosen and tested for this purpose. Experimental results show that the dimensionality reduction improves the searching speed while maintaining acceptable precision and recall. From a more general point of view, this paper demonstrates that information retrieval techniques can play a crucial role in solving this biologically critical but computationally expensive problem.

Keywords: Protein Structure Matching, Similarity Measure, Dimensionality Reduction

1 Introduction

Information science has been applied to computational biology, resulting in a new field called Bioinformatics, which investigates "the collection, archiving, organization and interpretation of biological data" (Orengo, Jones & Thornton 2003).

Discovering functional relationships between proteins is recognized as a central task of modern bioinformatics. The problem of comparing amino acid sequences in proteins has been investigated extensively in the past. The research focus has now been shifted towards higher level biological structures and functions. It has been found that it is common for proteins that do not share significant sequence similarity to have significant structural similarity (thus potentially functional similarity) (Mount 2001). When the sequence similarity is below a certain percentage, say 20%, only structure analysis can reveal the potential relationship which may be hidden at the sequence level(Bourne & Weissig 2003). It is known that the protein's unique three-dimensional structure often determines its properties. Finding proteins with similarly substructures is an important problem, as certain structural regions of a protein often perform some specific functions, and having one or more similar 3D substructures has been considered as an essential condition for potential protein interaction.

As 3D protein structures are large and complex, it is computationally expensive to identify possible locations and sizes of the matching structural regions for a given protein against a large protein database. A commonly used structure representation is the inter-atom distance matrix. As the complexity of the distance matrix representation is quadratic to the number of atoms, it is very expensive for processing a large number of proteins.

To alleviate this problem, we introduce a patch signature model which has been recently proposed based on a vector representation for protein structures. A structural region is defined as a patch formed by the vectors within the region. The patch signature is used to characterizes a patch. Compared to the traditional distance matrix representation, patch signature is more compact and linear to the number of atoms. The matching function between two patches is then defined as pair-wise comparisons between their patch signatures.

However, the matching stage can still be very expensive since the dimensionality of patch signature data can be large when the size of patch is large. A obvious solution to more efficient patch matching is to reduce the dimensionality of patch signatures while maximally preserve the matching function defined between two patches in the resultant

Copyright (c) 2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

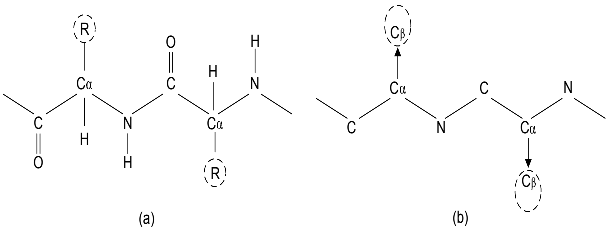


Figure 1: A fragment of amino acid chain.

lower dimensional space. Dimensionality reduction has been extensively applied in information retrieval. The goal is to find an "intrinsic" subspace, which is the best lower dimensional approximation of the original space depending on the objective function a dimensionality reduction algorithm tries to preserve. A well known approach is Singular Value Decomposition (SVD), which best preserves inner product in an Euclidean space and is the basis of the Latent Semantic Indexing (LSI)(Deerwester, Dumais, Landauer, Furnas & Harshman 1990)(Landauer, Foltz & Laham 1998). Recently, a novel Locality Preserving Projection (LPP) algorithm (He, Cai, Liu & Ma 2004) has been introduced to document indexing and demonstrated a better performance. Unlike SVD, LPP aims to preserve local geometrical structure in a manifold in terms of L_2 distance between data points. In this paper, we will address how the LPP and SVD can be applied to patch matching and demonstrate that they can largely improve efficiency (measured by CPU time) while maintaining an acceptable precision and recall.

The rest of the paper is organized as follows. Section 2 gives a brief introduction to protein structure and its 3D representations. We present a patch signature model and its similarity measure in Section 3. Two typical dimensionality reduction approaches, SVD and LPP, and their application in patch signature matching are studied in Section 4. Section 5 reports the experimental results. The related work are described in section 6. Section 7 finally concludes the paper and highlights the future work.

2 Preliminaries

A protein is a large molecule composed of one or more chains of amino acids in a specific order. Twenty standard amino acids have been identified in protein structures. As illustrated in Fig. 1(a), each amino acid contains a central atom C_α to which an amino (N-H) group and a Carboxyl ($C = O$) group are attached. The amino group, carboxyl group and C_α atom construct the *mainchain*(or backbone) of an amino acid. In addition, each amino acid (except Gly) has a sidechain (or R group) attached to its central atom C_α. It is the sidechain and sidechain alone which distinguishes one amino acid from another and furthermore confers the specific function to an amino acid(Bourne & Weissig 2003). The *sidechain* is typically connected to C_α via another atom C_β(Branden & Tooze 1998). A protein is constructed by amino acids that are linked by peptide bonds forming a polypeptide chain.

The amino acid sequence of a protein's polypeptide chain is called its primary structure, which can be represented a linear string of amino acids, abbreviated with one-letter codes.

Protein structure can be folded into a three-dimensional configuration as a set of points (atoms) in 3D space. For example, PDB (Protein Data Bank)(*Protein data Bank* n.d.) arranges a protein on an imaginary Cartesian coordinate frame and assigns (x,y,z) coordinates to each atom. This representation serves as a basis of different higher level representations. Different regions on the amino acid sequence form regular secondary structures, including the α helices and β sheets in the three-dimensional space. A 3D protein structure can usually be characterized by its mainchain (via C_α atoms) and/or sidechains (via C_β atoms).

For example, in the DALI(Holm & Sander 1993)(Holm & Sander 1996) system, a distance matrix containing all pairwise distances between C_α atoms is built, where each C_α-C_α distance reflects the relationship of two amino acids respectively centered by the two C_α atoms. If the distance between two amino acids (A_i and A_j) of protein A is similar to the distance between two amino acids (B_i and B_j) of protein B, amino acids A_i and A_j could be mapped to the amino acids B_i and B_j.

The VAST(Gibrat, Madej & Bryant 1996) and SARF(Alexandrov & Fischer 1996) systems use secondary structural elements (SSE). Each SSE in a protein is represented by position, length, and direction of a vector determined by the position of the C_α atoms along the SSE. It assumes that if two vectors representing two secondary structures are similar, the internal structure within secondary structures are similar.

The program SSAP(Orengo & Taylor 1996)(Orengo & Taylor 1989) represents 3D structure of protein as structural environments for amino acids, each of which is the set of vectors from the C_β atom to C_β atoms of all other amino acids in the protein.

There are some other methods such as Torsion (dihedral) Angles (Bergeron 2003). However, all the above methods are based on either mainchains (via C_α) or sidechains (via C_β) alone, thus they are insufficient to model the orientation of sidechains. A different way of representing a protein's structure as vectors of C_α-C_β atoms. A pair of C_α-C_β atoms in the same amino acid constructs a vector, denoted $\overrightarrow{C_\alpha C_\beta}$, from its C_α end to C_β end. More recently, the vector representation model(Spriggs, Artymiuk & Willett 2003, Huang, Zhou & Song 2005) has been operationalized. For each residue, a vector from C_α to C_β can be constructed. This vector representation involves not only the mainchain but also the sidechain information. The position of C_β atom is used to emphasize the functional part of the side-chain corresponding to the vector. The vector representation also offers a flexibility of generalizing the use of C_β to a psudo-atom (center of the sidechain). It has been argued in (Artymiuk, Spriggs & Willett 2005)(Spriggs et al. 2003):

"The vectorial representation is clearly an extremely simple description of the relative orientations of the side-chains in a 3D protein structure. It does, however, have the advantage that it does not overdefine the orientations of ends of side-chains, as could occur if a more precise representation was to be used that was based directly on the individual atomic coordinates in the PDB. "

There are currently over 30,000 proteins in the PDB database, containing 3D coordinates of all atoms in each protein. It is practical and relatively straightforward to build the vector model for each protein and calculate Euclidean distances between atoms. For the rest of this paper, a protein always means its vector model. We adopt this approach as a basis of our

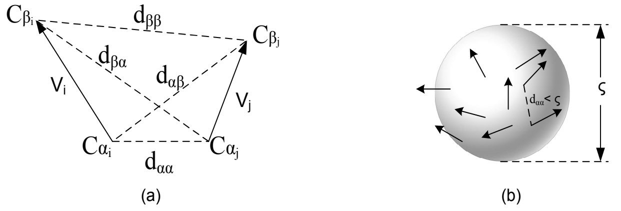


Figure 2: (a)Spatial relationship between two vectors. Four internal distances are denoted as $d_{\alpha\alpha}$, $d_{\beta\beta}$, $d_{\alpha\beta}$, and $d_{\beta\alpha}$. (b) An example of patch. Each vector represents an amino acid. The diameter is ε. The dashed line shows the α-α distance ($d_{\alpha\alpha}$) between two vectors.

model, which is formulated in the next section.

3 Problem Formulation

This section presents a protein structure matching problem, which has been first introduced in (Huang et al. 2005). The problem essentially deals with the identification of matching structural regions, called "patches", between two proteins.

3.1 Vector Representation of Protein Structures

A protein can be defined as a set P of three dimensional vectors:

$$P = \{v_i | 1 \leq i \leq N\} \tag{1}$$

where $N = |P|$.

Each v_i denotes a vector of $\overrightarrow{C_\alpha C_\beta}$ for amino acid i (Fig.1(b)). The length of a vector (i.e., the distance between its α-end and β-end) is typically fixed at 1.5 Å (angstrom).

3.2 Characterizing Protein Structures via patch Signatures

Since the proteins can be represented as geometric objects. The structures of the geometric objects have a direct influence on the proteins structure matching. We propose that 3D protein structure comparison can be performed by comparing the spatial relationship among vectors between two proteins. In other words, if two protein structures are similar, the spatial relationship among vectors of one structure must be similar to that of the other. The notion of *characterization of spatial relationship* refers to constraints which tie the vectors so that they have a fixed spatial relationship. That is, they can only rotate or translate globally as a whole without any internal change of positions. As the distances between atoms play a significant role in protein structure analysis, here we consider a distance-based characterization of spatial relationships between vectors. Since the PDB (Protein Data Bank) supplies coordinates of each atom of proteins in three-dimensional space, it is easy to calculate Euclidean distances between atoms.

The structural regions on a protein can be described as *patches*(Huang et al. 2005) which are subsets of vectors in the protein structure within a certain distance cutoff.

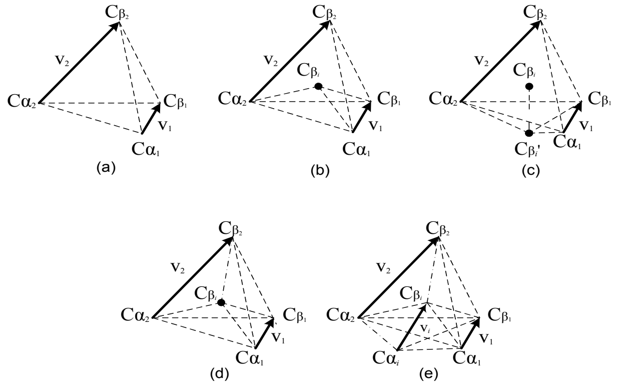


Figure 3: Patch Signature.

Definition 1 (Patch). *A patch is defined as a spherical region of protein P, whose diameter is ε (ie. a distance cut-off) (Fig.2)(b). More formally, $M = \{v_1, v_2, ..., v_Q\} \subseteq P$ ($Q > 2$) is a patch if ($\forall v_i, v_j \in M, d^{i,j}_{\alpha\alpha} \leq \varsigma$). In addition, M is called a non-extendable patch if and only if ($\forall v_i, v_j \in M, d^{i,j}_{\alpha\alpha} \leq \varsigma$) $\wedge$ ($\forall v_k \in M, \forall v_l \notin M, d^{k,l}_{\alpha\alpha} > \varsigma$).*

We can observe from the above definition that a non-extendable patch actually represents a maximal structural region with respect to a distance cutoff ς ($15\mathring{A}$ in this paper). Generally, a patch is a set of vectors with particular constraints on spatial arrangement.

Proposition 1. *For a k-sized patch ($k > 2$), $7k - 10$ internal distances are sufficient to characterize the spatial relationship among the vectors.*

A formal proof of this proposition can be found in (Huang 2005). As an example, we can look at one way of introducing the internal distances, as illustrated in Fig.3 (the dashed lines as internal distances). The first two vectors v_1 and v_2 form a stable triangular pyramid with the internal distances among their ends (Fig.3(a)). When the i^{th} ($i > 2$) vector v_i comes in, it constructs two triangular pyramids for tying to the original structure (i.e. v_1 and v_2), with four internal distances $d^{v_1,v_i}_{\alpha\alpha}$, $d^{v_1,v_i}_{\beta\beta}$, $d^{v_1,v_i}_{\alpha\beta}$, $d^{v_1,v_i}_{\beta\alpha}$, and other three distances $d^{v_2,v_i}_{\alpha\alpha}$, $d^{v_2,v_i}_{\beta\beta}$ and $d^{v_2,v_i}_{\alpha\beta}$ (Fig.3(e)). Therefore, for k vectors, the total number of internal distances is $4 + (k-2) \times 4 + (k-2) \times 3 = 7k - 10$.

Proof. Proof omitted. Refer to (Huang 2005) for details. □

For a k-sized patch ($k > 2$), the set of $7k - 10$ internal distances is called its *patch signature*, which identifies the spatial relationship between vectors in the patch. Proposition 1 proves that $O(k)$ distances are required. It is of significance because the patch matching algorithms presented later are based on a number of internal distances linear to k. The fewer distances involved, the faster in patch comparison.

A k-sized patch is an *unordered* collection of k vectors and in theory it has $k!$ representations of $7k - 10$ distances. Ordering the vectors is necessary for generating a unique representation of the patch.

To generate an ordering of vectors in a patch, a *basevector* v_{i_b} needs to be selected as a starting point, based on which an ordering function $\phi_{i_b} : q \rightarrow q' | q, q' = 1..k$ is defined. An detailed ordering algorithm was given in (Huang 2005). Throughout the rest of the paper, we assume that the vectors in any k-sized patch S are already ordered, denoted as $S_\triangleleft$. Now consider a k-sized patch $S_\triangleleft = \{v_{i_1}, v_{i_2}, ..., v_{i_k}\}$. According to Proposition 1, $S_\triangleleft$ can be represented uniquely by $7k-10$ distances (dimensions) in the following order:

$$S_{\triangleleft} = < d_{\alpha\alpha}^{i_1,i_2}, d_{\alpha\alpha}^{i_1,i_3}, ..., d_{\alpha\alpha}^{i_1,i_k}, d_{\alpha\alpha}^{i_2,i_3}, ..., d_{\alpha\alpha}^{i_2,i_k}, \\ d_{\beta\beta}^{i_1,i_2}, d_{\beta\beta}^{i_1,i_3}, ..., d_{\beta\beta}^{i_1,i_k}, d_{\beta\beta}^{i_2,i_3}, ..., d_{\beta\beta}^{i_2,i_k}, \\ d_{\alpha\beta}^{i_1,i_2}, d_{\alpha\beta}^{i_1,i_3}, ..., d_{\alpha\beta}^{i_1,i_k}, \\ d_{\beta\alpha}^{i_1,i_2}, d_{\beta\alpha}^{i_1,i_3}, ..., d_{\beta\alpha}^{i_1,i_k} >$$

Recall that in Section 2 we have mentioned the distance matrix approach, which would require three matrices to store all the $d_{\alpha\alpha}$, $d_{\beta\beta}$, $d_{\alpha\beta}$, and $C_{\beta\alpha}$ distances. The advantage of our patch signature model lies in its *linear* representation, based on which we shall develop more efficient patch comparison algorithms.

3.3 Patch Matching

The following is the definition of a matching function between two patch signatures.

Definition 2 (k-sized patch matching). *Given two k-sized patches, $S_{\triangleleft} = \{v_{i_1}, v_{i_2}, ..., v_{i_k}\}$ and $S'_{\triangleleft} = \{u_{i_1}, u_{i_2}, ..., u_{i_k}\}$, both represented by their $7k-10$ dimensional patch signatures, i.e. $S_{\triangleleft} =< s_1, s_2, ..., s_n >$ and $S'_{\triangleleft} =< s'_1, s'_2, ..., s'_n >$. They match each other (denoted as $S_{\triangleleft} \approx_\delta S'_{\triangleleft}$ or in short $S_{\triangleleft} \approx S'_{\triangleleft}$) if*

$$s_1 \approx s'_1 \wedge s_2 \approx s'_2 \wedge ...s_n \approx s'_n \qquad (2)$$

where "≈" means "equals to within a tolerance δ".

Based on the k-sized patch matching, we can then define the non-extendable patch and protein structure matchings as follows.

Definition 3 (Non-extendable Patch Matching). *For two non-extendable patches M and M', they match each other ($M \approx_\delta M'$) if there exists a k-sized patch $S \subseteq M$ and another k-sized patch $S' \subseteq M'$ such that $S_{\triangleleft} \approx_\delta S'_{\triangleleft}$ and $5 < k \leq 20$.*

Definition 4 (Protein structure matching). *For two proteins P and P', they have a matching structure if there exists non-extendable patches $M \subseteq P$ and $M' \subseteq P'$ such that $M \approx_\delta M'$.*

In summary, given a query protein Q, the general problem we investigate is to find all the proteins from a protein database such that the resultant proteins have a one or more matching non-extendable patches with Q, and to identify all the maximum sized matching patches. The maximum sized matching patches are of interest and will be presented to the biologists for post-processing and further investigation.

3.4 A Match-and-Expand Strategy

In this subsection, we introduce a match-and-expand strategy for fast protein structure matching.

If two non-extendable patches M and M' have a maximal matching patch of K vectors, they must also have matching sub-patches of 1, 2, $\cdots$, $K-1$ vectors. The match-and-expand strategy, similar to the philosophy of BLAST system(Altschul, Gish, Miller, Myers & Lipman 1990), first matches patches of the size $k(k \leq K)$ to reduce the number of candidates. A set of all patches of size k is pre-computed for all proteins in the database. In order to check if M and M' have a matching patch, the k-sized patches of M and M' are checked first. If no k-sized matching sub-patches are found, M and M' will not have any matching sub-patches. Otherwise, M and M' will be further checked in the expand step, starting from their matching k-sized patches, until finding maximum K sized matching patches. Operationally the expand stage can be accomplished by incrementally expanding k-sized sub-patches S and S' by one vector each time until maximum matching patches are reached.

The choice of k is important. If it is too small, then the match step may generate too many false hits; if it is too large, then the cost of materializing all k-sized patches can be very high. However, the choice of k is beyond the scope of this paper. We will focus on the match step.

We have defined the patch signature which is linear with respect to the number of atoms within a patch. The two $k-sized$ patches can then be matched by comparing their patch signatures. Though the dimensionality of this representation ($7k-10$) is much less than the traditional inter-atom distance matrix (C_{2k}^2), searching a large patch database is still expensive when k is large. A obvious solution to the problem is to reduce the dimensionality of patch signatures while maximally preserving the matching function between two patches in the lower-dimensional space. We will study two powerful dimensionality reduction approaches in the next section and discuss how to apply them on patch signature data.

4 Dimensionality Reduction on Patch Signatures

Dimensionality reduction has been extensively applied in information retrieval. The goal is to find an "intrinsic" subspace, which is an approximation of the original space but with a lower dimensionality. It has been demonstrated that there does exist an intrinsic semantic sub-space where the dimensions with lower eigenvalues carry redundant information and therefore can be truncated(Ding 1999).

On the other hand, projecting the original data to a lower dimensional space also helps discover some embedded "latent semantics" - i.e., some implicit associations which are unseen in the original high dimensional space.

A well known dimensionality reduction approach is the Singular Value Decomposition (SVD), which is the basis of the Latent Semantic Indexing (LSI)(Deerwester et al. 1990)(Landauer et al. 1998).

Recently, a Locality Preserving Projection (LPP) algorithm (He et al. 2004) has been introduced for document indexing and demonstrated better performance than SVD. Unlike SVD, which preserves inner product in an Euclidean space, the LPP aims to preserve local geometrical structure of data manifold.

Note that our patch signature matching function, defined in the last section, requires that the difference between values of each dimension of two data points should be within a tolerance. Neither SVD nor LPP is designed to directly preserve such a matching function. Therefore, we propose to use the Euclidean distance based measure between two patch signatures as an approximation of the previous pairwise matching function. Since k-sized patches can be equivalently treated as points in a $7k-10$ dimensional space, the similarity between two patches can then be measured by the Euclidean distance between them.

Definition 5 (Patch Similarity $\sim_{\delta'}$). *Given two k-sized patches $S_{\triangleleft} =< s_1, s_2, ..., s_n >$ and $S'_{\triangleleft} =< s'_1, s'_2, ..., s'_n >$. They are similar (denoted as $S_{\triangleleft} \sim_{\delta'}$*

$S'_\triangleleft$ or in short $S_\triangleleft \sim S'_\triangleleft$) if $d_2(S_\triangleleft, S'_\triangleleft) < \delta'$, where

$$d_2(S_\triangleleft, S'_\triangleleft) = \sqrt{\sum_{i=1}^{n} |s_i - s'_i|^2} \qquad (3)$$

Next, we will show theoretically how the Euclidean distance based similarity measure can return a super-set of the resultant matches from the pairwise matching and thus guarantees the recall of matching results.

Proposition 2. *If $S_\triangleleft \approx_\delta S'_\triangleleft$, then $d_2(S_\triangleleft, S'_\triangleleft) < \sqrt{n}\delta$*

Proof. This proposition can be proven trivially according to definition 3 and definition 6. □

The next two subsections will describe SVD and LPP algorithms respectively and give details in how they can be applied to the patch signature data.

4.1 Singular Value Decomposition (SVD)

Singular value decomposition (SVD) is a powerful technique from linear algebra. Given $m \times n$ patch signature matrix X with rank r, where m is the number of k-sized patches and n is the number (i.e., $7k - 10$) of dimensions, X can be decomposed to:

$$X = U\Sigma V^T \qquad (4)$$

where U and V are orthogonal $m \times r$ and $n \times r$ matrices respectively and Σ is an $r \times r$ diagonal matrix whose values are monotonically increasing non-zero singular values of X. The columns of U and V are the eigenvectors of XX^T and X^TX respectively.

Dimensional reduction is performed by taking only the first p eigen vectors and singular values to form:

$$X_p = U_p\Sigma_p V_p^T \qquad (5)$$

where U_p and V_p are $m \times p$ and $n \times p$ matrices composed of the first p columns of U and V respectively. According to the Eckart-Young theorem, X_p is the closest rank-p approximation by least square method to X in sense of both matrix Frobenius norm and 2-norm, i.e.

$$X_p = \min_{rank(B)=p} ||X - B||_2 \qquad (6)$$

$$X_p = \min_{rank(B)=p} ||X - B||_F \qquad (7)$$

Via SVD, the $j-th$ patch signature vector $S_{\triangleleft j}$ can be projected to a p-dimensional vector on the feature space of span V_p^T. The projected vector is actually recorded as the j-th row of U_p.

For a general exposition of the theory of SVD the reader is directed to (Golub & Van Loan 1996). The major difficulty of LSA is the choice of a suitable value for p. Tough the choice of optimal p can be theoretical, for example the work by Ding (Ding 1999), experimental approach is more widely used in information retrieval community, where an optimal p is derived by reference to some experiment. In our experiments we also adopt the experimental way.

4.2 Locality Preserving Projection

Locality Preserving Projection (LPP) (He & Niyogi 2003)(He et al. 2004) aims to preserve the intrinsic geometric structure in term of local neighborhood information of the data on a manifold.

Suppose a set of n-dimensional patches $x_1, x_2, ..., x_m$ in space $\Re^n$ form a $m \times n$ data matrix X. The core LPP algorithm includes the following steps:

1. Construct an adjacency graph with each data point (i.e., patch) as a node and put an edge between two point x_i and x_j if they are close enough. The closeness between x_i and x_j can be measured by their distance $||x_i - x_j||_2$. A simple but effective way of connecting two nodes is based on q nearest neighbors, i.e., x_i and x_j are the q nearest neighboring points;

2. A $m \times m$ adjacency matrix W is built whereby $W(i,j) = 1$ if x_i and x_j are connected; otherwise $W(i,j) = 0$.

There are some other options to the adjacency graph construction and adjacency matrix weighting. We do not compare these different options in this paper and will leave it as one of our future work.

3. Compute Eigenmaps by solving the following generalized eigenvector problem:

$$XLX^Ta_l = \lambda_l XDX^Ta_l \qquad (8)$$

where D is $m \times m$ diagonal matrix with $D_{ii} = \sum_j W_{ji}$, $L = D - W$ is the Laplacian matrix, λ_l is the l-th eigenvalue and a_l is the l-th eigenvector. The transformation matrix $A = [a_1, a_2, ..., a_p]$ can be formed, ordered by the eigenvalues $\lambda_1 < \lambda_2 < ... < \lambda_p$ where $p << n$.

4. Project the points to p-dimensional space $\Re^p$:

$$x_i \rightarrow x'_i = A^Tx_i \qquad (9)$$

Note that LPP is a linear approximation of Laplacian Eigenmaps (Belkin & Niyogi 2001). They both try to preserve locality via the following objective function:

$$min \sum_{ij}(x'_i - x'_j)^2 W_{ij} \qquad (10)$$

They are the same in the first two steps. The step 3 of the latter is to compute the generalized eigenvector for:

$$La_l = \lambda_l Da_l \qquad (11)$$

The rows of resultant $m \times p$ matrix A can be used as approximation of the original data in the lower dimensional space $\Re^p$:

$$x'_i = (a_1(i), a_2(i), ..., a_p(i)) \qquad (12)$$

The justification for their ability of preserving geometric structure on manifold is based on the Laplacian matrix L which is an approximation to the Laplace-Beltrami operator defined on the manifold (Belkin & Niyogi 2001).

5 Experiments

In this section, we set up the experiments and report the results of an extensive performance study conducted to evaluate the proposed representation model and the dimensionality reduction on protein patch data.

Table 1: Statistics of test data

Total number of proteins	811
Total number of vectors	190,669
Average number of vectors per protein	216
Average number of 16-sized patches per protein	5308

5.1 Experimental setup

5.1.1 Test Data

A total number of 811 sample proteins are selected for our initial experiments according to the PDB_LIST_20040601 (R-factor<0.2 and Resolution<1.9) in the WHATIF relational database. The PDB structures stored in the WHAT IF relational database are a representative set of sequence-unique (a sequence identity percentage cutoff of 30%) structures(*WHATIF relational database* n.d.). After pre-processing, the data statistics are shown in Table 1.

5.1.2 Query proteins

Ten different sized proteins are selected as queries. The average number of vectors per query is 238.

5.1.3 Baseline

To choose a baseline for comparison with our method, we perform pairwise matching of all distances between two patch signatures. The baseline matching results are assumed "correct matches".

The models we test in our experiments are the Euclidean distance based similarity search based on

- Dimensionality Reduction via SVD

- Dimensionality Reduction via LPP

5.1.4 Performance Indicators

Our programs are written in C++ and running on Pentium 4 CPU (2.8GHZ) with 1G RAM. The major performance indicators we used are:

- CPU time (in seconds) to complete a query

- Precision: percentage of returned patches being correct

- Recall: percentage of correct matching patches being returned

- F-measure: $\frac{2*Precision*Recall}{Precision+Recall}$

Note that all the experimental results reported later will be averaged for one query protein matches against one protein in the database.

5.1.5 Parameter settings

There are several parameters need to be set for our model and search method, four of which are fixed in our experiments:

- Distance cutoff (ς): $15\AA$

- Pair-wise matching tolerance (δ): $4\AA$

- Size of patches to match (k): 16 (leading to a total dimensionality 102 for patch signatures)

- Number of nearest neighbors in LPP (q): 10

Two other parameters are variables. We will test how the different settings of them affect the performance.

- Euclidean distance based similarity threshold δ': $1\AA$, $1.2\AA$, $1.5\AA$

- Size of reduced dimensionality p: 5, 10, 20, 30, 40, 50, 102

5.2 Experimental results

Table 2, 3, Fig.4, and Fig.5 summarize the experimental results. In addition, the CPU time for the baseline is 3.1 seconds. We can make the following observations:

Dimensionality reduction by both SVD and LPP under all the different parameter settings saves CPU time by from 3.2% up to 84%. This suggests that it does largely improve the efficiency for patch matching.

Larger threshold value δ' lead to increasing CPU time and recall, and decreasing precision. This co-relates our intuition. According to proposition 3, a threshold $\delta' = \sqrt{n}\delta = 40\AA$ guarantees 100% recall. The cost is losing precision. In the rest of our analysis, we take the F-measure as the main effectiveness indicator, as it represents the trade-off between precision and recall. We can observe that a much lower threshold like $1.2\AA$ is enough to obtain reasonable F-value.

The "intrinsic" dimensionality for either LPP or SVD is quite low (20 for SVD and 10-20 for LPP). In Fig.5, for each model the F-value grows rapidly until it reaches the peak, where the corresponding dimensionality is the intrinsic dimensionality. After this certain point, the F-value decreases while the dimensionality increases. This suggests that a large number of less significant dimensions carry no much meaningful information. This also indicates the usefulness and necessity of dimensionality reduction. It is also interesting to note the difference between LPP and SVD. The performance of SVD decreases more rapidly than LPP when the dimensionality increases. More theoretical comparison between the two approaches will be conducted in the future work.

6 Related Work

This paper deals with the problem of finding similar substructures. The most related techniques to our methods include protein structure modelling, such as geometric hashing and graph theoretical approach, and high-dimensional indexing for similarity search.

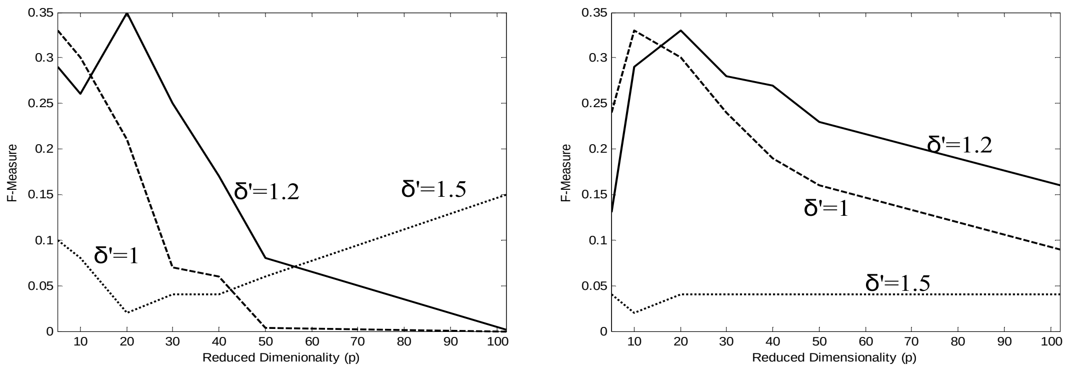
Geometric hashing (Wolfson 1997) was originally developed in computer vision and now used in protein structure comparison. It defines a set of reference frames for a structure. The coordinates of all points in the structure are re-calculated in a reference frame, forming a reference frame system. Geometric features of the structure are calculated based on the reference frame systems and stored in a hash table. This method ignores the sequential order of amino acids and gives the result invariant to the translation and rotation of the compared structures(Nussinov & Wolfson 1991) and thus is useful to discover matching substructures. However, we do not adopt this approach as it is computationally expensive. The number of reference frame systems to be constructed and the number of frame system comparisons are both

Table 2: Summary of SVD performance

p	δ'	CPU Time $\%ofbaselineCPUtime$	Precision	Recall	F-Measure
5	1	0.4 13%	0.21	0.79	0.33
	1.2	0.4 13%	0.17	0.90	0.29
	1.5	0.9 29%	0.05	0.99	0.10
10	1	0.5 16%	0.7	0.23	0.30
	1.2	0.5 16%	0.16	0.75	0.26
	1.5	0.8 26%	0.04	0.99	0.08
20	1	0.52 17%	0.92	0.42	0.21
	1.2	0.57 18%	0.28	0.46	0.35
	1.5	1.2 80%	0.01	0.99	0.02
30	1	0.6 19%	0.85	0.39	0.07
	1.2	0.6 19%	0.24	0.26	0.25
	1.5	1 32%	0.02	0.99	0.04
40	1	0.8 26%	0.22	0.003	0.06
	1.2	0.8 26%	0.31	0.12	0.17
	1.5	2 64%	0.02	0.98	0.04
50	1	0.8 26%	0.3	0.002	0.004
	1.2	0.9 29%	0.23	0.05	0.08
	1.5	1.5 48%	0.03	0.98	0.06
102	1	1.1 35%	0	0	N/A
	1.2	1.3 42%	0.14	0.001	0.002
	1.5	2.5 81%	0.08	0.98	0.15

Table 3: Summary of LPP performance

p	δ'	CPU Time $\%ofbaselineCPUtime$	Precision	Recall	F-Measure
5	1	0.5 16%	0.14	0.74	0.24
	1.2	0.5 16%	0.07	0.77	0.13
	1.5	0.6 19%	0.02	0.96	0.04
10	1	0.5 16%	0.25	0.47	0.33
	1.2	0.5 16%	0.20	0.52	0.29
	1.5	0.8 26%	0.01	0.96	0.02
20	1	0.8 26%	0.37	0.25	0.30
	1.2	0.8 26%	0.32	0.34	0.33
	1.5	1 32%	0.02	0.94	0.04
30	1	0.6 19%	0.51	0.16	0.24
	1.2	0.7 23%	0.43	0.21	0.28
	1.5	2.1 68%	0.02	0.97	0.04
40	1	0.7 23%	0.62	0.11	0.19
	1.2	0.7 23%	0.57	0.18	0.27
	1.5	1.3 42%	0.02	0.98	0.04
50	1	0.7 23%	0.67	0.09	0.16
	1.2	1 32%	0.53	0.15	0.23
	1.5	1.1 35%	0.02	0.96	0.04
102	1	1 32%	0.78	0.05	0.09
	1.2	1 32%	0.65	0.09	0.16
	1.5	2.1 68%	0.02	0.96	0.04



(a) SVD performance

(b) LPP performance

Figure 4: F-Measure.

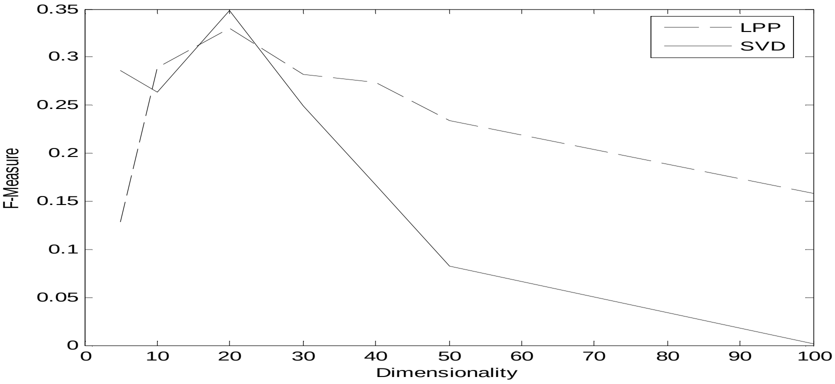


Figure 5: Intrinsic dimensionality(δ'=1.2).

combinatorial. Moreover, recalculation of the coordinates of points in a new reference frame via rotation and translation is also expensive.

The graph theoretical approach is used in systems, such as ASSAM(Spriggs et al. 2003, Grindley, Artymiuk, Rice & Willett 1993) and VAST(Gibrat et al. 1996), to find the maximal common substructure. The problem is often transformed to the clique problem. Vector representation is also used in ASSAM(Spriggs et al. 2003) which is developed to search for patterns of amino acid side chains in the 3D structures. A substructure is characterized by the distances among all pairs of vectors. This is therefore different from our model, where the overall spatial relationship of all the vectors in the substructure is characterized by its patch signature to make a more compact representation. ASSAM then detects cliques using a maximal common subgraph isomorphism algorithm borrowed from graph theory (Bron & Kerbosch 1973). As the clique detection problem is NP-Complete, many heuristic algorithms are developed. The most existing heuristic algorithms for the clique problem are partially enumerative and branch-and-bound based (Gardiner, Artymiuk & Willett 1997). However, they are insufficient to handle large scale data. For example, the test queries used for the experiments reported in (Spriggs et al. 2003) were all triad residues. In other words, the maximum size of cliques was three. A protein was "hit" once a matching substructure of size 3 was found. In our work, a query is a whole protein and we aim to find from the database all the matching substructures in any size. Therefore, we do not use the clique detection algorithms in our work. Instead, we developed a more scalable IR and database solution featured by a highly efficient query processing strategy.

7 Conclusions and Future Work

This paper presents a protein structure matching problem and formulates it as an information retrieval problem. A patch signature model is addressed based on a vector representation of protein structure. A protein structural region is defined as a patch, formed by a set of vectors within the region. A k-sized patch is then indexed by the $7k-10$ internal inter-atom distances constituting its patch signature. A matching function is defined to compare two patches based on their patch signatures. Though the dimensionality of this representation $(7k-10)$ is much less than the traditional inter-atom distance matrix (C_{2k}^2) approach, searching a large patch database is still expensive when k is large. We propose to apply dimensionality reduction to patch signatures and show how the two problems are adapted to fit each other. The Locality Preservation Projection (LPP) and Singular Value Decomposition (SVD)are chosen and tested for this purpose. Experimental results show that the dimensionality reduction improves the searching speed with acceptable precision and recall. From a more general point of view, this paper demonstrates that information retrieval techniques can play a crucial role in solving this biologically critical but previously computationally prohibitive problem. It is our hope that the marriage between information retrieval and bio-informatics will extend the boundaries of both areas.

From the experimental results, we can observe that there is still some room for further performance improvement in dimensionality reduction via both LPP and SVD (The best F-values are separately 33% and 35%). We will investigate other possibly more effective approximations to the pairwise patch matching function, other than the Euclidean distance used in this paper. On the other hand, more dimensionality reduction algorithms will be studied. At this stage, we focus on matching same sized patches. In the future, we plan to develop an efficient indexing mechanism for different sized patches. In this paper, we did not compare our approach to other protein structure matching algorithms. As a future work, we will also consider testing our approach on a collection of "homologs" produced from the SCOP database.

Acknowledgements

The work reported in this paper has been funded in part by the Australian Research Council (Grant No. DP0344488) and the Co-operative Centre for Enterprise Distributed Systems Technology (DSTC) through the Australian Federal Government's CRC Programme (Department of Education, Science and Training).

References

Alexandrov, N. & Fischer, D. (1996), 'Analysis of topological and nontopological structural similarities in the pdb: New examples with old structures', *Proteins* **25**, 354–365.

Altschul, S., Gish, W., Miller, W., Myers, E. & Lipman, D. (1990), 'Basic local alignment search tool', *J Mol Biol* **215**(3), 403–10.

Artymiuk, P., Spriggs, R. & Willett, P. (2005), 'Graph theoretic methods for the analysis of structural relationships in biological macromolecules', *Journal of the American Society for Information Science and Technology* **56**(5), 518–528.

Belkin, M. & Niyogi, P. (2001), Laplacian eigenmaps and spectral techniques for embedding and clustering, *in* 'Advances in Neutral Information Processing Systems 14 *NIPS*2001'.

Bergeron, B. (2003), *Bioinformatics Computing*, Pearson Education, Inc.

Bourne, P. & Weissig, H. (2003), *Structural Bioinformatics*, John Wiley and Sons.

Branden, C. & Tooze, J. (1998), *Introduction to Protein Structure*, Garland Publishing, Inc.

Bron, C. & Kerbosch, J. (1973), 'Algorithm 457 - finding all cliques of an undirected graph', *Communications of ACM* **1973**(16), 575–577.

Deerwester, S., Dumais, S., Landauer, T., Furnas, G. & Harshman, R. (1990), 'Indexing by latent semantic analysis', *Journal of American Society for Information Science* **41**, 391–407.

Ding, C. (1999), A similarity-based probability model for latent semantic indexing, *in* 'Proceedings of the Tweenty-Second Annual Internation ACM SIGIR Conference on Research and Development in Information Retrieval', pp. 59–65.

Gardiner, E., Artymiuk, P. & Willett, P. (1997), 'Clique-dection algorithms for matching three-dimensional molecular structures.', *Journal of Molecular Graphics and Modeling* **15**, 245–253.

Gibrat, J.-F., Madej, T. & Bryant, S. (1996), 'Surprising similarities in structure comparison', *Curr. Opin. Struct. Biol.* **6**, 377–385.

Golub, G. & Van Loan, C. (1996), *Matrix Computations*, John Hopkins University Press.

Grindley, H., Artymiuk, P., Rice, D. & Willett, P. (1993), 'Use of techniques derived from graph theory to compare secondary structure motifs in proteins', *J. Mol. Biol.* **229**, 707–721.

He, X., Cai, D., Liu, H. & Ma, W. (2004), Locality preserving indexing for document representation, *in* 'Proceedings of the 27th Annual Internation ACM SIGIR Conference on Research and Development in Information Retrieval', pp. 96–103.

He, X. & Niyogi, P. (2003), Locality preserving projections, *in* 'Advances in Neutral Information Processing Systems 16 *NIPS*2003'.

Holm, L. & Sander, C. (1993), 'Protein structure comparison by alignment of distance matrices', *J. Mol. Biol.* **233**, 123–138.

Holm, L. & Sander, C. (1996), 'Mapping the protein universe', *Science* **273**, 595–603.

Huang, Z. (2005), Indexing protein substructures for efficient similarity queries, Technical report, School of ITEE, University of Queensland, http://www.itee.uq.edu.au/ huang/Report.pdf.

Huang, Z., Zhou, X. & Song, D. (2005), High dimensional indexing for protein structure matching using bowties, *in* 'Proc. of 3rd Asia-Pacific Bioinformatics Conference', pp. 21–30.

Landauer, T., Foltz, P. & Laham, D. (1998), 'Introduction to latent semantic analysis', *Discourse Process* **25**(2&3), 259–284.

Mount, D. (2001), *Bioinformatics: Sequence and Genome Analysis*, Cold Spring Harbor Laboratory Press.

Nussinov, R. & Wolfson, H. (1991), 'Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques.', *Proc. Natl Acad. Sci.* **October-December**, 10495–10499.

Orengo, C., Jones, D. & Thornton, J. (2003), 'Bioinformatics: Genes, proteins and computers'.

Orengo, C. & Taylor, W. (1989), 'Protein structure alignment', *J. Mol. Biol.* **208**, 1–22.

Orengo, C. & Taylor, W. (1996), 'Ssap: Sequential structure alignment program for protein structure comparison', *Methods Enzymol.* **266**, 617–635.

Protein data Bank (n.d.), http://www.rcsb.org/pdb/.

Spriggs, R., Artymiuk, P. & Willett, P. (2003), 'Searching for patterns of amino acids in 3D protein structures', *J Chem Inf Comput Sci.* **43**(2), 412–21.

WHATIF relational database (n.d.), http://www.cmbi.kun.nl/gv/whatif/select/.

Wolfson, H. (1997), 'Geometric hashing: an overview.', *IEEE Comp. Science and Eng.* **October-December**, 10–21.

Recency-Based Collaborative Filtering

Yi Ding **Xue Li** **Maria E. Orlowska**

School of Information Technology and Electrical Engineering
University of Queensland
ITEE, University of Queensland, QLD 4072, Australia
Email: {ding,xueli,maria}@itee.uq.edu.au

Abstract

Collaborative filtering is regarded as one of the most promising recommendation algorithms. Traditional approaches for collaborative filtering do not take concept drift into account. For example, user purchase interests may be volatile. A new mother may be interested in baby toys, although previously she had no interest in these. A man may like romantic films while he preferred action movies one year ago. Collaborative filtering is characterized by concept drift in the real world. To make time-critical predictions, we argue that the target users' recent ratings reflect his/her future preferences more than older ratings. In this paper, we present a novel algorithm namely recency-based collaborative filtering to explore the weights for items based on their expected accuracy on the future preferences. Our proposed approach is based on item-based collaborative filtering algorithms. Specifically, we design a new similarity function to produce similarity scores that better reflect the reality. Our experimental results have shown that the new algorithm substantially improves the precision of traditional collaborative filtering algorithms.

Keywords: collaborative filtering, item-based approach, recency-based approach, concept drift, similarity measure

1 Introduction

With the evolution of the Internet, information overload is becoming a major problem for users. Different approaches are used in order to cope with this problem. Content-based filtering analyses the items' attributes to make recommendation, while collaborative filtering uses historical data on user preferences to predict items that a user might like. To date, collaborative filtering is best known for its use on e-commerce web sites. It has also been widely used in other areas, for example, filtering Usenet News, recommending TV shows and Web personalization.

Over the years, various approaches for collaborative filtering have been developed. However, in traditional approaches for collaborative filtering, data collection is regarded as static. Ratings produced at different times are weighted equally. Concept drift is not taken into account. Concept drift results from the fact that patterns revealed from analysing data is constantly changing. In collaborative filtering, concept drift means that user purchase preferences that we want to predict are always drifting. For example, a man previously liked romantic movies. So he rated *Casablanca* , a romantic movie released in 1942, the highest score 5 three years ago. But he changes his interest as time goes by. Currently, he dislikes romantic movies. He rated *Sweet Home Alabama* , a romantic movie released in 2002, the score 2 a month ago. Traditional collaborative filtering algorithms suppose these two scores have the same weights in predicting the user future preference for romantic movies.

Collaborative filtering is characterized by concept drift in the real world. The fast growth of e-commerce makes this concept drift factor more severe. In (Ali & Stam 04), the authors described the TiVo television show collaborative recommendation system which started four years ago. It has currently accumulated approximately 100 million user ratings, some of which are very old. In (Linden, Smith & York 2003), the authors mentioned that at Amazon.com old users can have a glut of information, based on thousands of ratings.

Since the value of these old ratings is questionable, we should seek to develop an algorithm that will deal with the changing data in collaborative filtering, thereby obtaining the most accurate prediction. That is to say, the algorithm must alleviate the influence of data representing outdated user preferences. A widely used approach is to discount old data at a constant rate. However, precisely calculating the discount rate of data can be quite difficult. A higher rate would lower the accuracy of the algorithm since it is supported by a less amount of training data and a lower rate would make the algorithm less sensitive to the current trend (Wang, Fan, Yu & Han 2003). Furthermore, in collaborative filtering, the discount rate depends on the duration of user preferences for items. This means the discount rate varies with different users and different items. The longer the target user's preference for a specific item lasts, the lower the discount rate.

In this paper we propose a novel algorithm namely recency-based dollaborative filtering, using weights for items based on their expected accuracy on the future preferences. Instead of discounting data using a discount function, our approach shall make decisions based on data distribution, rather than based solely on data arrival time. This is more reasonable and simpler compared to discount functions.

We use a simple example to illustrate the problem of concept drift in collaborative filtering. Assume the data points represent a user's preferences for a category of items in Figure 1.

One data point represents the user's rating on one

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

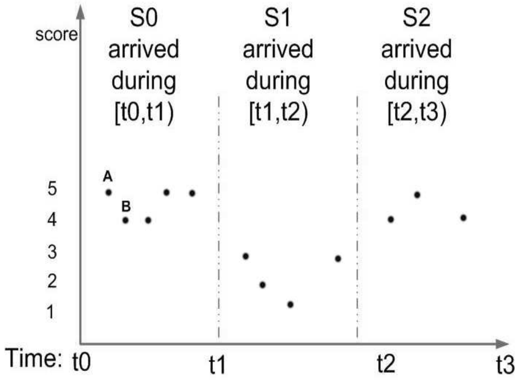


Figure 1: Concept Drift in Collaborative Filtering

item. For example, Figure 1 is used to represent the user Alice's preference for thriller movies. Data point A represents Alice's rating on one thriller: *The 39 Steps* . Data point B represents Alice's rating on another thriller: *The Bourne Identity* . For every data point, the value of x means the rating's produced time and the value of y means the score. Here, Alice's rating on *The 39 Steps* is 5 and Alice's rating on *The Bourne Identity* is 4. S_0 is the oldest data and S_2 is the newest data. Let S_i be the data that came in between time t_i and t_{i+1}. From Figure 1, we can see that Alice's preferences for thrillers are always changing. In S_0 Alice's scores on thrillers are relatively high, while in S_1 the scores are low. This means Alice's preference for thrillers went down at this period of time. However, in the time interval $[t_2, t_3)$, Alice recovered the preference for thrillers.

In summary, the challenge is: How do we deal with the changing data in collaborative filtering so that the most accurate prediction can be obtained? On the one hand, in order to reduce the influence of old data that may represent old preferences, we should use the most recent data. For example, we should only use the data in S_2. However, because of insufficient amount of data, the prediction precision is likely to be degraded. On the other hand, using all the historical data simply may also reduce the prediction precision. From Figure 1, we can see that the discrepancy between the underlying trends of S_1 and S_2 becomes the cause of the problem. To address the problem, we argue that the most recent rating reflects the user future preference most. Hence, in Figure 1, we should assign a high weight to S_0 and S_2 and a low weight to S_1. This is because S_2 represents the newest trend and, at the same time, S_0 and S_2 have similar data distribution.

The rest of the paper is organized as follows. Section 2 briefly presents some of the research literature related to our work. In section 3, we propose the new algorithm and describe different sub-tasks in detail. Section 4 presents our experimental work. It provides details of our data set, evaluation metrics, and results of different experiments and discussion of the results. The final section provides some concluding remarks and directions for future work.

2 Related Work

Recommender systems started attracting major research interest during the early nineties (Goldberg, Nichols, Oki & Terry 1992). Since that time, many techniques have been explored in collaborative filtering. Generally, they are categorized into two classes: memory-based algorithms and model-based algorithms (Breese, Heekerman & Kadic 1998). Memory-based algorithms represent the classical trend in collaborative filtering. This type of algorithm uses statistical techniques to find a set of neighbours and uses the nearest neighbours to predict the users' preferences. They include user-based collaborative filtering and item-based collaborative filtering. If the algorithm computes the similarity between different users, drawing on a set of users as nearest neighbours to do recommendation, it is called user-based collaborative filtering (Resnick, Iacovou, Suchak, Bergstorm & Riedl. 1994). Nevertheless, if the algorithm computes the similarity between different items and uses a set of items as nearest neighbours to do recommendation, it is called item-based collaborative filtering (Sarwar, Karypis, Konstan & Riedl 2001). On the other hand, model-based algorithms deploy the data to build a model that is then used for predictions. These models may use Bayesian network, clustering or association rules and most of these models are probabilistic models. In probabilistic models, users are grouped into different classes based on their rating patterns. Assuming that users in the same class have the same preferences, the probability of a target user belonging to different classes is computed. The models then use this probability to predict the target user's preferences.

Maritza L. *et al* made a comparison of different collaborative filtering algorithms: memory-based algorithm, dependency-networks algorithm, online learning algorithm and support vector machine and conducted many experiments in (L. & Perez-Alcazar 2004). The results from these experiments showed that for a wide range of conditions, memory-based algorithm outperforms support vector machine, dependency-networks and online methods. In memory-based algorithms, item-based approaches can dramatically improve the scalability of collaborative filtering and provide better quality compared to user-based approaches (Sarwar et al. 2001). Up until now, item-based collaborative filtering algorithms have been widely used in many real world applications such as at Amazon.com.

However, few works have discussed concept drift in collaborative filtering. Loren Terveen *et al* defined users' preferences using their personal history (Terveen, McMackin, Amento & Hill 2002). Kazunari Sugiyama *et al* explored a type of time-based collaborative filtering with detailed analysis of user's browsing history in one day (Sugiyama, Hatano & Yoshikawa 2004). Yi Ding *et al* proposed computing the time weights for different items in a manner that will assign a decreasing weight to old data (Ding & Li 2005). Recently, mining concept-drifting data in data streams has received growing interest. Haixun Wang et al proposed a general framework using weighted ensemble classifiers (Wang et al. 2003). Wei Fan pointed out that using old data indiscriminately helps produce a more accurate hypothesis only if there is no concept-drifting and the amount of old data chosen arbitrarily just happens to be right (Fan 2004). In (Wang et al. 2003)(Fan 2004), empirical study shows that selecting data wisely has substantial advantage over using new data or all data indiscriminately in ensuring prediction precision.

At the same time, there has been much effort dedicated to the improvement of similarity measure. In collaborative filtering, the quality of similarity function is a key issue. It influences the prediction precision to a great extent. Rong Jin *et al* proposed normalizing ratings of different users before computing similarity (Jin & Si 2004). Jonathan L. Herlocker *et al* proposed to devalue similarity weights for the users who just rated few items (Herlocker, Konstan, Terveen & Riedl 2004). Prasanna Ganesan *et al* explored new similarity measures that exploit a hierarchical domain structure (Ganesan, Garcia-Molina & Widon 2003).

However, all these approaches are based on Pearson Correlation Coefficient or Cosine Similarity measure. To date, no research has been conducted on the applicability of similarity measures in collaborative filtering. In order to produce similarity scores that better reflect the reality, we design a new similarity function for item-based collaborative filtering algorithms.

3 Proposed Algorithm

Our new algorithm is based on item-based collaborative filtering algorithms and makes an improvement on prediction accuracy. The main idea includes the follows:

First, we design a new similarity function for item-based collaborative filtering algorithms to produce similarity scores that better reflect the reality. We study the applicability of common similarity measures used in collaborative filtering and argue that in item-based collaborative filtering we look for items sharing common user preferences.

Second, we use a set of items as nearest neighbours to predict a user's preference for a specific item. To tackle the problem of concept drift, we assign the weights for items by estimating their expected prediction error on the future purchase preferences. We assume that the most recent rating is closest to the future preference. Consequently, the weight of the items can be approximated by computing the deviation of their ratings from the most recent rating.

In the following sections, we first describe the definition of collaborative filtering algorithms, and then we introduce the traditional item-based collaborative filtering algorithm. Finally, we propose our new algorithm and describe different sub-tasks of the algorithm in detail.

3.1 Item-Based Collaborative Filtering

The collaborative filtering problem can be defined as follows:

Given a database D as a tuple $< U_i, I_j, O_{ij} >$, where U_i identifies the i-th user of the system, I_j identifies the j-th item of the system and O_{ij} represents the i-th user's opinion on the j-th item, find a list of k recommended items for each user U.

In item-based collaborative filtering algorithms, an item is regarded as a vector in the user space. The whole process is divided into two phases:

Phase 1 Similarity Computation. Similarity between two items is computed by isolating the users who have rated them and then applying a similarity computation technique. There are two common techniques:

Cosine similarity: The similarity between different items is measured by computing the cosine of the angle between different vectors as:

$$sim(I_a, I_b) = \cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} = \frac{\Sigma_i^m O_{ia} \times O_{ib}}{\sqrt{\Sigma_i^m O_{ia}^2}\sqrt{\Sigma_i^m O_{ib}^2}} \tag{1}$$

where I_a identifies the a-th item of the system.

O_{ia} represents the i-th user rating on the a-th item.

Pearson correlation coefficient: The similarity between different items is measured as:

$$sim(I_a, I_b) = \frac{\Sigma_i^m (O_{ia} - \overline{O_i}) \times (O_{ib} - \overline{O_i})}{\sqrt{\Sigma_i^m (O_{ia} - \overline{O_i})^2}\sqrt{\Sigma_i^m (O_{ib} - \overline{O_i})^2}} \tag{2}$$

where $\overline{O_i}$ is the average of the i-th user's ratings

Phase 2 Preference Prediction. The prediction on an item i for a user j can be computed by using the sum of the ratings of the user to items weighted by similarity between different items as:

$$O_{ij} = \frac{\Sigma_{c=1}^k O_{ic} \cdot sim(I_j, I_c)}{\Sigma_{c=1}^k sim(I_j, I_c)} \tag{3}$$

where I_j identifies the j-th item

I_c identifies nearest neighbors of the j-th item

O_{ij} represents the i-th user's rating on the j-th item.

3.2 Recency-Based Collaborative Filtering

As discussed before, in the real world collaborative filtering is characterized by concept drift. To tackle the problem, we propose a novel algorithm namely recency-based collaborative filtering. Our approach is divided into two phases: Similarity Computation and Preference Prediction.

3.2.1 Similarity Measures

The notion of similarity is used to identify items having common "characteristics". In item-based collaborative filtering algorithms, from the perspective of intuition, we look for items sharing common user preferences. If all the user preferences on two items are very close, we consider these two items similar. On the other hand, if the user preferences on two items are different, we consider these two items not similar. Nowadays, in collaborative filtering there are two common approaches to compute the similarity between different items: Pearson Correlation Coefficient and Cosine Similarity. The details are shown in Equation 1 and Equation 2. Although these two similarity measures have been widely used in the real world, some of their properties are not applicable to item-based collaborative filtering algorithms.

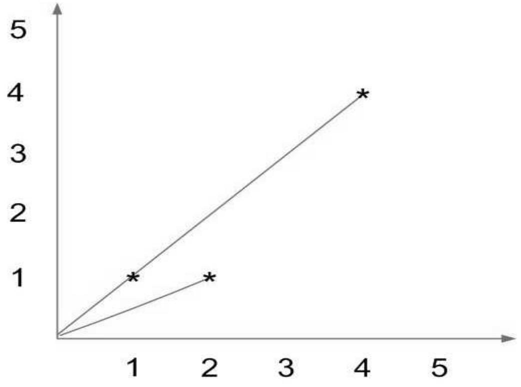
First, for Pearson Correlation Coefficient and Cosine Similarity, the similarity between different items is calculated by comparing user ratings on these items. However, ratings are determined not only by user preferences but also by the rating habits of users. There are two factors that can lead to rating variance among users with similar interests (Jin & Si 2004).

- Shift of average ratings. This problem refers to the fact that some users are more tolerant and therefore their ratings of items tend to be higher than other users. As a consequence, the average ratings for the tolerant users are higher than the strict users.
- Different rating scales. This problem refers to the factor that some conservative users tend to assign items to a narrow range of rating categories while others tend to assign items to a wide range of rating categories.

Although Pearson Correlation Coefficient takes the factor of shift of average into account by subtracting the corresponding user average, it ignores the influence of rating scale. Intuitively, we look for the items that share common user preferences rather than common user ratings. So we should convert the user ratings into the user preferences and then compare the user preferences to get more appropriate similarity scores.

Second, both Cosine Similarity and Pearson Coefficient Correlation are scale invariant. That is to say, they do not depend on the length of vector. From Figure 2 and Figure 3, we can see the property clearly.

Figure 2: Cosine Measure

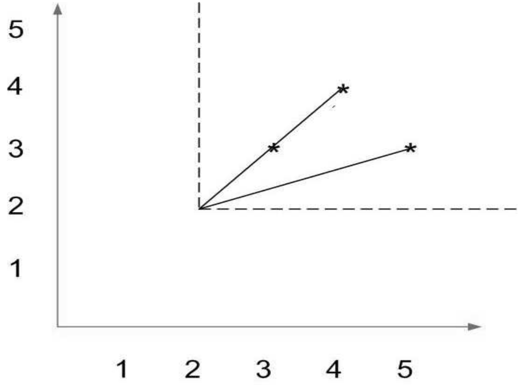


For cosine measure,

$$Sim^{(C)}(I_a, I_b) = Sim^{(C)}(\alpha I_a, I_b) \qquad \alpha > 0 \qquad (4)$$

For example, as illustrated in Figure 2, for points $x_1(1,1)$ $x_2(4,4)$ $x_3(2,1)$, we can get $Sim^{(C)}(x_1, x_3) = Sim^{(C)}(x_2, x_3)$.

Figure 3: Pearson Coefficient Correlation



For Pearson Coefficient Correlation,

$$\begin{aligned} Sim^{(P)}(I_a, I_b) &= Sim^{(C)}(I_a - \overline{I}, I_b - \overline{I}) \\ &= Sim^{(C)}(\alpha(I_a - \overline{I}), I_b - \overline{I}) \qquad \alpha > 0 \qquad (5) \end{aligned}$$

For example, as illustrated in Figure 3, for points $x_1(3,3)$ $x_2(4,4)$ $x_3(5,3)$, we can get $Sim^{(P)}(x_1, x_3) = Sim^{(P)}(x_2, x_3)$.

This property allows documents with the same composition, but different totals, to be treated identically. This makes Cosine Measure the most popular measure for text documents in information retrieval. However, this is not our designed property. In item-based collaborative filtering algorithms, the length of vector stands for how users like an item. To produce similarity scores that better reflect the reality, we design a new similarity measure for item-based collaborative filtering. The similarity between different items is computed as:

$$\begin{aligned} Sim(I_a, I_b) &= e^{-d} \\ d &= \frac{\Sigma_{i=1}^{m} |P_{ia} - P_{ib}|}{m} \\ P_{ij} &= \frac{O_{ij} - \overline{O}_i}{\sqrt{\frac{1}{N_i - 1}\Sigma_j (O_{ij} - \overline{O}_i)^2}} \end{aligned} \qquad (6)$$

Here, P_{ij} stands for the i-th user's preference for the j-th item

d stands for the average distance between the a-th item and the b-th item.

Equation 6 is derived using the following process: First we use the Gaussian normalization method to diminish the influence of shift of average ratings and different rating scales, thereby transforming the user ratings into the user preferences. The Gaussian normalization Equation is listed as follows:

$$P_{ij} = \frac{O_{ij} - \overline{O}_i}{\sqrt{\frac{1}{N_i - 1}\Sigma_j (O_{ij} - \overline{O}_i)^2}} \qquad (7)$$

Where O_{ij} stands for the i-th user's rating for the j-th item.

$\overline{O}_i$ stands for the average of the i-th user's ratings.

N_i stands for the number of the i-th user's ratings

We then use Manhattan distance to compare the user preferences on two items. Manhattan distance belongs to the family of Minkowski distances. The Minkowski distances are the standard metrics for geometrical problems. The equation is listed as follows:

$$L_p(I_a, I_b) = (\Sigma_{i=1}^{d} |I_{ia} - I_{ib}|^p)^{1/p} \qquad (8)$$

For p=1 we can get the Manhattan distance. The Manhattan distance between two items is the sum of the distances of their corresponding components. In item-based collaborative filtering, similarity between two items is computed by isolating the users who have rated them and then applying a similarity function. In this case, the Manhattan distance between two items is the sum of the distances of user preferences for two items. However, we may isolate the different number of the users when computing similarity between items. For example, there are 400 users who have rated both I_a and I_b. Accordingly, we use 400 user preferences to compute $Sim(I_a, I_b)$. There are 500 users who have rated both I_a and I_c. Accordingly, we use 500 user preferences to compute $Sim(I_a, I_c)$. To address this problem, we divide the Manhattan distance by m to get the average distance on one dimension. In Equation 6, m is the number of the users that have rated both items. Finally, we use the average distance to measure how close the user preferences for different items are. To avoid the value of zero, we use the exponential form: $s = e^{-d}$ to relate the average distance d and similarity s. From Equation 6, we can observe that the value of the similarity function is in the range of (0,1] and it reduces with the average distance. When the similarity between the two items is equal to 1, it means that user preferences on these items are very similar (identical). In contrast, when the similarity between two items is close to 0, it means the user preferences on these items are always different.

Usually, similarity measure is a function of a distance function in a given space but it does not have to be. For example, when the data is nominal data, the distance function is not applicable. The key point is that similarity measure as a function should have behaviour imitating our interpretation of similar objects.

3.2.2 Weights Based on Expected Accuracy

As discussed before, traditional approaches for item-based collaborative filtering use a set of items as nearest neighbours to predict a user's preference for a specific item. Ratings produced at different times are weighted equally. However, in the real world the user purchase interest is sensitive to time. To capture the current trend and get the most accurate prediction, we propose that in the phase of Preference Prediction, each rating is assigned a weight which is reversely proportional to the expected prediction error on the future preference. That is to say, Equation 3 is modified as:

$$O_{ij} = \frac{\Sigma_{c=1}^{k} O_{ic} \cdot sim(I_j, I_c) \cdot W_c}{\Sigma_{c=1}^{k} sim(I_j, I_c) \cdot W_c} \quad (9)$$

Where W_c represents the weight assigned to item I_c.

To achieve the appropriate weights, we need to know the actual rating of the user on the target item. However, this is unavailable. In this case, we assume the most recent rating in all the ratings on the nearest neighbours represents the current trend and is closest to the future preference. Consequently, the weight of item can be approximated by computing the deviation of the rating on the item from the most recent rating. The formula is listed as follows:

$$W_i = (1 - \frac{|O_i - O_r|}{|R|})^{\alpha} \quad (10)$$

Here, W_i represents the weight assigned to the i-th item. O_i represents the rating of the target user on the i-th item. O_r represents the most recent rating of the target user on the nearest neighbours. $|R|$ represents the rating scale. α is a parameter for a well-tuned performance.

From Equation 10, we can observe that the value of weight is in the range of [0,1] and the weight of the most recent rating is equal to 1. The more the rating on an item deviates from the recent rating, the lower the weight of the item.

3.3 An Example

The following example illustrates the whole process of our recency-based collaborative filtering algorithm. Assume that four users have rated five movies. The ratings are shown in Table 1.

Table 1: Users' Ratings on Movies

	Movie A	Movie B	Movie C	Movie D	Movie E
John	1	4	2	5	4
Helen	4	2	5	2	-
Bob	4	3	-	3	3
Alice	2	4	3	?	2

Here, the value 1 means the lowest rating and the value 5 means the highest rating. The symbol "-" means the user did not rate the movie. Suppose Alice's ratings on Movie A, Movie B, Movie C and Movie E produced on (22 9, 2004), (24 12, 2004), (22 10, 2002), (22 7, 2003) respectively. The values inside bracket represent the date Alice rated the movie. The time stamps for other ratings are also recorded. Our aim is to predict Alice's preference on movie D. The whole process is listed as follows:

Step 1: Similarity Computation

Table 2: Users' Preferences on Movies

	Movie A	Movie B	Movie C	Movie D	Movie E
John	-1.34	0.48	-0.73	1.09	0.48
Helen	0.5	-0.83	1.16	-0.83	-
Bob	1.5	-0.5	-	-0.5	-0.5
Alice	-0.78	1.30	0.26	?	-0.78

By using the Gaussian Normalization method (Equation 7), we transform the users' ratings to the users' preferences. The details are shown in Table 2.

Here, the value of preference is larger than zero, which means that the user likes the movie. The value of preference is equal to zero, which means that the user neither likes nor dislikes the movie. The value of preference is less than zero, which means that the user dislikes the movie. The lower the value of preference, the more the user does not like the movie.

We then calculate Manhattan distance between different movies using Equation 8. The details are shown in Table 3.

Table 3: Manhattan Distance

	Movie A	Movie B	Movie C	Movie D	Movie E
Movie A	0	7.23	2.31	5.76	3.82
Movie B	7.23	0	4.24	0.61	2.08
Movie C	2.31	4.24	0	3.81	2.25
Movie D	5.76	0.61	3.81	0	0.61
Movie E	3.82	2.08	2.25	0.61	0

From the Table 3, we can see that Manhattan distance is symmetric and reflexive. These properties satisfy the criteria of similarity function. However, we may isolate the different number of the users when computing Manhattan distance between different movies. For example, when we calculate Manhattan distance between Movie A and Movie B, we use four users' preferences for Movie A and Movie B: John, Helen, Bob and Alice. While when we calculate Manhattan distance between Movie C and Movie D, we just use two users' preferences for Movie C and Movie D: John and Helen. So we explore the average distance to measure how close the user preferences for different items are. The average distance between two movies is equal to Manhattan distance divided by the number of the users who have rated both movies. The details are shown in Table 4.

Table 4: Average Distance

	Movie A	Movie B	Movie C	Movie D	Movie E
Movie A	0	1.80	0.77	1.92	1.27
Movie B	1.80	0	1.41	0.20	0.69
Movie C	0.77	1.41	0	1.90	1.13
Movie D	1.92	0.20	1.90	0	0.30
Movie E	1.27	0.69	1.13	0.30	0

At last, we use the exponential form to transform Average distance into similarity score. The details are shown in Table 5.

Step 2: Selection of Nearest Neighbours

Since our aim is to predict Alice's preference on Movie D, we should select the nearest neighbours of Movie D. We compare the similarity between different items and use 0.5 as a threshold. If the similarity

Table 5: Similarity Scores

	Movie A	Movie B	Movie C	Movie D	Movie E
Movie A	1	0.1653	0.4630	0.1466	0.28
Movie B	0.1653	1	0.2441	0.8187	0.5016
Movie C	0.4630	0.2441	1	0.1496	0.323
Movie D	0.1466	0.8187	0.149	1	0.7408
Movie E	0.28	0.5016	0.323	0.7408	1

score between two items is higher than 0.5, we consider these two items similar. From row 5 in Table 5, we can see that the similarity score between Movie B and Movie D is 0.8187 and the similarity score between Movie D and Movie E is 0.7408. In this case, we select Movie B and Movie E as the nearest neighbours of Movie D.

Step 3: Weight Computation

From the perspective of statistic, we can conclude that the users' preferences for Movie B, Movie D and Movie E are similar. However, from row 5 in Table 1 we can find that Alice's ratings on Movie B and Movie E are quite different. That is to say, as time goes by, Alice changes her preference for this category of movies. As proposed before, to capture the current trend and obtain most accurate prediction, we assign different weights to Movie B and Movie E. Since Alice's rating on Movie B is the most recent rating in all the nearest neighbours and it represents Alice's current preference, we assign a weight to a movie according to the deviation of the rating on the movie from the rating on Movie B. By using Equation 10, the weights for Movie B and Movie E can be calculated as follows:

$W_{MovieB} = (1 - \frac{|4-4|}{5})^1 = 1$

$W_{MovieE} = (1 - \frac{|2-4|}{5})^1 = 0.6$

In this case, $|R|$ is equal to 5. We assign 1 to the parameter α .

Step 4: Preference Prediction

The prediction of Alice's rating on Movie D can be computed by using Alice's ratings on Movie B and Movie D weighted by similarity between them and expected accuracy weight as:

$O = \frac{4 \times 0.8187 \times 1 + 2 \times 0.7408 \times 0.6}{0.8187 \times 1 + 0.7408 \times 0.6} = 3.3$

3.4 Building the Model

Our proposed algorithm is based on item-based collaborative filtering. In the phase of Similarity Computation, we design a new similarity measure to get more appropriate similarity scores. In the phase of Preference Prediction, we explore the weights based on expected accuracy for different items to tackle the problem of concept drift in collaborative filtering. The input to this algorithm is the N × M user-item matrix C, N × M user-time matrix T that represents the time users' opinions on items were produced, a parameter n that specifies the number of item-to-item similarities that will be stored for each item and a parameter l that denotes the number of recommendation items. If the i-th user has no rating on the j-th item in the system, the values of C_{ij} and T_{ij} are both equal to zero. The output is an N × l matrix N that stores the N × l recommendation items. In it, every row represents every user. For each user there are l recommendation items. The algorithm is shown in Figure 4.

```
NewProposedCollaborativeFiltering (C,T,n,l)
1. Matrix M←ComputeItemSimilarity(C,n);
2. FindNearestNeighbours(C,n);
3. FindRecentRating(T);
4. GetWeights(C);
5. Matrix N←
6. PredictRecommendationItems(l);
```

Figure 4: Recency-Based Collaborative Filtering Algorithm

4 Experiments

We have conducted a set of experiments to examine the performance of our new algorithm. Particularly, we address the following three issues:

1 How do the similarity measures influence the prediction precision?

The quality of similarity function is a key issue in collaborative filtering. That is to say, the selection of similarity measures influences the prediction precision to a great extent. In this experiment, we compare our new designed similarity measure with two common similarity measures used in item-based collaborative filtering to examine the impact of similarity measures on the final performance of collaborative filtering algorithms.

2 How does concept drift influence the prediction precision?

As discussed before, in the real world collaborative filtering is characterized by concept drift. The problem of concept drift degrades the prediction precision of collaborative filtering. To tackle the problem, we proposed using weights based on expected accuracy for different items. In this experiment, we compare our approach with no-weight item-based approach. These two approaches are both incorporated into the Pearson Correlation Coefficient method.

3 How is our new algorithm compared to the existing collaborative filtering algorithm?

Our algorithm that includes the new similarity measure and weights based on expected accuracy is compared to the traditional item-based algorithm.

4.1 Experiment Design

We use two datasets in our experiments: EachMovie [1] and GroupLens [2]. EachMovie and GroupLens have been the most widely used common datasets in collaborative filtering research projects. EachMovie was collected during 18 months where 72,916 users rated 1628 movies (Herlocker et al. 2004). GroupLens consisted of 1,000,209 ratings for 3900 movies by 6040 users. The global statistics of these two datasets used in our experiments are showed in Table 6.

We alter the training size to be the first 60, 200 or 400 users for training. At the same time, we also utilize the protocol, *All But One*. In *All But One*, the newest rated items for each user are used for testing. By varying the number of training users, we can test our proposed algorithm for different configurations. In all the experiments the number of the nearest neigbours is set to 30 and α is set to 0.7. The evaluation metric used in our experiments is the

[1] www.research.compaq.com/SRC/eachmovie

[2] www.cs.usyd.edu.au/Research/GroupLens/data/million

Table 6: characteristics of EachMovie

Name of database	EachMovie	GroupLens
Number of Users	1394	574
Number of Items	1628	100
Avg. ♯ of rated Items/User	145.6	31.7
Number of Ratings	6	5

mean absolute error (MAE). MAE is a popular metric in collaborative filtering. It computes the average absolute deviation of recommendations from their true user-specified values. The MAE can be computed as:

$$MAE = \frac{\Sigma_{i=1}^{N} |p_i - q_i|}{N} \tag{11}$$

Here N identifies the number of the user's ratings. p_i identifies the predicted rating for the i-th item. q_i identifies the user's true rating for the i-th item.

4.2 Experiment(1): Impact of Similarity Measures

In the first experiment, we vary the similarity measures and compare our new similarity function with two common used similarity measures: Pearson Correlation Coefficient and Cosine Measure. The results are demonstrated in Table 7 and Table 8. Obviously, our new similarity function can substantially improve the prediction precision of collaborative filtering compared to Pearson Correlation Coefficient and Cosine Measure.

Table 7: MAE using different similarity measures on EachMovie. (*AllButOne*) A smaller value means a better performance

Training Users Size	similarity measures	MAE
60	Cosine Measure	0.2301
	Pearson Correlation	0.1808
	New Similarity	0.1720
200	Cosine Measure	0.2038
	Pearson Correlation	0.1982
	New Similarity	0.1855
400	Cosine Measure	0.2024
	Pearson Correlation	0.1856
	New Similarity	0.1760

Table 8: MAE using different similarity measures on GroupLens. (*AllButOne*) A smaller value means a better performance

Training Users Size	similarity measures	MAE
60	Cosine Measure	0.8549
	Pearson Correlation	0.8104
	New Similarity	0.8032
200	Cosine Measure	0.8873
	Pearson Correlation	0.8508
	New Similarity	0.8418
400	Cosine Measure	0.8901
	Pearson Correlation	0.8637
	New Similarity	0.8081

4.3 Experiment(2): Impact of Concept Drift

In the second experiments, we compare our weighted approach with no-weighted item-based collaborative filtering. These two approaches are both incorporate into the Pearson Correlation Coefficient method. The results are shown in Table 9 and Table 10. From Table 9 and Table 10, we can see that our weighted approach outperforms no-weighted item-based collaborative filtering in prediction precision.

Table 9: MAE using weights on EachMovie. (*AllButOne*) A smaller value means a better performance

Training Users Size	method	MAE
60	no-weighted	0.1808
	weighted	0.1775
200	no-weighted	0.1982
	weighted	0.1967
400	no-weighted	0.1856
	weighted	0.1841

Table 10: MAE using weights on GroupLens. (*AllButOne*) A smaller value means a better performance

Training Users Size	method	MAE
60	no-weighted	0.8104
	weighted	0.7883
200	no-weighted	0.8508
	weighted	0.8125
400	no-weighted	0.8637
	weighted	0.7707

4.4 Experiment(3): Comparison to the Classic Item-based Algorithm

In this experiment, we compare our recency-based collaborative filtering algorithm to the classic item-based collaborative filtering algorithm. The results are presented in Table 11, Table 12, Figure 5 and Figure 6. Obviously, our new algorithm is able to boost the prediction precision for all configurations.

Table 11: MAE using different algorithms on EachMovie. (*AllButOne*) A smaller value means a better performance

Training Users Size	method	MAE
60	traditional	0.1808
	new proposed	0.1711
200	traditional	0.1982
	new proposed	0.1845
400	traditional	0.1856
	new proposed	0.1735

4.5 Complexity Analysis

Our proposed algorithm's scalability is that it can create the expensive similar items table offline. The offline computation of the similar table is extremely time intensive, with $O(N^2M)$ as worst case. Here, N represents the number of items. M represents the number of users. The online component is just looking up similar items of the user's ratings. So our algorithm scales independently of the number of users and items. The algorithm is fast even for extremely large data sets.

5 Conclusions

In this paper, we present a new collaborative filtering algorithm namely recency-based collaborative filtering. Unlike the traditional approaches, we design a new similarity measure and propose using weights based on expected accuracy for items. The main contributions are:

Figure 5: MAE using different algorithms on EachMovie

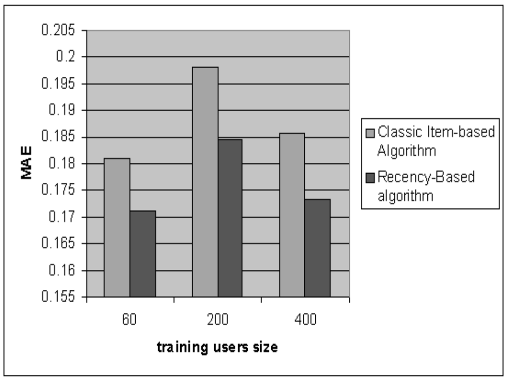


Table 12: MAE using different algorithms on GroupLens. (*AllButOne*) A smaller value means a better performance

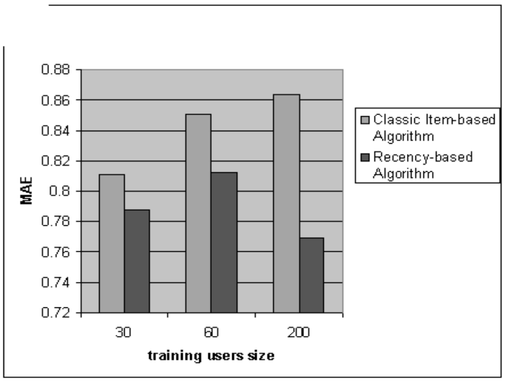
Training Users Size	method	MAE
60	traditional	0.8104
	new proposed	0.7880
200	traditional	0.8508
	new proposed	0.8130
400	traditional	0.8637
	new proposed	0.7695

- tackled the problem of concept drift in collaborative filtering
- Designed and tested a more appropriate similarity function for item-based collaborative filtering algorithms in the context of concept drift

Our experimental results have shown that the new algorithm can substantially improve the prediction precision of item-based collaborative filtering algorithms.

Our further work is to study collaborative filtering algorithms on streaming data. This issue brings us a new challenge that data are all contained not in a database ready for random access but are seen once only from online resources.

Figure 6: MAE using different algorithms on GroupLens



References

Ali, K. & Stam, W. v. (04), Tivo: making show recommendations using a distributed collaborative filtering architecture, *in* 'Proceedings of Conference on Knowledge Discovery in Data', Seattle, WA, USA, pp. 394 – 401.

Breese, J. S., Heekerman, D. & Kadic, C. (1998), Empirical analysis of predictive algorithms for collaborative filtering, *in* 'Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence(UAI)'.

Ding, Y. & Li, X. (2005), Time weight collaborative filtering, *in* 'Conference on Information and Knowledge Management, ACM CIKM, accepted paper'.

Fan, W. (2004), Systematic data selection to mine concept-drifting data streams, *in* 'Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining', Seattle, WA, USA, pp. 128 – 137.

Ganesan, P., Garcia-Molina, H. & Widon, J. (2003), 'Exploiting hierarchical domain structure to compute similarity', *ACM Transactions on Information Systems (TOIS)* **Volume 21**(Issue 1), 64 – 93.

Goldberg, D., Nichols, D., Oki, B. M. & Terry, D. (1992), 'Using collaborative filtering to weave an information tapestry', *communications of the ACM* **35**(12), 61–70.

Herlocker, J. L., Konstan, J. A., Terveen, L. G. & Riedl, J. T. (2004), 'Evaluating collaborative filtering recommender systems', *ACM Transactions on Information Systems (TOIS)* **Volume 22**(Issue 1), 5 – 53.

Jin, R. & Si, L. (2004), A study of methods for normalizing user ratings in collaborative filtering, *in* 'Annual ACM Conference on Research and Development in Information Retrieval', pp. 568 – 569.

L., Maritza, C. & Perez-Alcazar, J. d. J. (2004), A comparison of several predictive algorithms for collaborative filtering on multi-valued ratings, *in* 'ACM symposium on Applied computing', pp. 1033 – 1039.

Linden, G., Smith, B. & York, J. (2003), 'Amazon.com recommendations item-to-item collaborative filtering', *IEEE Internet Computing* pp. 76–80.

Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P. & Riedl., J. (1994), Grouplens: an open architecture for collaborative filtering of netnews, *in* 'ACM conference on Computer supported cooperative work', pp. 175 – 186.

Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. (2001), Item-based collaborative filtering recommendation algorithms, *in* 'Proceedings of International Conference on World Wide Web', pp. 285 – 295.

Sugiyama, K., Hatano, K. & Yoshikawa, M. (2004), Adaptive web search based on user profile constructed without any effort from users, *in* 'Proceedings of the 13th international conference on World Wide Web', New York, NY, USA, pp. 675 – 684.

Terveen, L., McMackin, J., Amento, B. & Hill, W. (2002), Specifying preferences based on user history, *in* 'Conference on Human Factors in Computing Systems', Minneapolis, Minnesota, USA, pp. 315 – 322.

Wang, H., Fan, W., Yu, P. S. & Han, J. (2003), Mining concept-drifting data streams using ensemble classifiers, *in* 'Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining', Washington, D.C, pp. 226 – 235.

Document Generality: its Computation for Ranking

Xin Yan **Xue Li** **Dawei Song**

School of Information Technology and Electrical Engineering,
University of Queensland
ITEE, University of Queensland, QLD 4072, Australia
Email: {yanxin, xueli}@itee.uq.edu.au
Knowledge Media Institute
The Open University
Walton Hall, Milton Keynes, MK7 6AA, United Kingdom
Email: dawei_song2005@hotmail.com

Abstract

The increased variety of information makes it critical to retrieve documents which are not only relevant but also broad enough to cover as many different aspects of a certain topic as possible. The increased variety of users also makes it critical to retrieve documents that are jargon free and easy-to-understand rather than the specific technical materials. In this paper, we propose a new concept namely document generality computation. Generality of document is of fundamental importance to information retrieval. Document generality is the state or quality of document being general. We compute document generality based on a domain-ontology method that analyzes scope and semantic cohesion of concepts appeared in the text. For test purposes, our proposed approach is then applied to improving the performance of document ranking in bio-medical information retrieval. The retrieved documents are re-ranked by a combined score of similarity and the closeness of documents' generality to that of a query. The experiments have shown that our method can work on a large scale bio-medical text corpus OHSUMED (Hersh, Buckley, Leone & Hickam 1994), which is a subset of MEDLINE collection containing of 348,566 medical journal references and 101 test queries, with an encouraging performance.

Keywords: generality, document ranking, re-ranking

1 Introduction

Generality is the state or quality of being general, according to its definition in Webster Dictionary. A document with high generality might be general or broad in its meaning such as tutorials and reviews. A document with low generality might be specific and narrow in its meaning, for example a journal paper talking about a specific research problem. Generality retrieval is an information searching behavior to find documents which are both relevant to the query and above a certain degree of generality. The trend of generality retrieval is resulted by the information explosion and the popularity of WWW searching. Generality of documents should act as an importance role in information retrieval.

On the one hand, information explosion somehow increases not only the quantity of information but also the variety. For instance a query for general AIDS information in PubMed[1], a medical searching service, may bring some troubles. Thousands of documents may be retrieved in a wide range such as treatment, drug therapy, transmission, diagnosis and history. User may need to have a glance of the topic on the whole, a kind of documents which are not only relevant but also broad enough in meanings to cover as many different aspects of a certain topic as possible, to be retrieved. In this example, user may request review articles of AIDS information.

On the other hand, the growing popularity of WWW information retrieval makes domain-specific information retrieval open to the public. Easy-to-understand and jargon free information is needed by users with insufficient domain knowledge. For example, the patient education materials and tutorials of diseases in bio-medical domain are often requested by the public rather than those materials which are technical and specific.

However, to the best of our knowledge, there is a lack of solutions in literature to satisfy the stringent requirement of generality-based retrieval. The first problem we need to solve is how to compute document generality. In this paper we develop a novel ontology-based document generality computation method via analyzing the scope and semantic cohesion of a document. Our method is then applied to improving the performance of document ranking in bio-medical information retrieval.

Document ranking is well known to be a critical component in information retrieval system. It is the computer judgements of how relevant a document is to a query comparing with other documents retrieved by the same query. Due to the quantity of search result and the limitation of user's time and patience, it is impractical for user to review all the retrieved documents and judge their relevances. In what order to present retrieved documents is a key problem in IR research area.

Based on an assumption that users have a sequential browsing behavior, document ranking determines the presentation order of those retrieved documents. The order is based on how close or relevant a document is to a query. In general, relevance is computed by similarity functions. In traditional IR models such as the vector space model (Salton, Wong & Yang 1975), documents are represented by vectors of keywords and ranked by how similar the document vectors are to the query vector. Two widely used similarity functions are cosine similarity and inner product.

Generality retrieval challenges the traditional document ranking since traditional ranking process is insufficiently based on similarity only. For a simple query "AIDS" in PubMed, we assume that a

Copyright (c) 2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

[1] http://pubmed.gov

user's information need is to retrieve general information about AIDS. One of the documents retrieved by PubMed, is a specific research paper namely *Multiple Dimensions of HIV Stigma and Psychological Distress Among Asians and Pacific Islanders*. Another article about general AIDS information, *HIV/AIDS: A Minority Health Issue*, is also retrieved. As a result of similarity-based ranking, the former document is ranked much higher than the latter one, whereas, the latter one is closer to user's information need for general AIDS information.

Based on the above discussions, we argue that the factor of "generality" should be taken into account in a document ranking process. Our purpose is to improve the query performance of domain specific (biomedical literature in this paper) information retrieval by re-ranking retrieved documents on generality.

In order to re-rank retrieved documents by generality, we need to know if the generality ranking is required. In practice, there are three ways to determine user's need of generality-based retrieval: manual, semiautomatic and automatic.

Manual Detection of Query Generality User explicitly labels query as general or specific to indicate if the general or specific documents are required.

Semiautomatic Detection of Query Generality User uses a set of pre-defined words such as "review", "introduction" and "tutorial" to test query. User's feedback is needed after retrieval in order to verify user's need of generality retrieval.

Automatic Detection of Query Generality System automatically estimates the generality of query as if it were a document.

In our research, we assume that user's needs of generality retrieval is pre-determined by the IR system through any of the three ways we mentioned above. The focus of our work is to investigate on how to rank documents by their generality.

A novel ontology-based document re-ranking framework is proposed. Based on the hypothesis that there is no dependence between the document generality and its similarity to a query, the documents are ranked by a combined score of similarity and the closeness of documents' generality to the query's. Experiments have been conducted on a large scale biomedical text corpus, OHSUMED (Hersh et al. 1994), which is a subset of MEDLINE collection containing 348,566 medical journal references and 101 test queries. By submitting those queries to our IR baseline system, the similarity of retrieved documents to queries are computed and scored. The correlation analysis between document generality and its similarity score further proves our hypothesis of the independent relationship between generality and similarity. The comparison of retrieval performances before and after re-ranking process reveals that our approach demonstrates an encouraging improvement on technical generality retrieval performance with a positive impact to the overall performance of 101 queries.

The remainder of this paper is organized as follows: Section 2 presents related work. Section 3 gives a detailed definition of generality. Our methods for re-ranking documents on generality are proposed in Section 4. Section 5 reports experimental setup and results. Section 6 concludes the paper and addresses future research directions.

2 Related Work

To our knowledge, no researches directly focusing on generality computation are currently available. The studies about so-called "aspect retrieval" and "subtopic retrieval" are mostly close to our work. Here we regard the related work in terms of two categories: the interactive generality retrieval and the automatic generality ranking. The former is about how the generality is concerned in an interactive IR process while the latter is about how generality is associated in a ranking process.

2.1 Interactive Generality Retrieval

Interactive generality retrieval, or so-called "aspect retrieval", is studied in the interactive track of TREC-6,7,8 (Swan & Allan 1998, Robertson, Walker & Beaulieu 1999, Hersh 2000). The purpose of these studies is to help user retrieve documents covering as many different aspects of a topic as possible in a limited time. An aspect is defined as one of the many possible answers to the topic (Over 1999). Aspects of topics and documents in the collection are defined and judged by human assessors in order to evaluate the performance of aspect retrieval. In the area of aspect retrieval, researches are mainly focused on user's searching behavior and the interface of retrieval system.

We regard the aspect retrieval problem as a simplified version of the generality retrieval problem since the intuition is that the more aspects broadly covered by a document, the more general the document is. However, generality is richer than the aspect retrieval. Generality implies not only the broadness but also the deepness of a document in its meaning. Therefore our research will broaden the aspect retrieval into a problem of generality.

Furthermore, in order to help user's generality retrieval, automatic methods need to be developed. Given a query, automatic generality rankings is a process of ranking the retrieved documents by systematically estimating their generality. In the case of a large number of documents returned for a query, it is insufficient to improve the efficiency of the generality retrieval by improving the interface between user and the retrieval system. Automatic generality retrieval may be more efficient to help user to sort out documents with the consideration of generality. In next subsections, some researches closely related to automatic generality ranking are discussed.

2.2 Automatic Generality Ranking

Studies concerning automatic generality ranking aim at finding approaches to automatically rank general documents more closely to a query.

The study of subtopic retrieval (Zhai, Cohen & Lafferty 2003) seeks an automatic solution for the aspect retrieval problem we mentioned above. Zhai et al. addressed that there is a need (e.g literature survey) to find documents that "cover as many different subtopics of a general topic as possible" (Zhai et al. 2003). Given a set of documents retrieved by a baseline IR system, subtopic retrieval method re-ranks those documents by their generality feature and their relevance to the query. Statistical language models and maximal marginal relevance (Carbonell & Goldstain 1998) were used to perform subtopic retrieval.

Another research (Liu, Zhang, Chen, Lyu & Ma 2004) namely "affinity rank" is close to the study of subtopic retrieval. Affinity rank is based on the assumption that in a vector space model, "the more neighbors a document has, the more informative it is; moreover, the more informative a document's neighbors are, the more informative it is as well" (Liu et al. 2004). Information richness was modeled by computing the principal eigenvector of a matrix M where each entry represents the value of a similarity function of each pair of documents in the vector space model.

The common feature of affinity ranking research and subtopic retrieval study is that document generality is based on the overall statistical properties of document in the collection rather than the concept generality. Concept generality is defined in our work as the generality of individual terms in the context of a given ontology. In WordNet, for example, hypernyms are defined as those concepts being more general than others; hyponyms are defined as those concepts being more specific than others. Since documents are composed of terms, document generality is consequently affected by the concept generality of all its terms.

Allen and Wu (Allen & Wu 2002) defined document generality as the mean generality of terms in the documents. For example, 64 selected words were determined manually as a reference collection for computing the generality. Half of the words in the collection were regarded as general and the other half as concrete. The joint entropy measure was used to verify that general terms were more related to each other than concrete terms. Thus, through the relatedness computation between the terms in documents and in those 64 terms of the reference collection, the generality of the terms in documents could be calculated.

However, some problems still remain unsolved. First, the generality of the terms in the reference list is determined by human experts. This is computationally infeasible to deal with a large number of words. Particularly if a term does not appear in the reference list, it is excluded from the generality computation. This is impractical in many applications that have a large vocabulary. It is expected that automatic methods can be developed to measure the concept generality objectively and efficiently for documents with a large domain-specific vocabulary. Secondly, not only the statistical term relatedness, but also the semantic relations between terms should be taken into account. Sometimes general terms may have low relatedness if they cross different domains. In the area of biomedical information retrieval, for example, a stomach medicine may be semantically related to a skin medicine in terms of their generality. However, they may not have a statistical relatedness at all, simply due to no co-occurrence in the text corpus. Third, in (Allen & Wu 2002), the generality was ranked for merely six documents and then manually judged for the evaluation. For dealing with large collections, this is obviously impractical. Finally, user generally would not prefer a document with high generality but low relevance to the query. Combining document generality with query generality should be considered. In next subsection, we briefly review some studies related to query generality.

2.3 Query Generality

To our knowledge, no researches considering query generality in document ranking or re-ranking process are currently available. Some definitions (He & Ounis 2004), (Plachouras, Cacheda, Ounis & Rijsbergen 2003), (Van Rijsbergen 1979) about query generality have been made long before the studies of document ranking. They mainly focus on the overall generality of retrieval rather than the generality of individual documents against a query. Van Rijsbergen (Van Rijsbergen 1979), (Plachouras et al. 2003) regarded query generality as "a measure of the density of relevant documents in the collection". Derived from Van Rijsbergen's definition, He and Ounis (He & Ounis 2004) defined query generality as:

$$\omega = -log(\frac{N_Q}{N}) \tag{1}$$

where N_Q is the total number of documents containing at least one query term and N is the total number of documents in the collection.

Based on these definitions of query generality, the more documents a query is related, the more generality the query has.

However, it is not sufficient to quantify the query generality purely based on this method. Let's consider two queries Q_1 "AIDS review" and Q_2 "SARS review". Q_1 requires literature reviews about AIDS, T_2 requires reviews about SARS, a newly discovered disease. In PubMed, Q_1 may result 19,311 documents. Whereas, there are only 396 documents returned by Q_2. Since it is hard to count the exact size of whole PubMed database, we assume that N is 11,000,000. According to Equation 1 the generality of Q_1 is around 6.3450. The generality of Q_2 is around 10.2320. Is Q_2 more general than Q_1? The answer is probably "no", because "SARS" is a newly discovered disease which has just less related documents in the collection than "AIDS".

In conclusion, there are some major differences between existing related work in the literature and our proposed approach.

1. We assume that the relevance judgment of a document is independent to that of the others retrieved by the same query. In the study of subtopic retrieval, relevance between two documents may depend on which documents a user sees the first.

2. We broaden the research problems of aspect retrieval, subtopic retrieval and affinity rank and propose the concept "generality" in document ranking.

3. Semantics inherence in the documents is considered in our research. We measure the ontology based semantic relationships of document concepts in order to compute generality. In literature, only statistical methods were used.

4. We consider both document generality and query generality. The documents are re-ranked by a combined score of similarity and the closeness of documents' generality to the query's. In literature, only document generality was considered.

In next section, we introduce the details of our ideas on re-ranking by generality.

3 Different Types of Generality

In our research, we divide generality into two categories based on user's information needs: technical generality and non-technical generality.

Technical Generality. How broad a document is for describing a certain topic. Documents with high technical generality are divided into two subcategories:

1. Summary
2. Review

Non-technical Generality. How deep a document is for describing a certain topic. Documents with high non-technical generality are divided into two categories:

1. Introduction
2. Tutorial

Technical generality should be considered when there is a need to retrieve summaries and technical review articles which broadly describe a certain topic. Non-technical generality should be considered when there is a need to retrieve introductive documents or tutorials that are jargon free and easy to understand. In this paper, we mainly focus on the study of technical generality, that is, on how to measure the document generality according to its broadness.

4 Proposed Approach

The intuition of our proposed computational generality is given as follows:

- *Document Scope* (DS) - We consider document as a collection of terms. The scope of a document is regarded as a coverage of terms onto the concepts in MeSH ontology. The more concepts matched within the MeSH the more specific the document is. Also, within a MeSH tree, the deeper the concepts appear, the more specific the document is.
- *Document Cohesion* (DC) - When there is a focused topic or theme discussed in a document, the terms are closely correlated in a certain context. The cohesion of a document is regarded as a computation of the associations between the concepts found in the MeSH tree. It reflects the frequencies of the associated concepts that appear in the MeSH ontology. The more closely the concepts are associated, the more specific the document is.

We formulate the problem of document ranking with generality as that: given a query Q , a rank $(R, \leq)$, $R = \{d_1, \dots d_n\}$ which is retrieved by Q, the similarity function $Sim(Q, d_i)$,

1. find a function $Gen(Q, d_i)$ to return the closeness of generality between d_i and Q
2. re-rank $(R, \leq)$ to $(R', \leq)$ so that $d_i \leq d_j \iff f'(Sim(Q, d_i), Gen(Q, d_i)) \leq f'(Sim(Q, d_j), Gen(Q, d_j))$ where $d_i, d_j \in R'$. f' is a function considering both $Sim(Q, d_i)$ and $Gen(Q, d_i)$.

We approach the generality ranking problem from two perspectives. The first is to consider the query generality. We believe that generality ranking depends on both query generality and document generality. To a specific query (i.e., a query with low generality), it is not proper to simply rank general documents higher than the specific ones. The second consideration is the semantics in documents. For instance, "HIV" is more specific than "virus" in terms of a given domain knowledge. The statistical analysis cannot reflect the semantic relationship between them.

A query can be regarded as a short document. In the same way, a query is to be computed for its generality as though it were a document. Then the documents are re-ranked by comparing the closeness of documents' generality scores to the query's.

On the other hand, the semantics of documents can be computationally gripped in terms of ontology. In our work, we use bio-medical documents together with an ontology database called MeSH hierarchical structure (or MeSH tree) in bio-medical domain. Our purpose is to compute generality of text by considering the semantic properties and relations of terms appearing in the MeSH tree. For example, stomach medicine and skin medicine both belong to "Chemicals and Drugs" no matter how different their usages are. Here we regard the terms in text which can be found in MeSH ontology as domain specific concepts or MeSH concepts. The terms in text which cannot be found in MeSH ontology are referred to non-ontology concepts.

In following subsections, we will describe the MeSH hierarchical structure and propose a method to identify MeSH concepts from text. We then present our approach to computational generality of documents.

4.1 Ontology: MeSH Hierarchical Structure

All the headings used to index OHSUMED (Hersh et al. 1994) documents are well organized in a hierarchical structure namely MeSH tree. Figure 1 is a fragment of the MeSH tree.

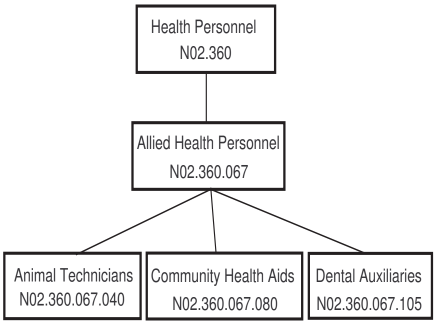


Figure 1: A Fragment of MeSH tree

The MeSH terms are numbered and organized based on a broader/ narrower relationships in the tree. In this example, the heading "Allied Health Personnel" is a kind of "Health Personnel" and "Community Health Aides" is a kind of "Allied Health Personnel".

Moreover, MeSH provides entry terms which may act as synonyms of a certain heading. In the given document example, the heading "Allied Health Personnel" has the following entry terms: "Allied Health Personnel", "Allied Health Paramedics", "Paramedical Personnel", "Specialists, Population Program" and "Paramedics". With entry terms, it is possible to take advantage of semantic relation between terms to identify synonyms.

4.2 Computation of Document Generality

4.2.1 Concept Identification

In order to use MeSH ontology to extract the semantic relations between terms, the MeSH concepts in the text corpus must be recognized. An algorithm of concept identification is proposed to match single or compound(noun) terms in the corpus with the concepts in the MeSH tree.

The algorithm is mainly concerned with the subsumed terms: a part of a compound term may match with a MeSH concept. For example, the compound "Plant Viruses" contains the term "Viruses". If we stop the concept identification process after a match of "Viruses" in the MeSH tree is found, then "Plant" will be mistakenly regarded as a term out of domain ontology. Indeed, "Plant Viruses" is also a MeSH concept. We solve the problem by introducing the conceptual marking tree (CMT) that is derived from the MeSH tree. The structure of a node in CMT is shown in Figure 2. A concept C is a sequence of terms $\{T_1 \ldots T_n\}$, where n is the length of C. The occurrence information of individual terms is stored separately in the cells of an array. In cell T_i, $0 \leq i \leq n$, we use P_i to store a set of position values $\{p_{i1} \ldots p_{im}\}$, where m is the term frequency of T_i in a document. p_{ij} $(0 \leq j \leq m)$ is the term position of the jth occurrence of T_i. The term position p_{ij} indicates that there are $(p_{ij} - 1)$ terms before T_1 from the beginning of a document.

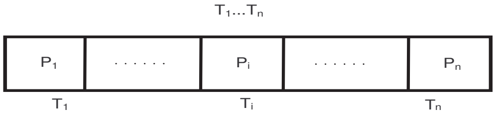


Figure 2: Data Structure of a Node in CMT

There are 3 steps to perform the conceptual marking for a document.

1. Pick up a term t which is the k-th term counted from the beginning of the document (initially $k = 0$).
2. Locate t in CMT.
3. Assign the position value k to p_{ij} in P_i. j will be increased by one automatically when a new element is added to P_i.
4. Increase k by one, then goto step 1.

For example, the following is a one-sentence document just containing one sentence:

```
Over 390 individual descriptions of
plant viruses or virus groups are
provided.
```
[2]

In this example, "plant viruses" and "viruses" are all MeSH concepts. We assume that stemming has been done so that "viruses" can be identified as "virus". After the CMT is created for this document, the concept "plant viruses" in CMT have two cells, T_1 = "plant", T_2 = "viruses". $p_{11} = 6$, $p_{21} = 7$, $p_{22} = 9$. The concept "viruses" has one cell T_1 = "viruses" where $p_{11} = 7$, $p_{12} = 9$.

After marking CMT, if it is always true that $p_{(i-1)j} = p_{(i)j} + 1$ $(1 \leq i \leq m)$, then the concept C is identified as a candidate concept at its jth occurrence in the document. If no other candidate concepts can be found with more compound terms than concept C in the same place of the document, then C is identified as the concept at its jth occurrence in the document. For the above example, we may find that the MeSH concept "viruses" may be identified as the candidate concept in position 7 and 9. However, the concept "plant viruses" has $p_{11} = p_{21} + 1$. Furthermore, it has two constituent terms but the concept "viruses" only has one. Thus it is "plant viruses" rather than "virus" which is identified as the concept at position 6.

4.2.2 Computing Document Scope

Document scope is about how broad or vague a document is for describing a certain topic. It is an important feature of document generality. Consider the following two definitions of SARS. Definition 1 comes from ABOUT [3], a web information service for daily life. Definition 2 is an official definition from the Department of Health in Hong Kong [4].

1. A viral respiratory illness that was recognized as a global threat in March 2003.
2. A viral respiratory infection caused by a coronavirus (SARS-CoV).

In above definition 2 we may identify three MeSH concepts: "respiratory infection", "coronavirus" and "SARS-CoV". However, in definition 1 which is for the general public, no MeSH concept is found. "Respiratory illness" is used to broadly describe SARS rather than a more narrowed concept "respiratory infection".

We mentioned that the scope of a document is regarded as a coverage of terms onto the concepts in MeSH ontology. The more concepts matched within the MeSH the more specific the document is. Also, within a MeSH tree, the deeper the concepts appear, the more specific the document is. In our computation of document scope, both MeSH concepts and non-ontology concepts in document are considered. Firstly a mean function of tree depths of all concepts in document is proposed to calculate document scope. The depth of a MeSH concept is measured by the distance between that concept and the root of the MeSH tree. The tree depth of a non-ontology concept in MeSH tree is zero. Secondly, we normalize the scope function within the range of 0 and 1.

It is often the case that a document contains a large percentage of non-ontology concepts but just a small percentage of MeSH concepts. This kind of documents may have a low average tree depth of all concepts and may be close to each other in terms of their computed scope values. Therefore, we need to make the scope function to be more sensitive to documents with low average tree depths compared with that of the documents with high average tree depths. In our research, we select an exponential function that can well satisfy our requirement for the distribution of scope function values.

$$Scope(d_i) = e^{-\left(\frac{\sum_{i=1}^{n} depth(c_i)}{n}\right)} \qquad (2)$$

In Equation 2, n is the total number of concepts of both MeSH concepts and general concepts. Moreover, stop words are excluded in this example. Function $depth(c_i)$ is to get the tree depth of concept i in the MeSH tree. As to a document which contains only non-ontology concepts, its document scope is 1, the maximum value. For a document which has maximum average tree depth of all its MeSH concepts, its scope is e^{-11}, the minimum value. The time complexity of scope-based ranking is $O(m \times n)$, m is the number of retrieved documents, n is the average concepts in those documents.

There are two typical examples where the concepts in documents may have different distributions in MeSH tree in terms of their subsumption relationships. Concept A subsuming concept B in the MeSH tree indicates that A is one of the parent nodes of B. The followings are illustrations of our scope algorithm in both examples.

Example One

A document may contain MeSH concepts that have no subsumption relationship between each

[2] http://www.dpvweb.net/dpv/index.php

[3] http://about.com

[4] http://www.info.gov.hk

other in the MeSH tree. In Figure 3, there is a piece of MeSH tree. Every labeled node is a MeSH concept. Suppose that d_i and d_j are two documents in the document collection. d_i is more general than d_j. Each of them contains only two concepts. The concepts o and p in d_i have matches found in the MeSH tree (the darkened nodes). The concepts k and h in d_j have matches found in the MeSH tree too. According to our algorithm, the average tree depths of d_i and d_j are respectively 3 and 4. The scope of d_i is 0.0498, which is greater than 0.0183, the scope of d_j.

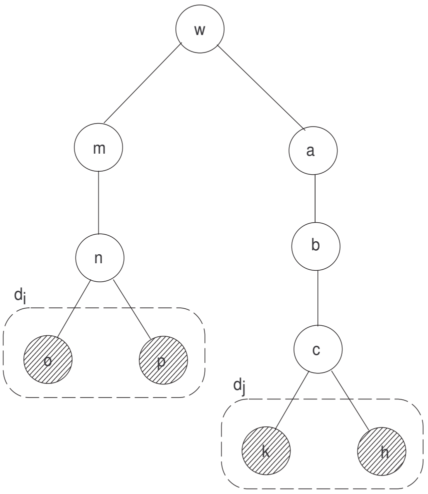


Figure 3: d_i and d_j with different document scope

Example Two

A document may contain MeSH concepts that have subsumption relationship between each other in the MeSH tree. In Figure 4, there is a piece of MeSH tree. Every labeled node is a MeSH concept. Suppose that d_i and d_j are two documents in the document collection. d_i is more general than d_j. Each of them contains only two concepts. The concepts m and n in d_i have matches found in the MeSH tree (the darkened nodes). The concepts c and h in d_j have matches found in the MeSH tree too. According to our algorithm, the average tree depths of d_i and d_j are respectively 1.5 and 3.5. The scope of d_i is 0.2231, which is greater than 0.0302, the scope of d_j.

In above SARS example, $S(d_1)$ is 1 because no MeSH concept can be found. As to $S(d_2)$, $Depth(“respiratory\ infection'') = 3$, $Depth(“coronavirus'') = 5.5$ since there are two nodes in MeSH tree representing “coronavirus”, one has a depth 5 and another is 6. An average tree depth is calculated in this example. $Depth(“SARS-CoV'') = 6.5$. The total number of concepts in definition 2 is 8. Therefore the value of $S(d_2)$ is 0.1534 which is smaller than $S(d_1)$. This result shows definition 2 has less generality than definition 1.

4.2.3 Computing Document Cohesion

With MeSH hierarchical structure (tree), it is possible to retrieve the semantic distance between MeSH concepts according to their positions in the tree.

We introduce the concept of document cohesion which is a state or quality that the elements of a text (e.g. clauses) “tend to hang together” (Morris & Hirst 1991). The intuition of our approach is based on a hypothesis that document with less cohesion would be more general. Consider two definitions of HIV: the first one comes from a web site called AIDS 101, Guide to HIV basics[5], and the second come from MeSH ontology. Obviously, definition 1 is more general than definition 2.

1. “HIV-1” is the virus most researchers believe causes AIDS.

2. HIV is a non-taxonomic and historical term referring to any of two species, specifically HIV-1 and/or HIV-2.

In definition 2, three MeSH concepts can be identified: “HIV”, “HIV-1” and “HIV-2”. In definition 1, “HIV”, “AIDS” and “virus” are identified as MeSH concepts.

What causes definition 1 to be more general than definition 2? We found that there is stronger cohesion in definition 2 than in definition 1. In other words, concepts in definition 2 are more strongly associated than those in definition 1. “HIV-1” and “HIV-2” are two types of “HIV” in terms of MeSH ontology. However, in definition 1, “HIV” is a kind of virus but “AIDS” is a kind of diseases. There is not a direct relationship between them. Moreover, “HIV” doesn't directly belong to “virus” in MeSH tree.

Following the above observations, it seems that the document generality is somehow related to document cohesion. The higher a document's degree of cohesion, the lower its generality.

We mentioned that the cohesion of a document is regarded as a computation of the associations between the concepts found in the MeSH tree. The more closely the concepts are associated, the more specific the document is. In terms of that, firstly, in our computation of document scope MeSH concepts in document are considered rather than non-ontology concepts. A mean function is used to calculate the average strength of associations between all pairs of MeSH concepts found in document. Secondly, we assume that the strength of association between two MeSH concepts is a monotonic decreasing function of the shortest path between them in the MeSH tree. The minimum value of the function is set to 0 when the shortest path between two MeSH concepts is as large as twice the maximum tree depth. The maximum value of the function is resulted when the shortest path between them equals to 1. In our

[5] http://www.sfaf.org/aids101/

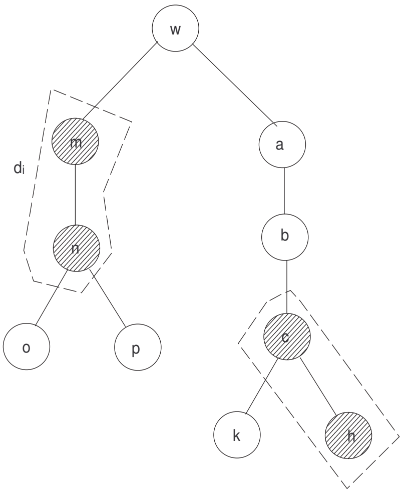


Figure 4: d_i and d_j with different document scope

research, the calculation of semantic association between concepts is based on the Leacock-Chodorow similarity (Leacock & Chodorow 1998) function which is a logistic function featured for measuring the shortest path between two concepts in the MeSH tree.

$$Cohesion(d_i) = \frac{\sum_{i,j=1}^{n} Sim(c_i, c_j)}{NumberofAssociations}, \; (n > 1, \; i < j) \tag{3}$$

$$Sim(c_i, c_j) = -log\frac{len(c_i, c_j)}{2D} \tag{4}$$

$$NumberofAssociations = \frac{n(n-1)}{2} \tag{5}$$

In Equation 3, n is the total number of MeSH concepts in a document d_i. $Sim(c_i, c_j)$ is a function computing the Leacock-Chodorow semantic similarity by using the shortest path $len(c_i, c_j)$ between c_i and c_j in the MeSH tree. $NumberofAssociations$ is the total number of associations among different MeSH concepts, which is defined in Equation 5.

In Equation 4, D is the maximum MeSH tree depth. In our experiments, D is 11. The scope of Equation 3 is $[0, -log(\frac{1}{22})]$. As to a document with zero or one MeSH concept only, its document cohesion is set to 0. For a documents with strongest associations among all the concepts within the document, its cohesion is $-log(\frac{1}{22})$, the maximum value. The time complexity of cohesion-based ranking is $O(m \times n^2)$, m is the number of retrieved documents, n is the average concepts in those documents.

There are two typical examples where the concepts in documents may have different distributions in MeSH tree in terms of their subsumption relationships. Concept A subsuming concept B in the MeSH tree indicates that A is one of the parent nodes of B. The followings are illustrations of our cohesion algorithm in both examples.

Example Three

A document may contain MeSH concepts that have no subsumption relationship between each other in the MeSH tree. In Figure 5, there is a piece of MeSH tree. Every labeled node is a MeSH concept. Suppose that d_i and d_j are two documents in the document collection. d_j is more general than d_i. Each of them contains only two concepts. The concepts o and p in d_i have matches found in the MeSH tree (the darkened nodes). The concepts x and y in d_j have matches found in the MeSH tree too. According to our algorithm, the length of the shortest path between o and p is 2. The shortest distance between x and y is 4. Thus the cohesion of d_i is 2.3979, greater than the generality of d_j, 1.7047.

Example Four

A document may contain MeSH concepts that have subsumption relationship between each other in the MeSH tree. In Figure 6, there is a piece of MeSH tree. Every labeled node is a MeSH concept. Suppose that d_i and d_j are two documents in the document collection. d_j is more general than d_i. Each of them contains only two concepts. The concepts o and n in d_i have matches found in the MeSH tree (the darkened nodes). The concepts i and y in d_j have matches found in the MeSH tree too. According to our algorithm, the length of the shortest path between o and n is 1. The shortest distance between i and y is 2. Thus the cohesion of d_i is 3.0910, greater than the generality of d_j, 2.3979.

4.2.4 Computing Document Generality

The following is the formula for the calculation of document generality.

$$DG(d_i) = \frac{Scope(d_i)}{Cohesion(d_i) + 1} \tag{6}$$

The query generality computation is similar to the computation of document generality. The difference between them is that we take ω, the Statistical Query Generality (SQG), in Equation 1 as an optional parameter for query generality calculation.

$$QG = \frac{SQG * Scope(Q)}{Cohesion(Q) + 1} \tag{7}$$

In Equation 7, QG is the query generality. The calculations of query cohesion and scope is the same as document cohesion and scope.

However, we argue that it is better to give high ranks to those documents whose generality are close to the queries'. For example, it is not suitable to give high ranks to the review or introduction papers on "malignant pericardial effusion" for the query "best treatment of malignant pericardial effusion in esophageal cancer". Thus, we rank the documents by comparing the closeness of documents' generality scores to the query's. In this research the generality closeness between query Q and document d_i is computed as the absolute value of the difference between $DG(d_i)$ and QG.

4.2.5 Correlation Analysis

The independent relationship between generality and similarity is a major hypothesis of this paper. Prov-

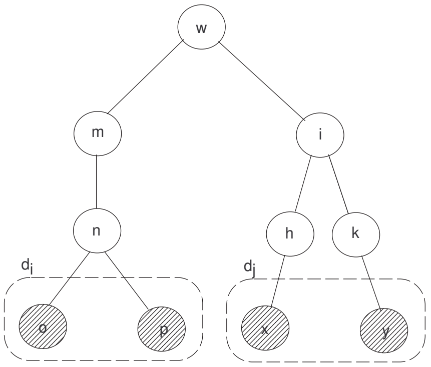


Figure 5: d_i and d_j with different document cohesion

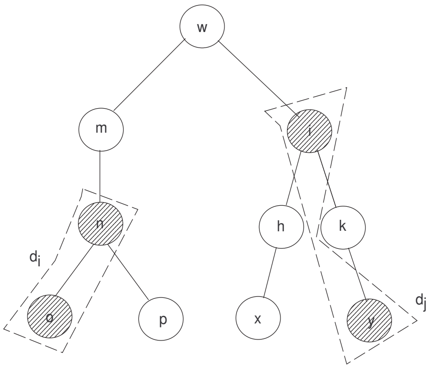


Figure 6: d_i and d_j with different document cohesion

ing this hypothesis is important for the further combination of similarity and generality in the re-ranking process.

Theoretically, the similarity computation itself, does not reflect the generality that exists within the documents. For example, we assume that user can explicitly specify for a given query, if the query is intended to be specific, or general, then the conventional similarity-based methods would not be able to retrieve documents which are specific or general in a given domain knowledge.

Practically, we clarify the relationship between generality and similarity by using intuitive scatter diagrams and Pearson's correlation coefficients. Pearson's correlation coefficient can measure the degree of association between two continuous variables. Scatter diagram can visualize their association. By submitting query to a IR system, the similarity of retrieved documents to query are computed and scored. Our proposed method to calculate document generality is then applied on those retrieved documents in order to get their generality score. The correlation analysis between document generality and its similarity score is then used to prove our hypothesis of the independent relationship between generality and similarity.

4.2.6 Combining Similarity and Generality

As an important step in our proposed approach, we consider both the document similarity and generality. Here information retrieval system is regarded as a black box. Through the query submitted as input, the output of the black box is a ranked list where documents are scored. Let $RScore(d_i)$ denote the similarity score given to a ranked document d_i and QG is the query generality. The final score considering both document similarity and generality is given in the following formula.

$$Score(d_i, Q) = RScore(d_i)^{\alpha} * e^{-|DG(d_i)-QG|^{\beta}} \quad (8)$$

α and β are parameters for a well tuned performance.

5 Experiment and Evaluation

It is necessary to evaluate the effectiveness of our proposed algorithm. The evaluation of effectiveness can be divided into two aspects. The first is on how it can improve the overall performance of a baseline IR system, while the second is on how it can improve the performance of the generality retrieval.

We evaluated the effectiveness of our proposed reranking algorithm on the overall query performance by comparing our algorithm against a baseline IR system.

5.1 Data Set and Queries

Our model has been evaluated on the OHSUMED (Hersh et al. 1994) corpus, which is a subset of Medline and contains 348566 medical references. There are a number of fields in a reference, such as title, abstract, author, source and publication type.

In OHSUMED (Hersh et al. 1994) there are 106 topics and their relevance judgments made by novice physicians. Each topic has two parts: the patient information and the physician's information need. In this research, 106 test queries are formed by combining both parts for each of the 106 topics. In addition, queries 8, 28, 49, 86, and 93 are dropped because there are no relevant documents identified for them. Therefore, a total number of 101 test queries are used in our experiments.

There are queries apparently asking for review information. The following eight review-type queries are selected to test the effect of query generality.

- No.4 reviews on subdurals in elderly
- No.11 review article on cholesterol emboli
- No.17 RH isoimmunization, review topics
- No.31 chronic pain management, review article, use of tricyclic antidepressants
- No.34 review article on adult respiratory syndrome
- No.54 angiotensin converting enzyme inhibitors, review article
- No.105 review of anemia of chronic illness
- No.106 HIV and the GI tract, recent reviews

5.2 Baseline and Pre-processing

Lucene[6] is used as the baseline IR system to index and retrieve the titles and abstracts of documents in OHSUMED collection (Hersh et al. 1994). We chose Lucene as our baseline IR system as it offers a full representative features of a traditional keyword matching IR system. All terms are filtered by the SMART 571 stop word list and stemmed using the Porter stemming algorithm. The MeSH concepts are identified by using our conceptual marking tree algorithm.

5.3 Evaluation Methodology

In our experiments, the baseline IR system is used to retrieve 1000 documents for each test query. We then cover all nine possible cases where query generality, document generality and SQG are used solely or together in a reasonable manner. Those nine cases are derived from our proposed Equation 6, 7 and 8 for re-ranking the documents retrieved by the baseline IR system. For example, DS is the case where only document scope (i.e. Equation 2) is considered in the computation of document generality. The score function in Equation 8 is then simplified as Equation 9 and 10 where α and β are parameters for a well tuned performance.

$$DG(d_i) = Scope(d_i) \quad (9)$$

$$Score(d_i, Q) = RScore(d_i)^{\alpha} * DG(d_i)^{\beta} \quad (10)$$

QS+QC+DS+DC is the case where the closeness between query generality (scope and cohesion) and document generality (scope and cohesion) is considered in the computation of generality re-ranking(i.e. Equation 8). QS and QC denote query scope and query cohesion, DS and DC denote document scope and document cohesion.

5.4 Performance Indicators

The performance of re-ranking is measured in two aspects. Firstly we compare the precision and recall of re-ranking with the original ranking given by baseline IR system[7] for all the 101 test queries. Secondly, we check if all the review type queries get larger improvement in term of average precision.

[6] http://lucene.apache.org/java/docs/index.html

[7] http://lucene.apache.org/java/docs/index.html

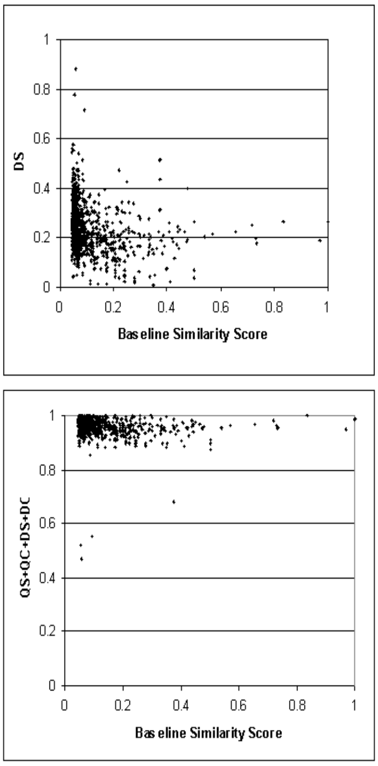


Figure 7: (Query No.5) Up: DS R = -0.24 Down: QS+QC+DS+DC R = -0.17

5.5 Experiment Results

In the upper part of Figure 7, the correlation between DS case and the baseline IR system is shown in a scatter diagram. In the lower part of Figure 7, the correlation between QS+QC+DS+DC case and the baseline IR system is shown in a scatter diagram.

Figure 8 shows the precision-recall graph in a certain range of precision and recall. Due to the limitation of space, Table 1 shows only the detailed precisions of one of the nine cases with the best performance at different recall levels. In Table 2, we show how the review type queries are improved by a comparison of mean average precision between our proposed re-ranking algorithms and the baseline IR system. The mean average precision ("MAP" in the tables) and the percentages of improvement in MAP ("%" in the tables) are summarized.

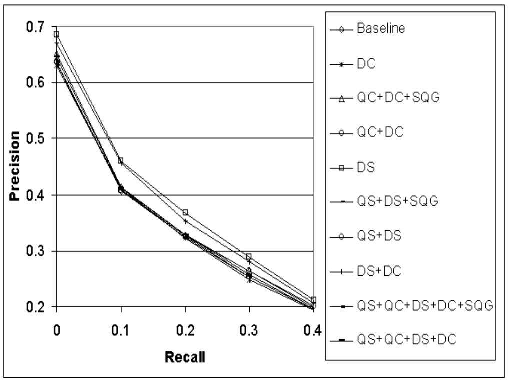


Figure 8: Precision Recall Graph of Overall Query Performance (Recall in [0, 0.4], Precision in [0.2, 0.7])

Table 1: Detailed Precision-Recall Comparisons

Recall	Baseline	DS
0	0.6369	**0.6858**
0.1	0.4071	**0.4591**
0.2	0.3239	**0.3674**
0.3	0.254	**0.2881**
0.4	0.1963	**0.2125**
0.5	0.1679	**0.1770**
0.6	0.1396	**0.1414**
0.7	0.088	**0.0917**
0.8	0.0544	**0.0565**
0.9	0.0223	**0.0236**
1	0.0018	**0.0023**
MAP	0.1849	**0.2036**
%		**10.11%**
R-prec	0.2246	**0.2800**
%		**24.67%**

Table 2: Precision Improvement on Review Type Queries

QNo.	Baseline	QS+QC+DS+DC
4	0.0821	**0.0827**
11	0.0741	**0.0935**
17	0.0021	**0.0023**
31	0.1522	**0.1525**
34	0.0193	0.0190
54	0.1099	**0.1124**
105	0.2950	0.2949
106	0.0085	**0.0087**
MAP	0.0929	**0.0958**
%		**3.07%**

5.6 Results Analysis

In Figure 7, it can be clearly seen that there is no strong relationship (e.g. linear relationship) between generality and similarity in the scatter diagrams. Moreover, the values of correlation coefficient are quite small too. Therefore it shows that generality and similarity are two different concepts without strong correlation between them.

Within all the cases, DS improve the query performance significantly for all 101 queries. There are a 10.11% improvement of MAP and 24.67% improvement of R-prec. This indicates that it is effective to do the re-ranking by considering both document generality and similarity.

The results show the better performance of QS+QC+DS+DC on review type queries. There is an encouraging 3.07% improvement for QS+QC+DS+DC. We performed a dependent t-test (Paired Two Sample for Means) which compares the paired precisions between the baseline and the QS+QC+DS+DC algorithm over different queries in Table 2. With a $p - value$ less than 0.05, it turns out that the improvement is significant. This also verifies our motivation discussed that the technical generality retrieval happens more often for review type queries from non-domain-expert user.

6 Conclusions

In this paper, we argued that there is a need of document generality computation in information retrieval. A novel approach to generality computation has been proposed. Our approach uses the MeSH ontology structure in bio-medical domain to compute the generality based on both statistical and semantic relationships between the terms. Then we applied our proposed generality computation method to the document re-ranking in bio-medical information retrieval. Traditional similarity-based document ranking methods are incorporated with the generality computations. The experiments of our approach have shown that "generality" is an important complement to the traditional similarity-based ranking. The intuition is that when search results are returned by IR system, user may expect to see the documents broadly describing a certain topic to be ranked on the top of the list, so that they can get an overview of the topic first rather than going into the specific ones immediately.

In our proposed framework of document reranking in bio-medical information retrieval, documents are scored and re-ranked by a combination of their similarity to query and the closeness of documents' generality to the query's. Experiments have been conducted on a large corpus namely OHSUMED (Hersh et al. 1994). Our approach shows an improved query performance and encourages us to pursue the further investigation. Our approach can also be applicable to other domains where the domain specific ontology is available.

There are some further works expected. Firstly the cohesion algorithm currently has an oversimplification since it considers semantic relationship between MeSH concepts only. Since there is a large percentage of non-ontology concepts in documents, it is necessary to consider statistical relationships between concepts. A possible solution is to consider the co-occurrence relationship of concepts (i.e. both MeSH and non-ontology concept). The more often two concepts co-occur, the stronger their association is.

Secondly, it is necessary to fully evaluate the effectiveness of our proposed algorithms on generality-based retrieval by comparing our algorithms with other baselines (Zhai et al. 2003), (Liu et al. 2004). More experiments on the evaluation frameworks in related work (Zhai et al. 2003), (Liu et al. 2004) need to be performed for the purpose of tuning the generality computation formulas.

Finally, the domain-independent generality ranking may need to be studied. Currently our proposed algorithms are domain dependent. We re-rank biomedical documents in the context of a given biomedical ontology. The performance of our re-ranking algorithms in a general domain-independent environment is unknown. However, the idea presented in this paper has shown a new way of document ranking and is promising towards the improvement of information retrieval in general.

7 Acknowledgments

The work reported in this paper has been funded in part by the Co-operative Centre for Enterprise Distributed Systems Technology (DSTC) through the Australian Federal Governments CRC Programme (Department of Education, Science and Training).

References

Allen, R. B. & Wu, Y. (2002), Generality of texts, *in* 'Proceedings of the 5th International Conference on Asian Digital Libraries: Digital Libraries: People, Knowledge, and Technology', pp. 111–116.

Carbonell, J. & Goldstain, J. (1998), The use of MMR, diversity-based reranking for reordering documents and producing summaries., *in* 'SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval', ACM Press, New York, NY, USA.

He, B. & Ounis, I. (2004), Inferring query performance using pre-retrieval predictors, *in* '11th Symposium on String Processing and Information Retrieval', Padova, Italy, pp. 43–54.

Hersh, W. (2000), TREC-8 interactive track report, *in* 'The Eighth Text REtrieval Conference', pp. 57–64.

Hersh, W., Buckley, C., Leone, T. J. & Hickam, D. (1994), OHSUMED: an interactive retrieval evaluation and new large test collection for research, *in* 'Annual ACM Conference on Research and Development in Information Retrieval', pp. 192 – 201.

Leacock, C. & Chodorow, M. (1998), Combining local context and wordnet similarity for word sense identification, *in* 'Fellbaum', pp. 265–283.

Liu, Y., Zhang, B., Chen, Z., Lyu, M. R. & Ma, W.-Y. (2004), Affinity rank: A new scheme for efficient web search, *in* 'The Thirteenth World Wide Web conference', Vol. 203-211, ACM, New York, USA.

Morris, J. & Hirst, G. (1991), 'Lexical cohesion computed by thesaural relations as an indicator of the structure of text', *Computational Linguistics* **17**(1), 21–48.

Over, P. (1999), TREC-7 interactive track report, *in* 'The Seventh Text REtrieval Conference', pp. 65–72.

Plachouras, V., Cacheda, F., Ounis, I. & Rijsbergen, C. v. (2003), University of glasgow at the web track: Dynamic application of hyperlink analysis using the query scope., *in* 'In Proceedings of the 12th Text Retrieval Conference TREC 2003', Gaithersburg.

Robertson, S. E., Walker, S. & Beaulieu, M. (1999), Okapi at TREC-7: automatic ad hoc, filtering, vlc and interactive track., *in* E. M. Voorhees & D. K. Harman, eds, 'The Seventh Text REtrieval Conference (TREC-7)', Gaithersburg, MD, USA.

Salton, G., Wong, A. & Yang, C. S. (1975), 'A vector space model for automatic indexing', *Communications of the ACM* **18**(11).

Swan, R. C. & Allan, J. (1998), Aspect windows, 3-D visualizations, and indirect comparisons of information retrieval systems, *in* 'SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval', ACM Press, New York, NY, USA, pp. 173–181.

Van Rijsbergen, C. J. (1979), *Information Retrieval*, London; Boston: Butterworths.

Zhai, C., Cohen, W. W. & Lafferty, J. (2003), Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval, *in* 'Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval', pp. 10–17.

Using Reflection for Querying XML Documents*

Markus Kirchberg, Faizal Riaz-ud-Din, Klaus-Dieter Schewe, Alexei Tretiakov

Massey University, Department of Information Systems &
Information Science Research Centre
Private Bag 11 222, Palmerston North, New Zealand
Email: [m.kirchberg|f.din|k.d.schewe|a.tretiakov]@massey.ac.nz

Abstract

XML-based databases have become a major area of interest in database research. Abstractly speaking they can be considered as a resurrection of complex-value databases using constructors for records, lists, unions plus optionality and references. XQuery has become the standard query language for XML. In this paper an implementation of XQuery based on linguistic reflection is proposed. That is, XQuery is translated into a query algebra for rational tree types based on simple operations and structural recursion for lists. The major purpose of using reflection is to expand path expressions in a type-safe way.

1 Introduction

As emphasised in (Lobin 2001) the original purpose of XML – same as SGML – was to support the description of content rather than layout of text documents. For instance, in the Orlando project (Ruecker 2000) XML is used to markup the content of novels of female British and Irish writers. Nevertheless, XML has become a major area of interest in database research. If some aspects of XML that make sense for text markup but not so much for databases are neglected, XML can be considered as a complex-value data model using constructors for records, lists, unions plus optionality and references (Abiteboul, Buneman & Suciu 2000, Siméon & Wadler 2003).

Using XML for databases requires schema definition, query and update languages. By now, XML Schema (World Wide Web Consortium (W3C) 2001) has become the W3C standard for defining schemata, while XQuery (Katz 2003, World Wide Web Consortium (W3C) 2004) is the recommended standard for querying XML documents. For updates only little work has been done so far, e.g. (Tatarinov, Ives, Halevy & Weld 2001).

In fact, XML Schema supports almost directly the definition of tree types using the mentioned constructors. XQuery combines ideas from various predecessor proposals for XML query languages (Abiteboul, Quass, McHugh, Widom & Wiener 1997, Buneman, Davidson, Hillebrand & Suciu 1996, Deutsch, Fernandez, Florescu, Levy & Suciu 1999). Most importantly, queries are composed of a matching part that binds variables to values according to a given XML document, and a construction part that creates new XML documents from these variables.

When it comes to implementing XML databases and in particular XQuery, there are two major lines of research. The first one, e.g. (DeHaan, Toman, Consens & Özsu 2003), attempts a translation to SQL based on a reification of XML via relational database technology. The drawback of this approach is that semantics may be lost in the translation from trees to relations. The alternative is to approach a direct implementation of XQuery, e.g. (Paparizos, Wu, Lakshmanan & Jagadish 2004). Our own research follows this second line.

The major difficulty in implementing XQuery results from path expressions, i.e. from the fragment of the language that subsumes XPath (Chen, Davidson & Zheng 2004). These difficulties are supported by the theoretical analysis of the complexity of XPath (Gottlob, Koch & Pichler 2003, Marx 2004).

Our work reported in this paper follows the idea of translating XQuery to a (general purpose) query algebra for rational trees as defined in (Schewe 2001). For short let RTA denote this query algebra. RTA uses operators for the type system, i.e. for the abstract system of types as defined in (Katz 2003). In particular, it exploits structural recursion (Tannen, Buneman & Wong 1992, Wadler 1992) for lists. Koch in (Koch 2005) has used a similar approach based on list comprehensions. However, his work is placed in a complexity-theoretic setting. The advantages of such a translation to a query algebra are the support of algebraic query optimisation, the easy implementation of the operations and the integration with programming languages, e.g. using the physical architecture from (Kirchberg, Schewe & Tretiakov 2003), and the easy extension to other constructors such as sets and multisets in case the order that comes with the list constructor is considered unnecessary or even undesired.

However, the translation of XQuery to RTA requires schema information, in particular for the path expressions. Thus, it is a natural idea to exploit type-safe linguistic reflection (Stemple, Fegaras, Sheard & Socorro 1990) to deal with this problem. This is what we do in this paper, i.e. we present a translation from essential parts of XQuery to RTA and show how this translation benefits from linguistic reflection.

Therefore, we first introduce an abstract model of XML, XQuery and RTA in Sections 2 and 3, respectively. In Section 4 we then outline the translation from XQuery to RTA. We focus on structural recursion and show that some of the functions used as parameters have a “complicated nature”, as they refer to path finding. For this we introduce linguistic reflection and show how it can be used to expand these functions of “complicated nature”. We conclude with a brief summary.

* The work reported in this paper was supported by FRST/NERF grant MAUX0025 “DIMO – Distributed Multi-Level Object Bases”.

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology, Vol. 49. Gill Dobbie and James Bailey, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

2 Abstract Model of XML and XQuery

In this section we describe some basics of XML and XQuery. Of course, as both of these are complex languages, we cannot describe all details and therefore take a more abstract view focusing more on the semantics than on the syntax.

2.1 XML Documents as Trees

Start with a type system that supports records, lists and unions. Using abstract syntax this type system can be described by

$$t = b \mid (t_1, \dots, t_n) \mid [t] \mid t_1 \oplus \cdots \oplus t_n.$$

Here b represents a (not further specified) collection of base types, e.g. the base types supported by XML such as *String*, *Integer*, *Double*, *ID*, etc. For reasons that will become clear, when we add references, we only use a single type *ID* for identifiers. Furthermore, assume that one of the base types is *Empty* with only one possible value. This type can be used to support optionality.

We use $(t_1, \dots, t_n)$ to denote an ordered record type with component types t_i, the type $[t]$ is used for finite lists, and $t_1 \oplus \cdots \oplus t_n$ is used for a (disjoint) union type with components t_i.

Each type t denotes a set of values called its *domain* $dom(t)$. Formally, we obtain these domains as follows:

- $dom(b_i) = V_i$, i.e. for each base type b_i we assume some set V_i of values of that type, e.g. $dom(EMPTY) = \{\perp\}$.
- $dom((t_1, \dots, t_n)) = dom(t_1) \times \cdots \times dom(t_n)$.
- $dom([t]) = \{[v_1, \dots, v_k] \mid k \in \mathbb{N}, v_i \in dom(t)\}$.
- $dom(t_1 \oplus \cdots \oplus t_n) = \{(i, v_i) \mid 1 \leq i \leq n, v_i \in dom(t_i)\}$.

Then an XML document can be represented by a value of some type t, which in turn is representable as a tree, provided the document does not contain references. In particular, we can treat attributes in the same way as subelements – which is no loss of generality for databases, whereas for text markup it may make a significant difference.

In order to also capture references, we extend the type system to

$$t = b \mid \ell \mid (t_1, \dots, t_n) \mid [t] \mid t_1 \oplus \cdots \oplus t_n \mid \ell : t,$$

where ℓ represents reference labels. The domains are simply $dom(\ell) = dom(ID)$ and $dom(\ell : t) = \{(i, v) \mid i \in dom(\ell), v \in dom(t)\}$. Following (Abiteboul et al. 2000) each occurrence of a value i of type *ID* in some complex value v that corresponds to a labelled type $\ell : t$ *defines* a reference, whereas each occurrence of a value i of type *ID* in v that corresponds to a label ℓ *uses* the reference. In XML Schema the usage of references corresponds to the type *IDREF*, whereas the definition of references corresponds to the type *ID*. Furthermore, *IDREFS* corresponds to a list type $[\ell]$ – in fact, here we would prefer to use a set type, but for simplicity and orthogonality of the constructors let us use only one bulk type constructor.

Example 2.1 Let us look at the following schema definition in XML Schema:

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="cellar">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="wines"/>
        <xs:element ref="wineries"/>
        <xs:element ref="regions"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="wines">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="wine"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="wineries">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="winery"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="regions">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="region"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="wine">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="year"
          type="xs:integer"
          minOccurs="0"/>
        <xs:element ref="blend"
          maxOccurs="unbounded"/>
        <xs:element name="price"
          type="xs:decimal"/>
      </xs:sequence>
      <xs:attribute name="w-id"
        type="xs:ID" use="required"/>
      <xs:attribute name="producer"
        type="xs:IDREF" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="blend">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="grape"
          type="xs:string"/>
        <xs:element name="percentage"
          type="xs:integer"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="winery">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="owner"
          type="xs:string" maxOccurs="unbounded"/>
        <xs:element name="area"
          type="xs:string"
          minOccurs="0"/>
        <xs:element name="established"
          type="xs:date" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="v-id"
        type="xs:ID" use="required"/>
      <xs:attribute name="in-region"
        type="xs:IDREF" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="region">
    <xs:complexType>
```

```
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
      </xs:sequence>
      <xs:attribute name="r-id"
        type="xs:ID" use="required"/>
      <xs:attribute name="famous-wines"
        type="xs:IDREFS" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

That is, a cellar contains a list of wines, wineries and regions. A wine is described by a name, a year (optional) and a blend, which is a sequence of grapes together with their percentages. A winery is described by a name, a list of owners, an area (optional) and an establishment date (optional). A region just has a name. Furthermore, there are references from a wine to the winery that produces it, from a winery to the region it is located in, and from a region to all its famous wines.

Using our type system, we obtain the following complex type definitions for representing this schema:

cellar = (wines, wineries, regions)
wines = [w-id : wine]
wineries = [v-id : winery]
regions = [r-id : region]
wine = (w-name, year $\oplus$ *Empty*, [blend], price, producer)
w-name = *String*
year = *Integer*
price = *Decimal*
producer = v-id
blend = (grape, percentage)
grape = *String*
percentage = *Integer*
winery = (v-name, [owner], area $\oplus$ *Empty*,
 established $\oplus$ *Empty*, in-region)
v-name = *String*
owner = *String*
area = *String*
established = *Date*
in-region = r-id
region = (r-name, famous-wines)
r-name = *String*
famous-wines = [w-id]

Here w-id, v-id and r-id are labels.

Example 2.2 Consider the following XML document that is in accordance with the schema defined in Example 2.1:

```
<cellar
  xmlns:xsi=
    "http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="cellar.xsd">
  <wines>
    <wine w-id="o11" producer="o1">
      <name>Marlborough Riesling</name>
      <year>2003</year>
      <blend>
        <grape>Riesling</grape>
        <percentage>100</percentage>
      </blend>
      <price>13.95</price>
    </wine>
    <wine w-id="o12" producer="o1">
      <name>Marlborough Gewurztraminer</name>
      <year>2000</year>
      <blend>
        <grape>Gewurztraminer</grape>
        <percentage>100</percentage>
      </blend>
      <price>17.95</price>
    </wine>
    <wine w-id="o13" producer="o1">
      <name>Everyday's Favourite</name>
      <blend>
        <grape>Sauvignon Blanc</grape>
        <percentage>65</percentage>
      </blend>
      <blend>
        <grape>Semillon</grape>
        <percentage>35</percentage>
      </blend>
      <price>5.95</price>
    </wine>
  </wines>
  <wineries>
    <winery v-id="o1" in-region="o2">
      <name>Marlborough Winery</name>
      <owner>Jacques Vine</owner>
      <owner>Claudine Vine</owner>
      <area>231 ha</area>
      <established>1987-01-01</established>
    </winery>
  </wineries>
  <regions>
    <region r-id="o2" famous-wines="o11 o12">
      <name>Marlborough</name>
    </region>
  </regions>
</cellar>
```

This XML document can be represented by the following complex value:

([(&o_{11},(Marlborough Riesling, 2003,
 [(Riesling, 100)], 13.95, &o_1)),
 (&o_{12},(Marlborough Gewurztraminer, 2000,
 [(Gewurztraminer, 100)], 17.95, &o_1)),
 (&o_{13},(Everyday's Favourite, $\perp$,
 [(Sauvignon Blanc, 65), (Semillon, 35)],
 5.95, &o_1))],
 [(&o_1,(Marlborough Winery, [Jacques Vine,
 Claudine Vine], 231 ha, 1987, &o_2))],
 [(&o_2,(Marlborough, [&o_{11}, &o_{12}]))])

2.2 XQuery in a Nutshell

XQuery is a query language allowing to extract sequences of subtrees and base type values from any number of XML document trees, and to combine them to construct a sequence of trees and basic values (the so-called **items**) forming the result of the query. In practice, most often the result of the query is a sequence consisting of a single tree.

In XQuery, the XML documents serving as input are identified by using the so-called input functions, of which the most commonly used one is doc, which accepts a URL corresponding to the location of an XML document as a parameter. For example, `doc("cellar.xml")` would retrieve the `cellar.xml` document from the current directory.

Sequences of subtrees are retrieved by using the so-called path expressions, consisting of one or more **steps** separated by a slash, /, or double slash, //. Each step acts on the sequence of items created by the previous step to form a further sequence, which either forms the output of the path expression (if the step is the last one), or serves as input for further steps. The following query, formed by combining an input function with a path expression, will result in a sequence of `name` elements representing wine names (assuming `cellar.xml` is the XML document introduced in the previous section): `doc("cellar.xml")/cellar/wines/wine/name`.

`doc("cellar.xml")/cellar` results in a sequence consisting of a single `cellar` element, `doc("cellar.xml")/cellar/wines` results in a sequence consisting of a single `wines` element (a subelement of the `cellar` element obtained in the previous step), `doc("cellar.xml")/cellar/wines/wine` will result in a sequence of all `wine` elements from `wines` and so on. Filtering can be applied to restrict which of the items are to be included in a given step. While / retrieves child items (branches immediately connected to the root), // forms a sequence consisting of all matching subtrees, at all depths. / and // are illustrated in examples 2.3 and 2.4, respectively.

As XQuery is a functional language, an XQuery program can be regarded as an expression formed by subexpressions which, at execution time, are evaluated in the order of precedence. The most commonly used type of expressions in XQuery are the so-called FLWOR (`for`, `let`, `where`, `order by`, `return`) expressions. In a FLWOR expression, a `for` clause binds each item of a sequence to a variable, and evaluates the rest of the expression with that binding, resulting in as many evaluations as there are items in the sequence. The `for` clause is illustrated in Example 2.4 below.

A `let` clause binds the whole sequence to a variable, and evaluates the rest of the expression just once, with that binding. The `let` clause is illustrated in Examples 2.5, 2.6, and 2.7.

The `where` clause serves as a filter: the rest of the FLWOR expression is executed only if the boolean expression associated with the `where` clause evaluates to true. This is illustrated in Example 2.5, 2.6, and 2.7.

The `order by` clause is used for sorting (we do not discuss it here any further).

Finally, the `return` clause is a constructor, instantiating an item that is to be included as the result of the query. By using `return`, items retrieved from different parts of the same document, or from different documents, can be combined together, resulting in sophisticated joins. As shown in Example 2.7, the constructor formed by using the `return` clause can include subqueries, whose output is incorporated into the sequence created by the constructor.

XML Query can make use of type information from XML Schema documents associated with XML documents inputted by the query by explicitly specifying the type of items to be included in sequences or to be constructed. In addition, parsers are able to analyse and to reject a query based on schema information only, if the query is found to construct items that do not match the declared types for constructor output.

Example 2.3 Assume the document in Example 2.2 is stored in cellar.xml. Then

```
<wines>
  {
    doc("cellar.xml")/cellar/wines/wine/name
  }
</wines>
```

is a simple query that will select the names of wines. For our example document the result would be

```
<wines>
   <name>Marlborough Riesling</name>
   <name>Marlborough Gewurztraminer</name>
   <name>Everyday's Favourite</name>
</wines>
```

Example 2.4 The query

```
<wine-makers>
  {
    for $N in doc("cellar.xml")//owner
    return
      <name>{ $N/text() }</name>
  }
</wine-makers>
```

returns the names of winery owners.

Example 2.5 The following is a query with a more interesting `where`-clause, which returns the names of Riesling wines:

```
<Rieslings>
  {
    for $W in doc("cellar.xml")/cellar/wines/wine
    let $N := $W/name, $B := $W/blend
    where $B/grape/text() = "Riesling"
      and $B/percentage/text() = 100
    return <name>{ $N/text() }</name>
  }
</Rieslings>
```

Example 2.6 The following query, which contains selection conditions on the paths, will produce a list of wines with their producers:

```
<wines>
  {
    let $db := doc("cellar.xml")
    for $W in $db//wine, $V in $db//winery
    let $P := $W/@producer, $N := $W/name,
      $M := $V/name, $I := $V/@v-id
    where $I = $P
    return
      <wine>
        <product>{ $N/text() }</product>
        <producer>{ $M/text() }</producer>
      </wine>
  }
</wines>
```

Example 2.7 The following is an example of a nested query:

```
<wines>
  {
    let $db := doc("cellar.xml")
    for $N in $db//wine/name
    return
      <wine>
        { $N }
        {
          for $W in $db//wine
          where $W/name = $N
          return $W/year
        }
      </wine>
  }
</wines>
```

The query lists the names of all wines, adding to each name the corresponding production year, when such information is available.

3 RTA: A Rational Tree Query Algebra

Following a basic idea in (Schewe 2001) we use a query algebra with operations "induced" from the type system plus a join-operation. For our purposes here it is more convenient to consider products instead of joins.

In doing so, let $\mathbb{1}$ denote a trivial type with only one value in its domain. We use a unique "forget"-operation $\mathtt{triv} : t \rightarrow \mathbb{1}$ for each type t. Assume further a boolean type $BOOL$ with constant values $\mathbf{T}$ and $\mathbf{F}$. Thus, we may consider the operations $\wedge : BOOL \times BOOL \rightarrow BOOL$ (conjunction), $\neg : BOOL \rightarrow BOOL$ (negation) and $\Rightarrow: BOOL \times BOOL \rightarrow BOOL$ (implication).

For record types we consider *projection* $\pi_i : (t_1, \dots, t_n) \rightarrow t_i$ and *product* $o_1 \times \cdots \times o_n : t \rightarrow (t_1, \dots, t_n)$ for given operations $o_i : t \rightarrow t_i$. As usual, we write $\pi_{i_1,\dots,i_k}$ as a shortcut for $\pi_{i_1} \times \cdots \times \pi_{i_k}$.

For union types we use the canonical embeddings $\iota_i : t_i \rightarrow t_1 \oplus \cdots \oplus t_n$. Other operations on union types take the form $o_1 + \cdots + o_n : t_1 \oplus \cdots \oplus t_n \rightarrow t$ for given operations $o_i : t_i \rightarrow t$.

For list types we may consider $\frown$ (concatenation), the constant $\texttt{empty} : \mathbb{1} \to [t]$ and the *singleton* operation $\texttt{single} : t \to [t]$ with well known semantics.

It should be noted here that document order is preserved through the use of lists. The ordering of the elements in the lists conforms to the ordering of the elements in the queried xml document (or conforms to specific re-ordering in the query itself) throughout the execution process.

3.1 Structural Recursion

In addition, we consider structural recursion $\texttt{src}[e, g, \sqcup] : [t] \to t'$ with a value e of type t', an operation $g : t \to t'$ and an operation $\sqcup : (t', t') \to t'$, which is defined as follows:

$$\begin{aligned}
\texttt{src}[e, g, \sqcup]([]) &= e \\
\texttt{src}[e, g, \sqcup]([x]) &= g(x) \\
\texttt{src}[e, g, \sqcup](X \frown Y) &= \texttt{src}[e, g, \sqcup](X) \sqcup \texttt{src}[e, g, \sqcup](Y)
\end{aligned}$$

Let us illustrate structural recursion by some more or less standard examples. First consider an operation $f : t \to t'$. We want to raise f to an operation $\texttt{map}(f) : [t] \to [t']$ by applying f to each element of a list. Obviously, we have $\texttt{map}(f) = \texttt{src}[[], \texttt{single} \circ f, \frown]$.

Next consider an operation $\varphi : t \to \mathit{BOOL}$, i.e. a predicate. We want to define an operation $\texttt{filter}(\varphi) : [t] \to [t]$, which maps a given list to the sublist of all elements satisfying the predicate φ. For this we may write $\texttt{filter}(\varphi) =$

$$\texttt{src}[[], \texttt{if\_else} \circ (\varphi \times \texttt{single} \times (\texttt{empty} \circ \texttt{triv})), \frown]$$

with the operation $\texttt{if\_else} : (\mathit{BOOL}, t, t) \to t$ with $(\mathbf{T}, x, y) \mapsto x$ and $(\mathbf{F}, x, y) \mapsto y$.

As a third example assume that t is a type, on which addition $+ : (t, t) \to t$ is defined. Then $\texttt{src}[0, \mathit{id}, +]$ with the identity id on t defines the sum of the elements in a list.

3.2 Querying XML with RTA

Let us simply illustrate now how RTA can be applied to query XML. We will use the queries from the previous section and write equivalent queries in RTA.

Example 3.1 Let us consider first the query in Example 2.3. In this case we basically have to analyse a path expression. For this assume that v_{in} is the complex value in Example 2.2, i.e. it represents the corresponding XML document cellar.xml.

The construct doc(cellar.xml)/cellar creates a list with the whole document as its only entry, which corresponds to applying the RTA-operation $\texttt{single}$ to v_{in}. Then /wines selects the first successor of the root. As v_{in} is a triple, this corresponds to applying $\texttt{map}(\pi_1)$ to $[v_{\text{in}}]$. This gives

$$\begin{aligned}
\texttt{map}(\pi_1)([v_{\text{in}}]) &= \texttt{src}[[], \texttt{single} \circ \pi_1, \frown]([v_{\text{in}}]) \\
&= \texttt{single} \circ \pi_1(v_{\text{in}}) \\
&= [\pi_1(v_{\text{in}})]
\end{aligned}$$

The effect of /wine in the XQuery path expression is to produce only the list of wines, i.e. $\pi_1(v_{\text{in}})$. This can be achieved by another application of structural recursion, in this case $\texttt{src}[[], \texttt{id}, \frown]$. This gives

$$\texttt{src}[[], \texttt{id}, \frown]([\pi_1(v_{\text{in}})]) = \pi_1(v_{\text{in}})$$

as desired. Finally, the effect of /name in the XQuery path expression is first to throw away the identifiers for wines, which can be achieved by applying π_2, then taking the first component, i.e. to apply π_1. Thus, the last step is the application of $\texttt{map}(\pi_1 \circ \pi_2)$.

In summary, the query in Example 2.3 corresponds to the query

$$\texttt{map}(\pi_1 \circ \pi_2) \circ \texttt{src}[[], \texttt{id}, \frown] \circ \texttt{map}(\pi_1) \circ \texttt{single}.$$

Applied to v_{in} we obtain the list value [Marlborough Riesling, Marlborough Gewurztraminer, Everyday's Favourite] as desired.

Example 3.1 already gives valuable hints, how a translation of XQuery into RTA might work. Basically, we follow the execution model for XQuery, which works on lists and applies operations to the elements of the list. So, basically each step corresponds to some structural recursion operation.

Example 3.1 also indicates the expected advantage from the translation into RTA, as we were able to simplify the algebraic query. This is a first step towards query "optimisation".

However, in Example 3.1 we used only explicit path expressions. The next example handles a query, in which we have to search for the path. We will see that this constitutes a much more complicated application of structural recursion.

Example 3.2 Let us now consider the query in Example 2.4. As in the previous example we first have to apply $\texttt{single}$ to v_{in} to achieve the same effect as doc(cellar.xml) in the XQuery path expression. However, the follow-on RTA-operation has to capture the effect of //owner, which can be done by structural recursion. That is, we apply $\texttt{src}[[], h, \frown]$ to $[v_{\text{in}}]$ with an operation h that searches for successors with the name owner.

The application of this operation h to some x can be defined recursively as follows:

```
if type($x$) = owner
then $[x]$
else
  if type($x$) = $(t_1, \dots, t_n)$
  then $h(\pi_1(x)) \frown \dots \frown h(\pi_n(x))$
  else
    if type($x$) = $[t]$
    then $\texttt{src}[[], h, \frown](x)$
    else
      if type($x$) = $t_1 \oplus \cdots \oplus t_n$
      then $h(\pi_2(x))$
      else $[]$
      endif
    endif
  endif
endif
```

Finally, we can neglect the return-clause, as it is just a renaming of tags, which do not appear in our anonymised complex values.

In summary, the corresponding RTA-query is $\texttt{src}[[], h, \frown] \circ \texttt{single}$ with

$$\begin{aligned}
h &= \texttt{if\_then} \circ (\varphi_1 \times \texttt{single} \times h_1) \\
h_1 &= \texttt{if\_then} \circ (\varphi_2 \times (\frown \circ (h \circ \pi_1 \times \frown \circ (\dots \\
&\qquad\qquad (h \circ \pi_{n-1} \times h \circ \pi_n) \dots))) \times h_2) \\
h_2 &= \texttt{if\_then} \circ (\varphi_3 \times \texttt{src}[[], h, \frown] \circ h_3) \\
h_3 &= \texttt{if\_then} \circ (\varphi_4 \times h \circ \pi_2 \times \texttt{empty})
\end{aligned}$$

and the obvious Boolean operations

$$\begin{aligned}\varphi_1(x) &\equiv \text{type}(x) = \text{owner}\\ \varphi_2(x) &\equiv \text{type}(x) = (t_1, \dots, t_n)\\ \varphi_3(x) &\equiv \text{type}(x) = [t]\\ \varphi_4(x) &\equiv \text{type}(x) = t_1 \oplus \dots \oplus t_n\end{aligned}$$

Example 3.2 shows that some of the operations used within RTA-queries require complex definitions. It is not difficult to see that the other examples of queries from the previous section require analogous techniques. We will present a general solution for the translation in Section 4.

3.3 Multi-Dimensional Extension

Let us finally mention a "multi-dimensional" extension of structural recursion, but let us restrict for simplicity to the binary case. That is, we define an operation $src2[f, g, h] : ([t_1], [t_2]) \rightarrow t$ with parameters $f : [t_2] \rightarrow t$, $g : (t_1, [t_2]) \rightarrow t$, and $h : (t, t) \rightarrow t$. Similarly to the "one-dimensional" case we define

$$\begin{aligned}\texttt{src2}[f, g, h]([], L_2) &= f(L_2)\\ \texttt{src2}[f, g, h]([x], L_2) &= g(x, L_2)\\ \texttt{src2}[f, g, h](X^\frown Y, L_2) &= h(\texttt{src2}[f, g, h](X, L_2),\\ &\qquad \texttt{src2}[f, g, h](Y, L_2))\end{aligned}$$

This can be used to define the product of lists (both of type $[t]$) as

$$L_1 \times L_2 = \texttt{src2}[[], g, ^\frown](L_1, L_2),$$

where [] is treated as a constant function, and g is defined by

$$g(x, L_2) = \texttt{src}[[], \texttt{single} \circ (x \times \texttt{id}), ^\frown](L_2).$$

4 Linguistic Reflection in Translating XQuery to RTA

Our goal is to translate XQuery into RTA. For this recall that XQuery is basically a functional language, so each query corresponds to a sequence of function calls. For instance, for the simple query in Example 2.3 we would first evaluate ⟨wines⟩ by simply printing it, then evaluate the expression { doc(cellar.xml)/... }, finally evaluate ⟨/wines⟩, which again amounts only to a simple print. Therefore, we concentrate on expressions of the form { ... } with the dots standing for a FLWOR-expression.

4.1 The Basic Translation Model

XQuery works on lists, and each part of a query corresponds to some function that is executed on all elements of the list. As we assume to be given a FLWOR-expression, we first look at the `for`-construct. In its simple form it has the form

`for` $X `in` ⟨path-expression⟩,

so we have to evaluate the path expression first:

- If doc(xxx.xml) appears in the path expression, then xxx.xml is some input document, which is represented by some complex value, say v_{in}. As we have already seen in Examples 3.1 and 3.2, the input-function doc simply corresponds to the RTA-operation `single`.
- If p/name appears in the path expression, we first translate p, say the result is trans(p). Then /name gives rise to an application of structural recursion, say $\texttt{src}[[], g_{/\text{name}}, ^\frown]$. Thus, the translation of p/name is

  $$\text{trans}(p/\text{name}) = \texttt{src}[[], g_{/\text{name}}, ^\frown](\text{trans}(p)).$$

  The difficult part is then to determine the operation $g_{/\text{name}}$. Note that all applications of structural recursion in Example 3.1 refer to this step.
- If p//name appears in the path expression, we proceed analogously. That is, //name gives rise to an application of structural recursion $\texttt{src}[[], g_{//\text{name}}, ^\frown]$, and the translation of p//name is

  $$\text{trans}(p//\text{name}) = \texttt{src}[[], g_{//\text{name}}, ^\frown](\text{trans}(p)).$$

  Note that the structural recursion in Example 3.2 refers to this step. It also indicates how to define $g = g_{//\text{name}}$ in general:

  $$\begin{aligned}g &= \texttt{if\_then} \circ (\varphi_1 \times \texttt{single} \times h_1)\\ h_1 &= \texttt{if\_then} \circ (\varphi_2 \times (^\frown \circ (g \circ \pi_1 \times {}^\frown \circ (\dots\\ &\qquad (g \circ \pi_{n-1} \times g \circ \pi_n) \dots))) \times h_2)\\ h_2 &= \texttt{if\_then} \circ (\varphi_3 \times \texttt{src}[[], g, ^\frown] \circ h_3)\\ h_3 &= \texttt{if\_then} \circ (\varphi_4 \times g \circ \pi_2 \times \texttt{empty})\end{aligned}$$

  with the Boolean operations

  $$\begin{aligned}\varphi_1(x) &\equiv \text{type}(x) = \text{name}\\ \varphi_2(x) &\equiv \text{type}(x) = (t_1, \dots, t_n)\\ \varphi_3(x) &\equiv \text{type}(x) = [t]\\ \varphi_4(x) &\equiv \text{type}(x) = t_1 \oplus \dots \oplus t_n\end{aligned}$$

  The crucial remaining part is to take care of the Boolean operations, as these require type-checks.
- If p[test] appears in the path expression, we first translate p. Furthermore, test will be translated into a Boolean condition ψ, and we can combine both using structural recursion, in this case a `filter`-operation, i.e.

  $$\text{trans}(p[\text{test}]) = \texttt{filter}(\psi)(\text{trans}(p)).$$

If there is more than one condition in the `for`-clause, say

`for` $X_1 `in` ⟨path-expression$_1$⟩, ...
$X_n `in` ⟨path-expression$_n$⟩,

each path expression will be translated separately resulting in RTA-operations $o_1, \dots, o_n$, each producing some list, say L_i. Then we have to combine these lists into one list containing all tuple combinations, i.e. we obtain $L_1 \times \dots \times L_n$.

The following `let`-clause simply binds further variables depending on the list resulting from evaluating the `for`-clause. As this may again require evaluating a path expression, we proceed analogously to translating the `for`-clause.

Example 4.1 Look at the query in Example 2.5. Analogous to Example 3.1 the `for`-clause will be translated to the operation

$$\texttt{src}[[], \texttt{id}, ^\frown] \circ \texttt{map}(\pi_1) \circ \texttt{single},$$

which will be applied to v_{in}. Now the first part of the `let`-clause corresponds to $\texttt{map}(\pi_1 \circ \pi_2)$ as already seen in Example 3.1. Similarly, the second part of the `let`-clause corresponds to $\texttt{map}(\texttt{first} \circ \pi_3 \circ \pi_2)$ with an operation `first` that selects the first element of a list.

However, we do not want to replace the $\$W$-values by the $\$N$-values or the $\$B$-values, but keep all three, so the `let`-clause defines the operation

$$\texttt{map}((\texttt{id} \times \pi_1 \times (\texttt{first} \circ \pi_3)) \circ \pi_2).$$

The remaining clauses in FLWOR expressions are easy to handle. A `where`-clause defines a `filter`-operation. The greatest difficulty is to determine the Boolean operation, which may again involve the translation of a path expression. An `order`-clause corresponds to applying a sorting-operation, which can be expressed by structural recursion. Finally, the `return`-clause only constructs a value, so the only difficulty that may occur is that this construction involves evaluating another query.

Example 4.2 Let us continue our previous example, as Example 2.5 contains a `where`-clause. The list resulting from the application of the operation in Example 4.1, which represents the combination of the `for`- and `let`-clause, contains triples, where the first component corresponds to a wine, the second one to its name and the third one to the first-listed component of its blend. Thus, applying $\pi_1 \circ \pi_3$ to such a triple gives the requested name of the first grape, while the application of $\pi_2 \circ \pi_3$ results in the corresponding percentage.

Thus, the first condition in the `where`-clause corresponds to the Boolean operation $\texttt{eq} \circ ((\pi_1 \circ \pi_3) \times \text{Riesling})$, in which Riesling is treated as a constant operation. Similarly, the second condition gives the Boolean operation $\texttt{eq} \circ ((\pi_2 \circ \pi_3) \times 100)$, and thus, the whole `where`-clause corresponds to the RTA-operation $\texttt{filter}(\varphi)$, where φ is defined by the Boolean operation

$$\wedge \circ ((\texttt{eq} \circ ((\pi_1 \circ \pi_3) \times \text{Riesling})) \times (\texttt{eq} \circ ((\pi_2 \circ \pi_3) \times 100))).$$

Finally, let us complete the translation of the query in Example 2.5. Taking all the parts together, we obtain the RTA-operation

$$\begin{gathered} \texttt{map}(\pi_2) \circ \texttt{filter}(\varphi) \circ \\ \texttt{map}((\texttt{id} \times \pi_1 \times (\texttt{first} \circ \pi_3)) \circ \pi_2) \circ \\ \texttt{src}[[], \texttt{id}, \frown] \circ \texttt{map}(\pi_1) \circ \texttt{single} \end{gathered}$$

4.2 Type-Safe Linguistic Reflection

In the previous subsection we have seen that the translation of XQuery can be done by parsing through FLWOR expressions and translating them step-by-step into RTA-operations, most of which happen to be special cases of structural recursion. More than this, all applications of structural recursion have the form $\texttt{src}[[], g, \frown]$ and the real difficulty is to determine the functions g. For this we identify two cases:

- We obtain a highly recursive operation g that searches through the whole document. Example 3.2 is a prototype for this case.
- We obtain an operation that is determined by the schema. Example 3.1 is a prototype for this case.

As the chances for query optimisation are much higher in the second case – as already seen in Example 3.1 – it will be advantageous to apply this case, wherever it is possible. However, this means to explore the schema. As shown in (Stemple et al. 1990) a type-safe way of doing this is to apply linguistic reflection.

$$\begin{tikzcd}[column sep=huge] \begin{matrix}\text{XQuery}\\ \text{code}\end{matrix} \arrow[r, "\textit{trans}"] \arrow[d, "\textit{drop}"] & \begin{matrix}\text{RTA}\\ \text{code}\end{matrix} \\ \begin{matrix}\text{value of}\\ \text{reflection type}\end{matrix} \arrow[r, "\textit{expand}"] & \begin{matrix}\text{value of}\\ \text{reflection type}\end{matrix} \arrow[u, "\textit{raise}"'] \end{tikzcd}$$

Figure 1: Linguistic Reflection

Linguistic reflection is the ability of a system to observe and manipulate its own components. This is done by extending the system with extra modules which are created, compiled and linked in by the system itself, either during execution or at compile-time. The language in which the system has been written would, of course, need to provide the ability for the system to behave in this manner.

The general idea is to consider constructs in a query that are used for defining an operation, such as /name for $g_{/\text{name}}$ in the previous subsection, as macros that have to be expanded. This can be done by ignoring that they represent query code, thus *drop* this meaning, and treat them as values of some type. The expansion function will then take this value plus the schema, which is represented as a value of some other type, and create a new value. This new value will finally be *raised* back to an executable operation. This is illustrated by Figure 1.

Therefore, we will define reflection types in the next subsection, and finally illustrate, how the expansion procedure for paths works.

4.3 Reflection Types

In order to represent XSchema schemata we need at least reflection types for types, elements, attributes, and schemata. So we get the reflection type $\text{type}_{\text{rep}} = \text{Xtype}_{\text{rep}} \oplus \text{RTtype}_{\text{rep}}$ indicating that we are using types within XSchema and the rational tree types. For the types that are used to describe XSchema types we then get the following definitions:

$\text{Xtype}_{\text{rep}} = \text{xs\_complex\_type}_{\text{rep}} \oplus \text{xs\_simple\_type}_{\text{rep}}$
$\text{xs\_simple\_type}_{\text{rep}} = \textit{String}$
$\text{xs\_complex\_type}_{\text{rep}} = \text{xs\_sequence}_{\text{rep}} \oplus \text{xs\_choice}_{\text{rep}}$
$\text{xs\_sequence}_{\text{rep}} = [(\text{name}_{\text{rep}} \oplus \text{xs\_element}_{\text{rep}},$
$\qquad \text{min}, \text{max} \oplus \textit{Empty})]$
$\text{xs\_choice}_{\text{rep}} = [(\text{name}_{\text{rep}} \oplus \text{xs\_element}_{\text{rep}},$
$\qquad \text{min}, \text{max} \oplus \textit{Empty})]$
$\text{min} = \textit{Integer}$
$\text{max} = \textit{Integer}$
$\text{name}_{\text{rep}} = \textit{String}$
$\text{xs\_element}_{\text{rep}} = (\text{name}_{\text{rep}}, \text{Xtype}_{\text{rep}}, [\text{xs\_attribute}_{\text{rep}}])$
$\text{xs\_attribute}_{\text{rep}} = (\text{name}_{\text{rep}}, \text{xs\_simple\_type}_{\text{rep}}, \text{use}_{\text{rep}})$
$\text{use}_{\text{rep}} = \textit{String}$

and finally

$\text{XSchema}_{\text{rep}} = (\text{namespace}_{\text{rep}}, \text{name}_{\text{rep}},$
$\qquad [\text{xs\_element}_{\text{rep}}])$
$\text{namespace}_{\text{rep}} = \textit{String}$

Example 4.3 The value (1, (1, (1, [((1,“wines”),1,(1,1)), ((1,“wineries”),1,(1,1)), ((1,“regions”),1,(1,1))]))) of type type_{rep} represents the complex type used for the element “cellar” in Example 2.1. The leading 1s indicate that it is a value of an XSchema type, a complex type, and a sequence type, respectively. Note that the min- and max-values are the defaults.

Analogously, the value (1, (1, (1, [((1,”region”),0,(2,⊥))]))) represents the complex type for the element “regions” in Example 2.1.

The value (“http://www.w3.org/2001/XMLSchema”, “cellar”, [(“cellar”, (1, (1, (1, [((1,“wines”),1,(1,1)), ((1,“wineries”),1,(1,1)), ((1,“regions”),1,(1,1))]))), []), . . .]) of type $\text{XSchema}_{\text{rep}}$ represents the schema in Example 2.1. Here the dots stand for representations of all the elements used in the schema.

The value (“wine”, (1, (1, [((2, (“name”, (2, “*String*”), [])), 1, (1,1)), ((2, (“year”, (2, “*Integer*”), [])), 0, (1,1)), ((1, “blend”), 1, (2,⊥)), ((2, (“price”, (2, “*Decimal*”), [])), 1, (1,1))])), [(“w-id”, “*ID*”, “required”), (“producer”, “*IDREF*”, “required”)]) of type $\text{xs\_element}_{\text{rep}}$ represents the element specification for “wine” in Example 2.1. It would of course be part of the value representing the schema.

Analogously, for the types used with RTA we obtain the following reflection types:

$\text{RTtype}_{\text{rep}} = (\text{name}_{\text{rep}}, \text{type\_exp}_{\text{rep}})$
$\text{type\_exp}_{\text{rep}} = \text{base\_type}_{\text{rep}} \oplus \text{label}_{\text{rep}} \oplus \text{record\_type}_{\text{rep}}$
$\oplus\ \text{list\_type}_{\text{rep}} \oplus \text{union\_type}_{\text{rep}}$
$\oplus\ \text{labelled\_type}_{\text{rep}} \oplus \text{name}_{\text{rep}}$
$\text{base\_type}_{\text{rep}} = \mathit{String}$
$\text{label}_{\text{rep}} = \mathit{String}$
$\text{record\_type}_{\text{rep}} = [\text{type\_exp}_{\text{rep}}]$
$\text{list\_type}_{\text{rep}} = \text{type\_exp}_{\text{rep}}$
$\text{union\_type}_{\text{rep}} = [\text{type\_exp}_{\text{rep}}]$
$\text{labelled\_type}_{\text{rep}} = (\text{label}_{\text{rep}}, \text{type\_exp}_{\text{rep}})$

Example 4.4 The type winery from Example 2.1 will be represented by the value (“winery”, (3, [(7, “v-name”), (4,(7, “owner”)), (5,[(7, “area”), (1, “*Empty*”)]), (5,[(7, “established”), (1, “*Empty*”)]), (7, “in-region”)])) of type $\text{RTtype}_{\text{rep}}$.

Similarly, the type in-region is represented by (“in-region”, (2, “r-id”)), and the type wineries is represented by the value (“wineries”, (4, (6, (“w-id”, (7, “wine”))))).

Finally, we also need representation types for the RTA-operations. For this, the following is sufficient:

$\text{Operation}_{\text{rep}} = \text{base\_op}_{\text{rep}} \oplus \text{projection}_{\text{rep}}$
$\oplus\ \text{product}_{\text{rep}} \oplus \text{embedding}_{\text{rep}}$
$\oplus\ \text{sum}_{\text{rep}} \oplus \text{src}_{\text{rep}} \oplus \text{composition}_{\text{rep}}$
$\text{base\_op}_{\text{rep}} = \mathit{String}$
$\text{projection}_{\text{rep}} = \mathit{Integer}$
$\text{product}_{\text{rep}} = [\text{Operation}_{\text{rep}}]$
$\text{embedding}_{\text{rep}} = \mathit{Integer}$
$\text{sum}_{\text{rep}} = [\text{Operation}_{\text{rep}}]$
$\text{src}_{\text{rep}} = \text{Operation}_{\text{rep}} \times \text{Operation}_{\text{rep}} \times \text{Operation}_{\text{rep}}$
$\text{composition}_{\text{rep}} = [\text{Operation}_{\text{rep}}]$

Example 4.5 The values (1,“single”) and (1,“concat”) represent the operations `single` and concatenation $\frown$ on lists, respectively. The value (6, ((1,“empty”), (2,3), (1,“concat”))) represents the operation $\texttt{src}[[], \pi_3, \frown]$. The value (7, [(2,2), (2,1), (1,“single”)]) represents the operation $\pi_2 \circ \pi_1 \circ \texttt{single}$.

4.4 The Expansion Procedure for Paths

Let us now look at the problem of expanding paths, as this turned out to be the core of the translation problem. We have seen above that /name gives rise to a structural recursion operation $\texttt{src}[[], g_{/\text{name}}, \frown]$, so we have to determine a representation of $g_{/\text{name}}$. In order to do so, we first determine the RT type that corresponds to an element in the schema using an operation

$$\text{expand\_elt\_type} : (\mathit{String}, \text{XSchema}_{\text{rep}}) \to \text{RTtype}_{\text{rep}},$$

i.e. we associate with an element name and a representation of a schema a rational tree type. This can be achieved by parsing through the schema representation and then applying the rules for type transformation that we used in Section 2. In particular, we blur the differences between subelements and attributes, and we replace references by occurrences of the base type *ID*:

```
$\text{expand\_elt\_type}(n, S) =$
    $\text{expand\_elt\_type}'(n, \text{search}(n, \pi_3(S)), S)$

$\text{search}(n, S) =$
    if $\pi_1(\text{first}(S)) = n$
    then $(\pi_2 \times \pi_3)(\text{first}(S))$
    else $\text{search}(n, \text{rest}(S))$
    endif

$\text{expand\_elt\_type}'(n, (e, L), S) =$
    case $\pi_1(\pi_2(e)) = 2$
      then $(n, (1, \pi_2(\pi_2(e))))$
    case $\pi_1(\pi_2(\pi_2(e))) = 1$
      then $(n, (3, \text{check\_ID}($
              $\text{parse\_seq}(\pi_2(\pi_2(\pi_2(e)))^\frown L, S))))$
    case $\pi_1(\pi_2(\pi_2(e))) = 2$
      then $(n, (5, \text{check\_ID}($
              $\text{parse\_seq}(\pi_2(\pi_2(\pi_2(e)))^\frown L, S))))$
    endcase

$\text{parse\_seq}(L, S) =$
    if $L = []$
    then $[]$
    elsif $\text{first}(L) = ((1, n'), m, M)$
    then if $m = 1 \wedge M = (1, 1)$
          then $[\pi_2(\text{expand\_elt\_type}(n', S))]^\frown$
                $\text{parse\_seq}(\text{rest}(L), S)$
          elsif $m = 0 \wedge M = (1, 1)$
          then $[(5, [\pi_2(\text{expand\_elt\_type}(n', S)),$
                $(1, \text{“}\mathit{Empty}\text{”})])]^\frown \text{parse\_seq}(\text{rest}(L), S)$
          else $[(4, \pi_2(\text{expand\_elt\_type}(n', S)))]^\frown$
                $\text{parse\_seq}(\text{rest}(L), S)$
          endif
    elsif $\text{first}(L) = ((2, e), m, M)$
    then if $m = 1 \wedge M = (1, 1)$
          then $[\pi_2(\text{expand\_elt\_type}'(\pi_1(e), (\pi_2(e),$
                $\pi_3(e)), S))]^\frown \text{parse\_seq}(\text{rest}(L), S)$
          elsif $m = 0 \wedge M = (1, 1)$
          then $[(5, [\pi_2(\text{expand\_elt\_type}'(\pi_1(e), (\pi_2(e),$
                $\pi_3(e)), S)), (1, \text{“}\mathit{Empty}\text{”})])]^\frown$
                    $\text{parse\_seq}(\text{rest}(L), S)$
          else $[(4, \pi_2(\text{expand\_elt\_type}'(\pi_1(e), (\pi_2(e),$
                $\pi_3(e)), S)))]^\frown \text{parse\_seq}(\text{rest}(L), S)$
          endif
    elsif $\text{first}(L) = (n', t, u)$
    then if $t \neq \text{“}\mathit{IDREFS}\text{”}$
          then $[(1, t)]^\frown \text{parse\_seq}(\text{rest}(L), S)$
          else $[(4, (1, \text{“}\mathit{ID}\text{”}))]^\frown \text{parse\_seq}(\text{rest}(L), S)$
          endif
    endif

$\text{check\_ID}(L) =$
    if $L = []$
    then $[]$
    elsif $\pi_1(\text{first}(L)) = 1 \wedge \pi_2(\text{first}(L)) = \text{“}\mathit{ID}\text{”}$
    then $(3, [\text{first}(L), (3, \text{rest}(L))])$
    elsif $\text{first}(\text{check\_ID}(\text{rest}(L))) = (1, \text{“}\mathit{ID}\text{”})$
    then $(3, [(1, \text{“}\mathit{ID}\text{”}), (3, [\text{first}(L)]^\frown$
          $\pi_2(\text{first}(\text{rest}(\text{check\_ID}(\text{rest}(L))))))])$
    else $L$
    endif
```

Example 4.6 If S represents the schema from Example 2.1, we obtain

$$\begin{aligned}
&\text{expand\_elt\_type}(\text{“wine”}, S) = \\
&\quad (\text{“wine”}, (3, [(1, \text{“}\mathit{ID}\text{”}), (3, [(1, \text{“}\mathit{String}\text{”}), \\
&\qquad (5, [(1, \text{“}\mathit{Integer}\text{”}), (1, \text{“}\mathit{Empty}\text{”})]), \\
&\qquad (4, (3, [(1, \text{“}\mathit{String}\text{”}), (1, \text{“}\mathit{Integer}\text{”})])), \\
&\qquad (1, \text{“}\mathit{Decimal}\text{”}), (1, \text{“}\mathit{ID}\text{”})])]))
\end{aligned}$$

and expand_elt_type(“name”,S) = (“w-name”, (1,“*String*”)) assuming in the latter case that we add some renaming to avoid name-conflicts.

Now that we got the transformation of types, we can define the expansion of paths, i.e. we get an operation

$$\text{expand\_elt} : (\mathit{String}, \text{XSchema}_{\text{rep}}) \to \text{Operation}_{\text{rep}}$$

such that expand_elt(n, S) will be a representation of the operation $g_{/n}$. In order to define the expansion we need both the types for the element n and its parent, and the position at which the type of n appears inside the type of its parent. We then parse through the parent type and determine the operation $g_{/n}$ according to the given position:

```
expand_elt$(n, s)$ =
    parse_type$(\pi_2(\text{expand\_elt\_type}(n, S)),$
        $\pi_2(\text{first}(\text{parents}(n, \pi_3(S)))),$
        $\pi_2(\text{expand\_elt\_type}($
            $\pi_1(\text{first}(\text{parents}(n, \pi_3(S)))), S)))$
parents$(n, L)$ =
    if $L = []$
    then $[]$
    else list_match$(n, \pi_1(\text{first}(L)),$
      $\pi_2(\pi_2(\pi_2(\text{first}(L)))), 1) \frown \text{parents}(n, \text{rest}(L))$
    endif
list_match$(n, n', L, i)$ =
    if $L = []$
    then $[]$
    else match$(n, n', \pi_1(\text{first}(L)), i)$
        $\frown$list_match$(n, n', \text{rest}(L), i + 1)$
    endif
match$(n, n', e, i)$ =
    if $\pi_1(e) = 1 \wedge \pi_2(e) = n$
    then $[(n', i)]$
    elsif $\pi_1(e) = 2 \wedge \pi_1(\pi_2(e)) = n$
    then $[(n', i)]$
    else $[]$
    endif
parse_type$(t, i, t')$ =
    if $\pi_1(t') = 3 \wedge \text{first}(\pi_2(t')) = (1, \text{“}\mathit{ID}\text{”})$
    then $(7, [\text{parse\_type}(t, i, \text{first}(\text{rest}(\pi_2(t')))), (2, 2)])$
    else case $t = t'$
             then $(1, \text{“id”})$
           case $\pi_1(t') = 3$
             then $(2, i)$
           case $\pi_1(t') = 4$
             then $(1, \text{“id”})$
           case $\pi_1(t') = 5$
             then $(7, [(1, \text{“if\_else”}), (3, [(7, [(1, \text{“}\mathit{eq''}),$
               $(3, [(2, 1), (1, i)])]), (1, \text{“single”}),$
               $(7, [(1, \text{“empty”}), (1, \text{“triv”})])])])$
           endcase
    endif
```

Example 4.7 If S represents the schema in Example 2.1 we obtain expand_elt(“wine”, S) = (1, “*id*″), and expand_elt(“name”, S) = $(7, [(2, 1), (2, 2)])$.

Similarly, we get an operation expand_att such that expand_att(n, S) will be a representation of the operation $g_{/@n}$. We omit the details.

Finally, let us look at the expansion of paths containing some //name. In this case we get an operation

$$\text{expand\_elt}^* : (\mathit{String}, \text{XSchema}_{\text{rep}}) \to \text{Operation}_{\text{rep}}$$

such that expand_elt$^*(n, S)$ will be a representation of the operation $g_{//n}$. We already saw the general structure of this operation, when we discussed the basic model of the translation into RTA, so let us now concentrate on the Boolean operations only. The only condition that involves the element name n is φ_1. So, let

$$\text{expand\_bool1} : (\mathit{String}, \text{XSchema}_{\text{rep}}) \to \text{Operation}_{\text{rep}}$$

be such that expand_bool1(n, S) will be a representation of the operation φ_1 associated with n. Thus, we get:

$$\begin{aligned}
&\text{expand\_bool1}(n, S) = \\
&\quad (7, [(1, \text{“eq”}), (3, [(1, \text{“type”}), \\
&\qquad (7, [(1, \text{expand\_elt\_type}(n, S)), (1, \text{“triv”})])])])
\end{aligned}$$

The other operations can be obtained analogously.

5 Conclusion

In this paper we addressed the translation of XQuery to a complex value query algebra. The model underlying this algebra is based on a type system that supports constructors for records, lists and unions as well as optionality and references. This captures the rational tree structures represented by XML documents. The query algebra uses operations defined on this type system, in particular structural recursion for lists.

We demonstrated that the basic execution model of XQuery easily gives rise to nested structural recursion. However, the function parameters involved in these operations require complex definitions resulting from information about the schema. These functions can be generated using compile-time linguistic reflection.

An obvious advantage of this approach is type safety. Furthermore, a translation to a simple query algebra enables the support of algebraic query optimisation, the easy implementation of the operations and thus the integration with programming languages, the easy extension to other constructors such as sets and multisets in case the order that comes with the list constructor is considered unnecessary or even undesired. In particular, much of the complexity resulting from path expressions is captured in the translation process.

References

Abiteboul, S., Buneman, P. & Suciu, D. (2000), *Data on the Web: From Relations to Semistructured Data and XML*, Morgan Kaufmann Publishers.

Abiteboul, S., Quass, D., McHugh, J., Widom, J. & Wiener, J. (1997), ‘The LOREL query language for semi-structured data’, *International Journal on Digital Libraries* **1**(1), 68–88.

Buneman, P., Davidson, S., Hillebrand, G. & Suciu, D. (1996), A query language and optimization techniques for unstructured data, *in* ‘Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data’, Montréal, Canada, pp. 505–516.

Chen, Y., Davidson, S. B. & Zheng, Y. (2004), BLAS: An efficient XPath processing system, *in* G. Weikum, A. C. König & S. Deßloch, eds, 'Proceedings of the 2004 ACM SIGMOD-SIGART-SIGACT Symposium on Principles of Database Systems (SIGMOD 2004)', ACM, Paris, France, pp. 47–58.

DeHaan, D., Toman, D., Consens, M. P. & Özsu, M. T. (2003), A comprehensive XQuery to SQL translation using dynamic interval encoding, *in* A. Y. Halevy, Z. G. Ives & A. Doan, eds, 'Proceedings of the 2003 ACM SIGMOD-SIGART-SIGACT Symposium on Principles of Database Systems (SIGMOD 2003', ACM, San Diego, California, USA, pp. 623–634.

Deutsch, A., Fernandez, M., Florescu, D., Levy, A. & Suciu, D. (1999), 'A query language for XML', *Computer Networks* **31**(11-16), 1155–1169.

Gottlob, G., Koch, C. & Pichler, R. (2003), The complexity XPath query evaluation, *in* 'Proceedings of the 22nd ACM SIGMOD-SIGART-SIGACT Symposium on Principles of Database Systems (PoDS 2003)', ACM, San Diego, California, USA, pp. 179–190.

Katz, H., ed. (2003), *XQuery from the Experts – A Guide to the W3C XML Query Language*, Addison-Wesley.

Kirchberg, M., Schewe, K.-D. & Tretiakov, A. (2003), A multi-level architecture for distributed object bases, *in* 'Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS)', Vol. 1, ICEIS Press, pp. 63–70.

Koch, C. (2005), On the complexity of nonrecursive xquery and functional query languages on complex values, *in* 'Principles of Database Systems', ACM.

Lobin, H. (2001), *Informationsmodellierung in XML und SGML*, Springer-Verlag.

Marx, M. (2004), Conditional XPath, the first order complete XPath dialect, *in* 'Proceedings of the 23rd ACM SIGMOD-SIGART-SIGACT Symposium on Principles of Database Systems (PoDS 2004)', ACM, Paris, France, pp. 13–22.

Paparizos, S., Wu, Y., Lakshmanan, L. V. S. & Jagadish, H. V. (2004), Tree logical classes for efficient evaluation of XQuery, *in* G. Weikum, A. C. König & S. Deßloch, eds, 'Proceedings of the 2004 ACM SIGMOD-SIGART-SIGACT Symposium on Principles of Database Systems (SIGMOD 2004)', ACM, Paris, France, pp. 71–82.

Ruecker, S. (2000), A conceptual taxonomy of SGML tags, *in* X. Zhou, J. Fong, X. Jia, Y. Kambayashi & Y. Zhang, eds, 'WISE 2000, Proceedings of the First International Conference on Web Information Systems Engineering, Volume II (Workshops)', IEEE Computer Society, pp. 2–10.

Schewe, K.-D. (2001), On the unification of query algebras and their extension to rational tree structures, *in* M. E. Orlowska & J. F. Roddick, eds, 'Database Technologies – Proceedings of the 12th Australasian Database Conference (ADC 2001)', IEEE Computer Society, Gold Coast, Queensland, Australia, pp. 52–59.

Siméon, J. & Wadler, P. (2003), The essence of XML, *in* 'Proceedings of the 30th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2003)', ACM.

Stemple, D. W., Fegaras, L., Sheard, T. & Socorro, A. (1990), Exceeding the limits of polymorphism in database programming languages, *in* F. Bancilhon, C. Thanos & D. Tsichritzis, eds, 'Advances in Database Technology - EDBT'90', Vol. 416 of *LNCS*, Springer-Verlag, pp. 269–285.

Tannen, V., Buneman, P. & Wong, L. (1992), Naturally embedded query languages, *in* J. Biskup & R. Hull, eds, 'Database Theory (ICDT 1992)', Vol. 646 of *LNCS*, Springer-Verlag, pp. 140–154.

Tatarinov, I., Ives, Z., Halevy, A. & Weld, D. (2001), Updating XML, *in* 'Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data', Santa Barbara, California, pp. 413–424.

Wadler, P. (1992), 'Comprehending monads.', *Mathematical Structures in Computer Science* **2**(4), 461–493.

World Wide Web Consortium (W3C) (2001), 'XML Schema', `http://www.w3c.org/TR/xmlschema-0`, `http://www.w3c.org/TR/xmlschema-1`, `http://www.w3c.org/TR/xmlschema-2`.

World Wide Web Consortium (W3C) (2004), 'XQuery', `http://www.w3c.org/TR/xquery`.

An Optimization for Query Answering on $\mathcal{ALC}$ Database

Pakornpong Pothipruk **Guido Governatori**

School of Information Technology and Electrical Engineering
University of Queensland, Australia 4072,
Email: pkp@itee.uq.edu.au

Abstract

Query answering over OWLs and RDFs on the Semantic Web is, in general, a deductive process. To this end, OWL, a family of web ontology languages based on description logic, has been proposed as the language for the Semantic Web. However, reasoning even on $\mathcal{ALC}$, a description logic weaker than OWL, faces efficiency problem. To obviate this problem, at least for $\mathcal{ALC}$, we propose a partition approach that improves the efficiency by splitting the search space into independent Aboxes. Each partition class, i.e., an Abox, can be queried independently. The answer to a query is the simple combination of the answers from each Abox. We prove the correctness of this approach and we outline how to represent compactly the content of each independent Abox. This work can be seen as an optimization for querying a deductive semi-structured database.

Keywords: Description Logic, Query Optimization, Web Database.

1 Introduction

The *Semantic Web*, originated from an idea of the creator of the Web Tim Berners-lee (Berners-Lee 1999), is an effort to bring back structure to information available on the Web. The structures are semantic annotations that conform to an explicit specification (called ontology) of the intended meaning of a piece of information. Thus the the Semantic Web contains implicit knowledge, and information on the Semantic Web is often incomplete since it assumes open-world semantics. In this perspective query answering on the Semantic Web is a deductive process (Stuckenschmidt 2003).

RDF, a semi-structure data page, is a basic component for the Semantic Web. Thus, in the Semantic Web perspective, there are huge number of RDF data pages. In addition, A family of web ontology languages (OWL) based on *Description Logic* (DL) has been proposed as the languages to represent and reason with the Semantic Web. In the nutshell, querying the Semantic Web is the process of reasoning over OWLs and RDFs, based on DL reasoning. DL emphasizes clear unambiguous languages supported by complete denotational semantics and tractable/intractable reasoning algorithms (McGuinness, Fikes, Stein & Hendler 2003). Nevertheless, DL still faces problems when applied in context of the Web. One of them is the efficiency of query answering. Consequently, there is an urgent need of optimizations for querying the Semantic Web.

There are many works about DL reasoning optimization. However, most of them focus only on ontology reasoning with out any data, i.e., DL-Tbox reasoning. In fact, DL-Abox reasoning, which is the basis for querying OWL and RDF on the Semantic Web, was seriously studied by some researchers recently. At present, there are only two prominent works for DL-Abox reasoning optimization, i.e., Instance Store (Horrocks, Li, Turi & Bechhofer 2004) and RACER (Haarslev & Möller 2002). Nevertheless, none of above works can eliminate a chunk of individuals at-a-time from retrieval reasoning. Thus, we create optimization techniques that support this idea. We present here a novel optimization technique for instance checking, an Abox reasoning. We also introduce a technique of instance retrieval, another Abox reasoning.

2 Preliminary: Description Logic

The Semantic Web community implicitly adopted DL as a core technology for the ontology layer. One of the reasons behind this is that this logic have been heavily analyzed in order to understand how constructors interact and combine to affect tractable reasoning, see (Donini, Lenzerini, Nardi & Nutt 1991). Technically, we can view the current Semantic Web, not including rule, proof and trust layers, as a DL knowledge base. Thus, answering a query posed on the Semantic Web (RDF and ontology layers) can be reduced to answering a query posed on a DL knowledge base, not taking into account low-level operations, such as name space resolution.

Description logic itself can be categorized into many different logics, distinguished by the set of constructors they provide. We focus on $\mathcal{ALC}$ description logic since it is the basis of many DL systems.

The language of $\mathcal{ALC}$ consists of an alphabet of distinct concept names **CN**, role names **RN**, and individual names **IN**, together with a set of constructors for building concept and role expressions (Tessaris 2001).

Formally, a description logic knowledge base is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is a Tbox, and $\mathcal{A}$ is an Abox. The Tbox contains a finite set of axiom assertions. Axiom assertions are of the form

$$C \sqsubseteq D \mid C \doteq D,$$

where C and D are concept expressions. Concept expressions are of the form

$$A \mid \top \mid \perp \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C,$$

where A is an atomic concept or concept name in **CN**, R is a role name in **RN**, $\top$ (top or full domain) is the most general concept, and $\perp$ (bottom or empty set) is the least general concepts. The Abox contains a finite set of assertions about individuals of the form $a : C$ or $C(a)$ (concept membership assertion) and $(a,b) : R$ or $R(a,b)$ (role membership assertion), where a, b are names in **IN**.

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

The semantics of description logic are defined in terms of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \bullet^{\mathcal{I}})$, consisting of a nonempty domain $\Delta^{\mathcal{I}}$ and a interpretation function $\bullet^{\mathcal{I}}$. The interpretation function maps concept names into subsets of the domain ($A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$), role names into subsets of the Cartesian product of the domain ($R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$), and individual names into elements of the domain. The only restriction on the interpretations is the so called unique name assumption (UNA), which imposes that different individual names must be mapped into distinct elements of the domain. Given a concept name A (or role name R), the set denoted by $A^{\mathcal{I}}$ (or $R^{\mathcal{I}}$) is called the *interpretation*, or *extension*, of A (or R) with respect to $\mathcal{I}$.

The interpretation is extended to cover concepts built from negation ($\neg$), conjunction ($\sqcap$), disjunction ($\sqcup$), existential quantification ($\exists R.C$) and universal quantification ($\forall R.C$) as follows:

$$\begin{aligned}
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &= \left\{x \in \Delta^{\mathcal{I}} | \exists y. \langle x,y\rangle \in R^{\mathcal{I}} \land y \in C^{\mathcal{I}}\right\} \\
(\forall R.C)^{\mathcal{I}} &= \left\{x \in \Delta^{\mathcal{I}} | \forall y. \langle x,y\rangle \in R^{\mathcal{I}} \to y \in C^{\mathcal{I}}\right\}
\end{aligned}$$

An interpretation $\mathcal{I}$ satisfies (entails) an inclusion axiom $C \sqsubseteq D$ (written $\mathcal{I} \models C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it satisfies an equality $C \doteq D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$. It satisfies a Tbox $\mathcal{T}$ if it satisfies each assertion in $\mathcal{T}$. The interpretation $\mathcal{I}$ satisfies a concept membership assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and satisfies a role membership assertion $R(a,b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. $\mathcal{I}$ satisfies an Abox $\mathcal{A}$ (written $\mathcal{I} \models \mathcal{A}$) if it satisfies each assertion in $\mathcal{A}$. If $\mathcal{I}$ satisfies an axiom (or a set of axioms), then we say that it is a *model* of the axiom (or the set of axioms). Two axioms (or two sets of axioms) are *equivalent* if they have the same models. Given a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ we will say that the knowledge bases entails an assertion α (written $\mathcal{K} \models \alpha$) iff for every interpretation $\mathcal{I}$, if $\mathcal{I} \models \mathcal{A}$ and $\mathcal{I} \models \mathcal{T}$, then $\mathcal{I} \models \alpha$.

The DL Abox can be viewed as a semi-structured database, consisting of a collection of RDF data, while the DL Tbox contains a set of constraints for the data in Abox. Thus, the Tbox can be compared to data modeling in database, e.g., Entity-Relationship data model. However, the semantics of description logics are defined in terms of an interpretation which differentiate description logics from databases. In addition, the domain of interpretation can be chosen arbitrarily, and it can be infinite. The non-finiteness of the domain and the open-world assumption are distinguishing features of description logics with respect to the modeling languages in the database. Even description logics, as modeling languages, overlap to a large extent with modeling languages in database (Baader, Calvanese, McGuinness, Nardi & Patel-Schneider 2003), the particular feature of description logics is in the reasoning capabilities that are associated with it. In other words, while modeling has general significance, the capability of exploiting the description of the model to draw conclusions about the problem at hand is a particular advantage of modeling using description logics. In the next section, we introduce basic reasoning tasks in description logics.

2.1 Reasoning in Description Logic

A description logic knowledge base basically supports two major kinds of reasoning tasks, i.e., Tbox reasoning (ontology reasoning), and Abox reasoning (data reasoning with ontology). Note that, both kinds of reasonings are based on satisfiability problem. Also, keep in mind that, like other logic system, the knowledge base contains implicit knowledge that can be made explicit through inferences.

2.1.1 Reasoning for Tbox

In description logic, basic reasoning services for concepts in Tbox $\mathcal{T}$ consist of (Baader et al. 2003):

- Knowledge base consistency: a Tbox $\mathcal{T}$ is consistent iff it is satisfiable, i.e., there is at least a non empty model for $\mathcal{T}$. An interpretation $\mathcal{I}$ is a model for $\mathcal{T}$ if it satisfies every assertion in $\mathcal{T}$.
- Satisfiability: a concept C is satisfiable with respect to $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is nonempty. $\mathcal{I}$ is also a model of C.

Definition 1 ***(Satisfiability w.r.t. Knowledge base)*** *A concept expression C is satisfiable with respect to a knowledge base Σ if there exists a model $\mathcal{I}$ of Σ such that $C^{\mathcal{I}}$ is nonempty, $(\exists \mathcal{I}, \mathcal{I} \models \Sigma \land \mathcal{I} \models C)$. $\mathcal{I}$ is also a model of C.*

Note that the Definition 1 resembles the definition of entailment, but not the same. Thus, we also present here the definition of entailment and its complement for the sake of clarity.

Definition 2 ***(Entailment)*** *A concept C is entailed by a knowledge base Σ $(\Sigma \models C)$ if, for every models $\mathcal{I}$ of Σ, each $\mathcal{I}$ is also a model of C $(\forall \mathcal{I}, \mathcal{I} \models \Sigma \to \mathcal{I} \models C)$.*

Definition 3 ***(Non-entailment)*** *A concept C is not entailed by a knowledge base Σ $(\Sigma \not\models C)$ if there exists model $\mathcal{I}$ of Σ such that $\mathcal{I}$ is not a model of C, i.e., $C^{\mathcal{I}}$ is empty $(\exists \mathcal{I}, \mathcal{I} \models \Sigma \land \mathcal{I} \not\models C)$.*

However, entailment problem can be reduced to satisfiability problems.

Definition 4 ***(Entailment Reduction to Unsatisfiability)*** *A concept C is entailed by a knowledge base Σ $(\Sigma \models C)$ iff $\Sigma \cup \{\neg C\}$ is unsatisfiable, i.e has no model, or $\forall \mathcal{I}, \mathcal{I} \models \Sigma \to \mathcal{I} \not\models \neg C$.*

Definition 5 ***(Non-entailment Reduction to Satisfiability)*** *A concept C is not entailed by a knowledge base Σ $(\Sigma \not\models C)$ iff $\Sigma \cup \{\neg C\}$ is satisfiable, i.e., model exists, or $\exists \mathcal{I}, \mathcal{I} \models \Sigma \land \mathcal{I} \models \neg C$.*

- Subsumption: a concept D subsumes a concept C with respect to $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$.
- Equivalence: two concepts C and D are equivalent with respect to $\mathcal{T}$ ($C \doteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \doteq D$) if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$.
- Disjointness: two concepts C and D are disjoint with respect to $\mathcal{T}$ if, for every model $\mathcal{I}$ of $\mathcal{T}$, $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$.

If the Tbox T is clear from the context, we can drop the qualification "with respect to $\mathcal{T}$ ". In the special case where Tbox is empty, we simply write , $\models C \sqsubseteq D$ if D subsumes C, and , $\models C \doteq D$ if C and D are equivalent.

Subsumption, equivalence and disjointness tests can be reduced to concept (un)satisfiability test, which is, in turn, can be achieved through tableaux algorithm (see section 2.1.3). Consequently, concept satisfiability test is the key inference for Tbox reasoning.

- Subsumption: C is subsumed by D ($C \sqsubseteq D$) iff $C \sqcap \neg D$ is unsatisfiable with respect to $\mathcal{T}$.
- Equivalence: C and D are equivalent ($C \doteq D$) iff both $C \sqcap \neg D$ and $D \sqcap \neg C$ are unsatisfiable with respect to $\mathcal{T}$.
- Disjointness: C and D are disjoint iff $C \sqcap D$ is unsatisfiable with respect to $\mathcal{T}$.

However, in the description logic without full negation, e.g., $\mathcal{AL}$, subsumption and equivalence cannot be reduced to unsatisfiability test.

2.1.2 Reasoning for Abox

Recall that Abox consists of only two kinds of assertion: concept membership assertion of the form $C(a)$ and role membership assertion of the form $R(a,b)$. Hence Abox alone cannot be seen as a knowledge base, it must be coupled with Tbox. Consequently, Abox reasoning will always be done with respect to a Tbox.

In description logic, basic reasoning services for Abox consist of (Baader et al. 2003):

- Instantiation test or instance check: determine whether an assertion is entailed by Abox $\mathcal{A}$ or not. Since, in this work, we consider only $\mathcal{ALC}$ which does not contain any role constructor to form complex roles, role membership assertion test will be easy, i.e., simply find the occurrence of that role assertion in the Abox. At the time of writing, instance check generally refers to only concept membership assertion testing. To check for a concept membership assertion, we just check whether the assertion is entailed by the Abox. The assertion is entailed by Abox ($\mathcal{A} \models C(a)$) if every interpretation that satisfies $\mathcal{A}$, i.e., every model of $\mathcal{A}$, also satisfies $C(a)$.
- Realisation: given an individual a and a set of concepts, find the most specific concept C from the set such that $\mathcal{A} \models C(a)$.
- Retrieval: given an Abox $\mathcal{A}$ and concept C, find all individuals a such that $\mathcal{A} \models C(a)$.
- Abox consistency: Abox $\mathcal{A}$ is consistent iff it is consistent with respect to Tbox $\mathcal{T}$. Consequently, we must use Tbox in this reasoning, i.e., for $\mathcal{ALC}$, expanding the Abox with unfolded Tbox concepts (Tessaris 2001). Unfolded concept or expanded concept C' is obtained by replacing names in the description of the original concept C with their descriptions in $\mathcal{T}$. Note that C is satisfiable with respect to $\mathcal{T}$ iff C' is satisfiable, i.e., the original concept and the consistency preserving expanded/unfolded concept are in fact equivalent, $C \doteq_{\mathcal{T}} C'$ (Horrocks 1997). Therefore expansion of $\mathcal{A}$ with respect to $\mathcal{T}$ (the Abox $\mathcal{A}'$) can be obtained by replacing each concept assertion $C(a)$ in $\mathcal{A}$ with the assertion $C'(a)$. In every model of $\mathcal{T}$, a concept C and its expansion C' are interpreted in the same way. Hence, $\mathcal{A}$ is consistent with respect to $\mathcal{T}$ iff $\mathcal{A}'$ is consistent. $\mathcal{A}'$ is consistent iff it is satisfiable, i.e., there is at least a nonempty model for $\mathcal{A}'$. Note that $\mathcal{A}'$ also represents the whole knowledge base $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$.

Theorem 6 ***(Expanded Abox)*** *Given a satisfiable Tbox $\mathcal{T}$, an Abox $\mathcal{A}$ is satisfiable with respect to $\mathcal{T}$ iff its expansion $\mathcal{A}'$ is satisfiable.*

Proof *see (Baader et al. 2003)*

It is easy to see that Theorem 7 logically follows from Theorem 6.

Theorem 7 ***(Expanded Abox Entailment)*** *Given a satisfiable Tbox $\mathcal{T}$, a knowledge base $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ entails a concept expression Q ($\langle \mathcal{T}, \mathcal{A} \rangle \models Q$) iff $\mathcal{A}'$ entails Q ($\mathcal{A}' \models Q$).*

Realisation and retrieval can be reduced to instantiation test. They can be done through series of instantiation tests. In addition, we can reduce instantiation test to consistency problem for Abox since $\mathcal{A} \models C(a)$ iff $\mathcal{A} \cup \{\neg C(a)\}$ is inconsistent. Concept satisfiability can also be reduced to Abox consistency test since C is satisfiable iff $\{C(a)\}$ is consistent, where a is an arbitrary individual name.

At this point, it turns out that there is one main inference problem, i.e., consistency check for Abox, to which all other reasoning services can be reduced. Since tableaux algorithms can be used to solve such test, thus all of the reasoning services in description logic can be achieved through the application of tableaux algorithm (Baader et al. 2003). In addition, most of the practical description logic systems, such as FACT (Horrocks 1997) and RACER (Haarslev & Möller 2002), exploit tableaux algorithm as the basis for their reasoning. In fact, tableaux becomes the de-facto standard for reasoning in description logic. We will explain tableaux in section 2.1.3.

2.1.3 Tableaux algorithm

Traditionally, tableaux algorithm was designed to prove the satisfiability of a concept expression. The main idea behind this algorithm (Baader & Hollunder 1991) is based on notational variant of the first order tableaux calculus. In fact, a tableaux algorithm tries to prove the satisfiability of a concept expression C by demonstrating a nonempty model of C. It constructively builds a model for a given concept. If it can build a model, then the concept is satisfiable.

First, we will present tableaux algorithm for testing the satisfiability of a Tbox concept and describe its operation. Then, we will show how this algorithm can also be applied to Abox consistency test. However, this topic is intended to illustrate the general principles only. For proof of termination, soundness and completeness of the algorithm, consult (Baader et al. 2003).

Before we proceed with the algorithm, we need to introduce an appropriate data structure for representation of each tableau. The original paper by Schmidt-Schauβ and Smolka (Schmidt-Schauß & Smolka 1991), and also many other papers on tableaux algorithms for description logic (Baader et al. 2003), exploit a notion of a constraint system for this purpose. A constraint system is a set of constraints, which are syntactic elements of the forms:

• $x : C$	concept constraint
• $\langle x, y \rangle : R$	role constraint
• $x \neq y$	inequality constraint

Although the syntax is similar to the one for Abox assertions, there is a difference in the meaning of the x and y terms. Since we are going to test the satisfiability of a Tbox concept expression, no individual is involved. Consequently, x and y are not individuals, but variables, so the unique name assumption does not apply to them unless stated explicitly by an inequality constraint. The presence of at-most number restriction may lead to the identification of different individual names. However, our considered description logic does not contain at-most number restriction constructor, thus we do not need inequality constraint here. A constraint system can be seen as a graph where the nodes are individual variable names, and the edges corresponds to role names. Each node has associated concept expressions which are, in fact, constraints of the variable corresponded to the node.

To test satisfiability of a concept expression C with respect to Tbox $\mathcal{T}$, firstly, unfold the concept expressions with their definitions defined in $\mathcal{T}$, and get the new expanded concept C'. Next, transform the concept expression into a negation normal form (NNF), i.e., push negation signs as far as possible into the description, using combination of de Morgan's law and usual rules for quantifiers, for example $\neg\exists R.C = \forall R.\neg C$ and $\neg\forall R.C = \exists R.\neg C$. The concept expression is in negation normal form if negations apply only to concept names, not to any compound terms. For example the concept $C = \neg\exists R.A \sqcap \neg(\exists R.D \sqcup \forall S.\neg D)$ can be transformed into NNF, $C_0 = \forall R.\neg A \sqcap \forall R.\neg D \sqcap \exists S.D$. Now, we get a new concept expression as a constraint system S.

The process of constructing a model proceeds by completing (or extending) a constraint system S, using a set of consistency preserving transformation-completion (or expansion) rules in figure 2.1.3. These rules modify a constraint system by adding or rewriting the constraints. Apply these rules to the constraint system until no more rules apply. When no rule is applicable, the constraint system is said to be *completed* if there is no obvious contradiction, so called *clash*, i.e., $\{x : A, x : \neg A\}$ or $\{x : \bot\}$, occurs. The concept C' is consistent iff there exists a completed constraint system, i.e., a nonempty model. Since $C \doteq_{\mathcal{T}} C'$, the concept C is consistent with respect to $\mathcal{T}$ iff the expanded concept C' is consistent.

The $\rightarrow_{\sqcap}$ -rule
Condition: $x : C_1 \sqcap C_2$ is in S, and both $x : C_1$and $x : C_2$ are not in S.
Action: $S = S \cup \{x : C_1, x : C_2\}$

The $\rightarrow_{\sqcup}$ -rule
Condition: $x : C_1 \sqcup C_2$ is in S, and neither $x : C_1$nor $x : C_2$ is in S.
Action: $S' = S \cup \{x : C_1\}$, $S'' = S \cup \{x : C_2\}$

The $\rightarrow_{\forall}$ -rule
Condition: $x : \forall R.C$ is in S, $\langle x, y\rangle : R$ is in S, and $y : C$ is not in S.
Action: $S = S \cup \{y : C\}$

The $\rightarrow_{\exists}$ -rule
Condition: $x : \exists R.C$ is in S,
and there is no z such that both $\langle x, z\rangle : R$ and $z : C$ are in S.
Action: $S = S \cup \{\langle x, y\rangle : R, y : C\}$ where y is new variable name.

Figure 1: Completion rules for $\mathcal{ALC}$ satisfiability test

Note that the only rule that is non-deterministic is the disjunction rule ($\rightarrow_{\sqcup}$). For more detail proof of these rules, see (Donini, Lenzerini, Nardi & Schaerf 1996).

In order to test Abox consistency using tableaux algorithm, we simply allow x and y in constraint system notation to be able to represent individual name. Consequently, concept constraint and role constraint in a constraint system become concept membership assertion and role membership assertion in Abox respectively. The algorithm is the same as above, except, instead of a concept expression, we try to prove satisfiability of the whole expanded Abox $\mathcal{A}'$.

2.2 Reasoning Complexity and Optimization

Since, all basic reasoning services can be reduced to satisfiability problem, which, in turn, can be achieved through tableaux algorithm, complexity of reasoning services provided in almost description logic systems are based on complexity of tableaux algorithm. Recall that description logic was invented because of inefficiency of first-order logic. In fact, description logic is one of two major approaches for mitigation of inefficiency of reasoning for logic. Its prominent significance is that trade-offs between expressive power and efficiency of reasoning were studied throughout a decade.

We denote problem that is solvable using algorithm with polynomial time (PTIME) worst-case complexity as *tractable* problem, and *intractable* problem otherwise. Normally, tractable is preferred. However, there is worse class of problem, *undecidable* problem. We says the problem is undecidable if there is no solving algorithm that terminates, and *decidable* otherwise. To the best of our knowledge, the most expressive description logic that its concept satisfiability problem is decidable in PTIME worst-case complexity is $\mathcal{ALN}$.

Logics with PTIME worst-case complexity have been criticized for their too limited expressive power (Doyle & Patil 1991). Thus we need the language that is more expressive. Consequently, we choose $\mathcal{ALC}$ which is expressive enough, since it is a subset of mostly every expressive description logics (Donini & Massacci 2000).

3 The Efficiency Issue

In DL, there are two standard types of queries allowed, i.e., boolean query and non-boolean query, which are in turn instance checking (or instantiation test) and retrieval Abox reasoning services respectively.

A boolean query $\mathcal{Q}_b$ refers to a formula of the form

$$\mathcal{Q}_b \leftarrow QExp,$$

where *QExp* is an assertion about individual, e.g.,

$$\mathcal{Q}_b \leftarrow Tom : (Parent \sqcap \exists hasChild.Employee)$$

The query will return one of the member of the boolean set $\{True, False\}$. $\mathcal{Q}_b$ will return *True* if and only if every interpretation that satisfies the knowledge base $\mathcal{K}$ also satisfies *QExp*, and return *False* otherwise.

A non-boolean query $\mathcal{Q}_{nb}$ refers to a formula of the form

$$\mathcal{Q}_{nb} \leftarrow QExp,$$

where *QExp* is a concept expression, e.g.

$$\mathcal{Q}_{nb} \leftarrow Parent \sqcap \exists hasChild.Employee$$

In this case the query will return one of the member of the set $\{\bot, \mathcal{M}\}$, where $\bot$ refers to the empty set, and $\mathcal{M}$ represents a set of models $\{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$, where each of them satisfies *QExp* with respect to the knowledge base $\mathcal{K}$. The query will return $\mathcal{M}$ if and only if there exists at least one such model, otherwise return $\bot$.

A non-boolean query (retrieval) can be trivially transformed into a set of boolean queries for all candidate tuples, i.e., retrieving sets of tuples can be achieved by repeated application of boolean queries with different tuples of individuals substituted for variables. However, answering a boolean query is in fact an entailment problem. For example, answering the boolean query:

$$\mathcal{Q}_b \leftarrow Tom : (Parent \sqcap \exists hasChild.Employee),$$

is the problem of checking whether

$$\mathcal{K} \models Tom : (Parent \sqcap \exists hasChild.Employee).$$

In a DL (supporting full negation, e.g., $\mathcal{ALC}$), boolean query or instance checking can be reduced to knowledge base satisfiability test: $\mathcal{K} \models C(a)$ iff $\mathcal{K} \cup \{\neg C(a)\}$ is unsatisfiable.

(Donini & Massacci 2000) gave a tableaux algorithm for solving $\mathcal{ALC}$ satisfiability problem with respect to a Tbox. They proved that their algorithm has EXPTIME-complete worst-case complexity. To the best of our knowledge, this is the latest known result of complexity proof for the $\mathcal{ALC}$ satisfiability problem with respect to a Tbox. Nevertheless, the ontology language OWL, in particular OWL-DL, of the Semantic web technology is based on $\mathcal{SHOIQ}(\mathcal{D})$ which is even more expressive than $\mathcal{ALC}$. Since the query answering is in fact an instance checking (or a retrieval reasoning service) which can be reduced to a satisfiability problem. It is easy to verify that the existing DL reasoning services are still not enough to be used solely with the Semantic web technology. One way to mitigate the problem is to optimize the algorithm more and more. We propose an optimization technique for answering a query over a description logic knowledge base. This technique is coherent with nature of the Web in that it supports multiple-Abox environment, which is corresponding to multiple data source environment in the Web.

4 The Approach

This contribution focuses on finding an answer to the question: “How can we (efficiently) answer a query in $\mathcal{ALC}$ based-Semantic Web system, given single ontology $\mathcal{T}$, and multiple data sources $\mathcal{A}$s, examining the minimum number of data sources?”. We refer to an Abox as a data source.

The idea of this section is based on the observation that $2^m > 2^n + 2^p$, where $m = n + p$ for $n, p > 1$. This means that if we can split the search space into independent parts, query the parts independently from each other, and combine the answers, then we have a considerable improvement of the performance of the query system. This idea agrees with the understanding of the Semantic Web as a collection of sometime “unrelated” data sources. In addition we propose to attach to each data source a data source description (or source description), a compact representation of the content of the data page. This idea is similar to the intuition behind indexes in databases. In the same way that type of indexes is more or less appropriate for particular queries, source descriptions depend on the type of queries. On the other hand, as we will see in the rest of this section, the relationships among data sources are not influenced by queries. They are determined by the structure of the data itself.

In this approach, the knowledge base architecture is shown in the following figure:

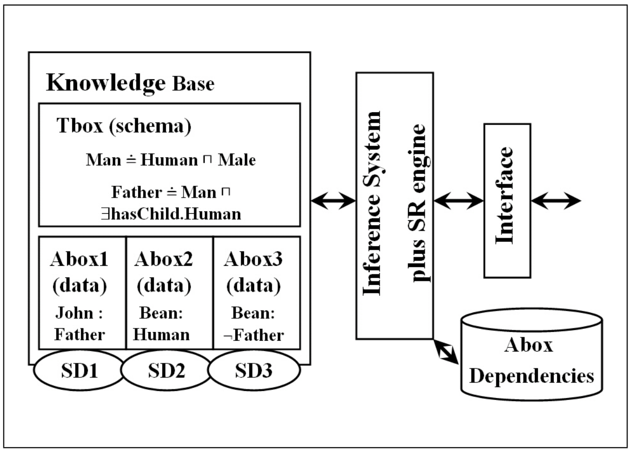


Figure 2: The Knowledge Base Architecture

The intuition here is to associate to every Abox $\mathcal{A}$ a source description $SD(\mathcal{A})$, and to supplement the inference engine with information about the mutual dependencies of the Aboxes in the system, in order to determine which Aboxes are relevant and must be queried.

The first step is to associate to every Abox its domain.

Definition 8 *Given an Abox $\mathcal{A}$, let $H_{\mathcal{A}}$ be the Herbrand universe of $\mathcal{A}$ (i.e., the set of all the individual occurring in expression in $\mathcal{A}$). For any interpretation $\mathcal{I}$, $\Delta^{\mathcal{I}}_{\mathcal{A}}$, the domain of $\mathcal{A}$ is defined as follows:*

$$\Delta^{\mathcal{I}}_{\mathcal{A}} = \left\{ d \in \Delta^{\mathcal{I}} | a \in H_{\mathcal{A}} \wedge a^{\mathcal{I}} = d \right\}.$$

Definition 9 (Multiple Assertional Knowledge Base) *Given a set $\bar{\mathcal{A}}$ of Aboxes $\mathcal{A}_1, \ldots, \mathcal{A}_k$, i.e., $\bar{\mathcal{A}} = \{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ and a Tbox $\mathcal{T}$, the* multiple assertional knowledge base *is the knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{A}$ is the Abox obtained from the union of all the Aboxes in $\bar{\mathcal{A}}$, i.e., $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \ldots \cup \mathcal{A}_k$.*

A consequence of the above definition is that the interpretation domain of $\mathcal{A}$ is equivalent to the union of interpretation domains of the $\mathcal{A}_j$s ($\Delta^{\mathcal{I}}_{\mathcal{A}} = \bigcup_{1 \leq j \leq k} \Delta^{\mathcal{I}}_{\mathcal{A}_j}$). Since $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ by definition, thus, for arbitrary C, $C^{\mathcal{I}}_{\mathcal{A}} = \bigcup C^{\mathcal{I}}_{\mathcal{A}_j}$, for $j \in \{1, \ldots, k\}$ and $C_{\mathcal{A}_j}$ is the concept C that occurs in a concept membership assertion in $\mathcal{A}_j$.

We approach the problem in 5 steps:

1. Determine dependencies among data sources, and group data sources which are dependent on each other together.
2. Associate each data source (or group of data sources) with a source description.
3. When one queries the knowledge base, exploit a procedure to find irrelevant data sources (or groups of data sources) with respect to the query, taking into account source descriptions and the query. Eliminate the irrelevant data sources (or groups of data sources) from query answering process, yielding a set of possible relevant data sources (or groups of data sources) to be queried.
4. For each remaining data source (or group of data sources) from the previous step, apply the existing query answering procedure to each of them, yielding answer from each of them.
5. Simply combine answers from the queried data sources (or groups of data sources) together, since each data source (or group of data sources) is independent with the other.

Since a reasoning procedure for simple query answering in the fourth step exists (Tessaris 2001), we will focus on other steps, which are in fact the steps of the data source space partitioning and reduction using source description.

The approach can be implemented by the following algorithm.

```
Algorithm 1 $partitioned\_QA(query, \mathcal{A})$:
depset = {}
answer = False
for all $\mathcal{A}_m \in \bar{\mathcal{A}}$ do
    $AG(\mathcal{A}_m)$ = create_abox_graph($\mathcal{A}_m$)
end for
for all 2-combinations $\{\mathcal{A}_i, \mathcal{A}_j\}$ of $\mathcal{A}$ do
    if find_abox_dependency($AG(\mathcal{A}_i)$,$AG(\mathcal{A}_j)$) = True
        then add dep($\mathcal{A}_i, \mathcal{A}_j$) to depset
    end if
end for
$\mathcal{A}^g$ = combine_dependent_abox($\mathcal{A}$,depset)
create_source_description($\mathcal{A}^g$)
for all $\mathcal{A}_h \in \mathcal{A}^g$ do
    if query_relevancy($SD(\mathcal{A}_h)$,$query$) = True then
        answer = answer $\vee$ instance_checking($\mathcal{A}_h$,$query$)
    end if
end for
```

First, since an Abox $\mathcal{A}_i$ can overlap with another Abox $\mathcal{A}_j$, we must consider multiple Aboxes at the same time. However, we will not treat all of the Aboxes as a single Abox, because, in this case, the associated reasoning is computational expensive. Consequently, we need some additional procedure to determine dependencies among Aboxes since we need to know which Aboxes should be considered together. In other word, we need to group dependent Aboxes together and treat them as a new single Abox consisting of multiple dependent Aboxes. To make this clear, we need to formally define the dependency between Aboxes in the context of Abox reasoning.

Firstly, we will introduce graph notation for an Abox.

Definition 10 (Abox Graph) *An Abox graph for an Abox $\mathcal{A}$, $AG(\mathcal{A})$, consists of a set N of nodes (vertexes), a set E of edges (arcs), and a function f from E to $\{(a,b) \mid a,b \in N\}$. Each edge, label ed R_i, represents exactly a role name of a role membership assertion $R_i(a,b) \in \mathcal{A}$. Hence, each* *node represents exactly one individual name. An Abox graph is a directed multigraph.*

The create_abox_graph function will produce an Abox graph $AG(\mathcal{A}_m)$ for each Abox $\mathcal{A}_m$. We will say that an Abox $\mathcal{A}$ is *connected* if its Abox graph $AG(\mathcal{A})$ is weakly connected.

Definition 11 (Abox Dependency) *Given two connected Aboxes $\mathcal{A}_1$ and $\mathcal{A}_2$, where $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$; $\mathcal{A}_1$ and $\mathcal{A}_2$ depend on each other if the graph of Abox $\mathcal{A}$ is (weakly) connected, and independent otherwise.*

Proposition 12 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two independent Aboxes in multiple assertional knowledge base. Let $\Delta^{\mathcal{I}}_{\mathcal{A}_1}$ and $\Delta^{\mathcal{I}}_{\mathcal{A}_2}$ be the domains of $\mathcal{A}_1$ and $\mathcal{A}_2$, then:*

- $\Delta^{\mathcal{I}}_{\mathcal{A}_1} \cap \Delta^{\mathcal{I}}_{\mathcal{A}_2} = \emptyset$*;*
- *for any concept C, $C^{\mathcal{I}}_{\mathcal{A}_1} \cap C^{\mathcal{I}}_{\mathcal{A}_2} = \emptyset$, where $C^{\mathcal{I}}_{\mathcal{A}_i}$ is the extension of C in $\Delta^{\mathcal{I}}_{\mathcal{A}_i}$.*

If $\mathcal{A}$ is unconnected, i.e., $\mathcal{A}_1$ and $\mathcal{A}_2$ are independent on each other, then it means that $\mathcal{A}_1$ and $\mathcal{A}_2$ do not share any common node (individual) because Aboxes $\mathcal{A}_1$ and $\mathcal{A}_2$ are already connected by themselves. Thus, we can use Abox graphs to determine Abox dependency.

For any unordered pair of Aboxes $\{\mathcal{A}_i, \mathcal{A}_j\}$, we determine the Abox dependency between the two Aboxes ($\mathcal{A}_i$ and $\mathcal{A}_j$). According to the definition, Abox dependency can be detected using the connectivity of the Abox graph of $\mathcal{A}$, i.e., $AG(\mathcal{A})$, where $\mathcal{A} = \mathcal{A}_i \cup \mathcal{A}_j$. Thus, we can exploit any UCONN (undirected graph connectivity problem) algorithm for this purpose. The function find_abox_dependency($AG(\mathcal{A}_i)$,$AG(\mathcal{A}_j)$) returns True if two Aboxes $\mathcal{A}_i$ and $\mathcal{A}_j$ depend on each other, and False otherwise. If the function returns True, then we add dep($\mathcal{A}_i, \mathcal{A}_j$) to the set "depset", i.e., the set that stores dependency value of each pair of Aboxes. Then we virtually combine dependent Aboxes together as a group by the function combine_dependent_abox($\mathcal{A}$,depset). The Abox $\bar{\mathcal{A}}$ will become $\mathcal{A}^g$, i.e., the set of already-grouped Aboxes and ungrouped Aboxes. Each Abox in $\mathcal{A}^g$ is independent of each other.

Next, we need to show two things:

1. if two Aboxes depend on each other, then a DL reasoning service, in particular instance checking and retrieval, needs to take into account the two Aboxes together;
2. if two Aboxes are independent of each other, then a DL reasoning over the two Aboxes can be done separately over each of them.

The following theorem supports the last step in our approach. It provides the reason why we can simply combine the answer from each $\mathcal{A}_i \in \mathcal{A}^g$ together. In other words it states that the the instance checking (a query answering) problem over $\mathcal{A}^g$ can be reduced to separate instance checking problems over each $\mathcal{A}_i$.

Theorem 13 (Independent Abox for Boolean Query) *Given two connected Aboxes $\mathcal{A}_1$ and $\mathcal{A}_2$, where $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, If $\mathcal{A}_1$ and $\mathcal{A}_2$ are independent on each other, then for any boolean query $\mathcal{Q}$ and Tbox $\mathcal{T}$, $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathcal{Q}$ if and only if $\langle \mathcal{T}, \mathcal{A}_1 \rangle \models \mathcal{Q}$ or $\langle \mathcal{T}, \mathcal{A}_2 \rangle \models \mathcal{Q}$.*

Proof First, we prove the only if direction, and we will assume that both $\mathcal{A}_1$ and $\mathcal{A}_2$ are consistent with $\mathcal{K}$, since if one of them is not then the theorem trivially holds.

Since $\mathcal{A}_1$ and $\mathcal{A}_2$ are *independent* on each other, by Proposition 12, we have $\Delta^{\mathcal{I}}_1 \cap \Delta^{\mathcal{I}}_2 = \emptyset$, where $\Delta^{\mathcal{I}}_1$ and $\Delta^{\mathcal{I}}_2$ are the domains of $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively.

Suppose $\langle \mathcal{T}, \mathcal{A}_1 \rangle \not\models \mathcal{Q}$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle \not\models \mathcal{Q}$. These mean $\exists \mathcal{I}_1$ such that $\mathcal{I}_1 \models \mathcal{A}_1$, $\mathcal{I}_1 \models \mathcal{T}$, $\mathcal{I}_1 \models \neg\mathcal{Q}$, and $\exists \mathcal{I}_2$ such that $\mathcal{I}_2 \models \mathcal{A}_2$, $\mathcal{I}_2 \models \mathcal{T}$ and $\mathcal{I}_2 \models \neg\mathcal{Q}$. Note that $\mathcal{I}_1$ and $\mathcal{I}_2$ are arbitrary interpretations of $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively with the only constraint of being interpretations of $\mathcal{T}$.

Since $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ and $\Delta^{\mathcal{I}}_1 \cap \Delta^{\mathcal{I}}_2 = \emptyset$, we can create an interpretation $\mathcal{I}$ of $\mathcal{A}$ such that $\mathcal{I}$ is the union of the interpretation $\mathcal{I}_1$ of $\mathcal{A}_1$ and the interpretation $\mathcal{I}_2$ of $\mathcal{A}_2$ ($\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2$). More precisely, $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \bullet^{\mathcal{I}} \rangle$ is defined as follows:

(i) $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}}_1 \cup \Delta^{\mathcal{I}}_2$ because $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, where $\Delta^{\mathcal{I}}$ is the domains of $\mathcal{A}$

(ii) For any constant a,

$$a^{\mathcal{I}} = \begin{cases} a^{\mathcal{I}_1} & \text{if } a \text{ occurs in } \mathcal{A}_1 \\ a^{\mathcal{I}_2} & \text{if } a \text{ occurs in } \mathcal{A}_2 \end{cases}$$

(iii) For any concept C, $C^{\mathcal{I}} = C^{\mathcal{I}_1} \cup C^{\mathcal{I}_2}$

(iv) For any role R, $R^{\mathcal{I}} = R^{\mathcal{I}_1} \cup R^{\mathcal{I}_2}$

Since $\Delta^{\mathcal{I}}_1 \cap \Delta^{\mathcal{I}}_2 = \emptyset$, then it is immediate to verify that $\mathcal{I}$ is indeed an interpretation, and $\mathcal{I} \models \mathcal{T}$, since $\mathcal{I}_1 \models \mathcal{T}$ and $\mathcal{I}_2 \models \mathcal{T}$.

Since $\mathcal{I}_1 \models \neg\mathcal{Q}$ and $\mathcal{I}_2 \models \neg\mathcal{Q}$, from (iii), we can immediately verify $\mathcal{I} \models \neg\mathcal{Q}$, i.e., $(\neg\mathcal{Q})^{\mathcal{I}} = (\neg\mathcal{Q})^{\mathcal{I}_1} \cup (\neg\mathcal{Q})^{\mathcal{I}_2}$, where $\mathcal{I}$ is the interpretation of $\mathcal{A}$. From (ii), (iii) and (iv), we can also infer that $(\mathcal{A})^{\mathcal{I}} = (\mathcal{A}_1)^{\mathcal{I}_1} \cup (\mathcal{A}_2)^{\mathcal{I}_2}$, i.e., $\mathcal{I} \models \mathcal{A}$.

Since $\mathcal{A}_1$ and $\mathcal{A}_2$ are assumed to be consistent by themselves, we only need to prove that there is no clash between $\mathcal{A}_1$ and $\mathcal{A}_2$. For an arbitrary concept C, by general definition in description logic, we get $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. In addition, we get $(\neg C)^{\mathcal{I}} = (\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}) \subseteq \Delta^{\mathcal{I}}$. Thus, for arbitrary C, $C^{\mathcal{I}_1} \subseteq \Delta^{\mathcal{I}}_1$ and $(\neg C)^{\mathcal{I}_2} \subseteq \Delta^{\mathcal{I}}_2$. Since $\Delta^{\mathcal{I}}_1 \cap \Delta^{\mathcal{I}}_2 = \emptyset$, therefore, $C^{\mathcal{I}_1} \cap (\neg C)^{\mathcal{I}_2} = \emptyset$, which infers that no clash can occur between $\mathcal{A}_1$ and $\mathcal{A}_2$.

Thus for the interpretation $\mathcal{I}$ of $\mathcal{A}$, we have $(\mathcal{A})^{\mathcal{I}} \neq \emptyset$ and $(\neg\mathcal{Q})^{\mathcal{I}} \neq \emptyset$, i.e., $\mathcal{I} \models \mathcal{A} \wedge \mathcal{I} \models \neg\mathcal{Q}$ which is the definition of $\mathcal{A} \not\models \mathcal{Q}$. Therefore, $\mathcal{A} \models \mathcal{Q}$ only if $\mathcal{A}_1 \models \mathcal{Q}$ or $\mathcal{A}_2 \models \mathcal{Q}$ which infers $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathcal{Q}$ only if $\langle \mathcal{T}, \mathcal{A}_1 \rangle \models \mathcal{Q}$ or $\langle \mathcal{T}, \mathcal{A}_2 \rangle \models \mathcal{Q}$.

For the if direction, we assume that either 1) $\langle \mathcal{T}, \mathcal{A}_1 \rangle \models \mathcal{Q}$ or 2) $\langle \mathcal{T}, \mathcal{A}_2 \rangle \models \mathcal{Q}$. In both cases, by monotonicity, we obtain $\langle \mathcal{T}, \mathcal{A}_1 \cup \mathcal{A}_2 \rangle \models \mathcal{Q}$ which is $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathcal{Q}$. □

In the second step of the approach, we associate each Abox (or group of Aboxes) with a source description, using create_source_description($\mathcal{A}^g$). A source description can be view as a surrogate of each data source. Surrogate refers to a brief representation of an information source that is designed to convey an indication of the information source's intent (Goodchild 1998). A good surrogate has two major properties: (1) it corresponds to some common understanding in the user's community, and (2) it can be organized in a way that is searchable.

Source descriptions are used to determine the relevancy of each Abox $\mathcal{A}_h \in \mathcal{A}^g$ with respect to a query. Source descriptions depend on the type of the query. For boolean queries, the source description of each Abox $\mathcal{A}_h \in \mathcal{A}^g$ can be a simple list of all individuals appearing in the Abox $\mathcal{A}_h$. The idea is if the query does not satisfy $SD(\mathcal{A}_h)$ (necessary and sufficient conditions), it is guaranteed that the query over Abox $\mathcal{A}_h$ will fail, i.e., it returns False. This is done by the function query_relevancy($SD(\mathcal{A}_h)$,*query*). This function returns False if the query does not satisfy $SD(\mathcal{A}_h)$, i.e., the Abox $\mathcal{A}_h$ is fully irrelevant to the query, and will contribute nothing to the answer of the query. The function works by extracting an individual from the query, then checking if it is in the source description $SD(\mathcal{A}_h)$ or not. If it is, then it queries the Abox

$\mathcal{A}_h$, using normal boolean query answering procedure instance_checking($\mathcal{A}_h$,*query*).

This can be formalised as follows:

Definition 14 *Let $\mathcal{A}$ be an Abox, the boolean query source description for $\mathcal{A}$ ($SD_b(\mathcal{A})$) is the the Herbrand universe of $\mathcal{A}$, i.e., $SD_b(\mathcal{A}) = H_{\mathcal{A}}$.*

We can now prove soundness and completeness of the proposed algorithm.

Theorem 15 (Soundness and Completeness) *Let $\mathcal{Q}$ be a boolean query. Let $\mathcal{A}\ /\!\sharp\mathcal{Q}$ represents when query_relevance($SD_b(\mathcal{A}),\mathcal{Q}$) returns False, i.e., $\mathcal{A}$ is not relevant the query $\mathcal{Q}$, and let $\mathcal{A}\sharp\mathcal{Q}$ represents otherwise. If $\mathcal{A} \not\sharp \mathcal{Q}$ then $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$ if and only if $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$.*

Proof Suppose $\mathcal{A} \not\sharp \mathcal{Q}$. This means $a \notin SD_B(\mathcal{A})$, where $\mathcal{Q}$ is $C(a)$.

First, we prove the only if direction. Suppose $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$. However, $\bar{\mathcal{A}}-\{\mathcal{A}\} \subseteq \bar{\mathcal{A}}$. By monotonicity, we obtain $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$.

Therefore, $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$ only if $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$.

For the if direction, suppose $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$. We, then, prove by case.

Case 1: $\mathcal{Q}$ is a tautology. It is immediate to verify that $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$ is true.

Case 2: $\mathcal{Q}$ is not a tautology. From Lemma 16, we obtain $\langle\mathcal{T},\mathcal{A}\rangle \not\models \mathcal{Q}$. In addition, $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$ is equal to $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\cup\{\mathcal{A}\}\rangle \models \mathcal{Q}$. By Theorem 13, we get $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$ or $\langle\mathcal{T},\mathcal{A}\rangle \models \mathcal{Q}$. Since $\langle\mathcal{T},\mathcal{A}\rangle \not\models \mathcal{Q}$, we obtain $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$.

These cases cover all possibilities. Therefore, $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$ if $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$.

Therefore, if $\mathcal{A} \not\sharp \mathcal{Q}$ then $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$ if and only if $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$. □

Lemma 16 *Let $\mathcal{Q}$ be a boolean query $C(a)$. If $a \notin SD_B(\mathcal{A})$ and $\mathcal{Q}$ is not a tautology, then $\langle\mathcal{T},\mathcal{A}\rangle \not\models \mathcal{Q}$.*

Proof Suppose $a \notin SD_B(\mathcal{A})$. By definition, this means $a \notin \Delta^{\mathcal{I}}_{\mathcal{A}}$. In other words, there is no a in $\mathcal{A}$, i.e., $C(a)$ is definitely not in $\mathcal{A}$. Suppppose $\mathcal{Q}$ is not a tautology. At this state, we want to prove $\langle\mathcal{T},\mathcal{A}\rangle \not\models C(a)$ which is equal to proving that $\langle\mathcal{T},\mathcal{A}\rangle \cup \{\neg C(a)\}$ is satisfiable. Since $\langle\mathcal{T},\mathcal{A}\rangle$ is consistent, $\langle\mathcal{T},\mathcal{A}\rangle \cup \{\neg C(a)\}$ will be unsatisfiable only if either $C(a)$ is in $\mathcal{A}$ or $C(a)$ is a tautology. Since neither of them is true, $\langle\mathcal{T},\mathcal{A}\rangle \cup \{\neg C(a)\}$ is satisfiable. Consequently, we prove $\langle\mathcal{T},\mathcal{A}\rangle \not\models C(a)$.

Therefore, if $a \notin SD_B(\mathcal{A})$ and $\mathcal{Q}$ is not a tautology, then $\langle\mathcal{T},\mathcal{A}\rangle \not\models \mathcal{Q}$. □

Finally, in the last step we simply combine the answers together using disjunction. Again this step is justified by Theorem 13.

5 Example for the Approach

Suppose we have a knowledge bases $\mathcal{K} = \langle\mathcal{T},\mathcal{A}\rangle$, where the data is distributed over four Aboxes, i.e., $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4$.

Suppose the Tbox $\mathcal{T}$ is as follows:

$$\begin{aligned}\mathcal{T} = \{&Male \doteq \neg Female,\\ &Man \doteq Human \sqcap Male,\\ &Woman \doteq Human \sqcap Female,\\ &Father \doteq Man \sqcap \exists hasChild.Human,\\ &Mother \doteq Woman \sqcap \exists hasChild.Human,\\ &Parent \doteq Father \sqcup Mother,\\ &Employee \doteq Human \sqcap \exists workFor.Org,\\ &Org \doteq ProfitableOrg \sqcup CharityOrg,\\ &ProfitableOrg \doteq \neg CharityOrg,\\ &ProfitableOrg \doteq Company \sqcup Partnership \sqcup SoleProprietorship\\ &NewspaperCompany \doteq Company \sqcap \exists publish.Newspaper\}\end{aligned}$$

Suppose that the four Aboxes are as follows:

$$\begin{aligned}\mathcal{A}_1 = \{&Man(John), Man(Clark), Woman(Chloe),\\ &SoleProprietorship(BKKDelight), hasChild(John,Clark),\\ &workFor(Clark,DailyPlanet), attend(Clark,MetroNews05),\\ &hasChild(John,Chloe), workFor(Chloe,BKKDelight),\\ &enrolAt(Chloe,DKE)\}\\ \mathcal{A}_2 = \{&ResearchGroup(DKE), School(ENG), University(UQ),\\ &Workshop(MetroNews05), partOf(DKE,ENG),\\ &facultyOf(ENG,UQ), locatedIn(UQ,Australia)\}\\ \mathcal{A}_3 = \{&NewspaperCompany(DailyPlanet), Workshop(MetroNews05),\\ &NewspaperCompany(Inquisitor), locatedIn(DailyPlanet,USA),\\ &sponsoredBy(MetroNews05,DailyPlanet),\\ &sponsoredBy(MetroNews05,Inquisitor),\\ &sponsoredBy(MetroNews05,DKE)\}\\ \mathcal{A}_4 = \{&CharityOrg(WorldHelp), CharityProgram(TsuHelp),\\ &CharityProgram(QuakeHelp),\\ &locatedIn(WorldHelp,Germany),\\ &create(WorldHelp,TsuHelp), create(WorldHelp,QuakeHelp)\}\end{aligned}$$

We simply create the Abox graph for each Abox, yielding four Abox graphs: $AG(\mathcal{A}_1)$, $AG(\mathcal{A}_2)$, $AG(\mathcal{A}_3)$, and $AG(\mathcal{A}_4)$. For every combination of two Aboxes of $\mathcal{A}$, we determine the Abox dependency between them using the find_abox_dependency function. The function will combine the two Aboxes together, and apply UCONN algorithm to the graph of the combined Abox.

If the graph is connected, then the two Aboxes depend on each other. We, then, add dep($\mathcal{A}_i,\mathcal{A}_j$) to the set depset as they are dependent, i.e., the function returns True. We get

$$\text{depset} = \{\text{dep}(\mathcal{A}_1,\mathcal{A}_2), \text{dep}(\mathcal{A}_1,\mathcal{A}_3), \text{dep}(\mathcal{A}_2,\mathcal{A}_3)\}$$

Since we know that $\mathcal{A}_1$ depends on $\mathcal{A}_2$, and also on $\mathcal{A}_3$, we virtually group them together, i.e., we will consider the data in both three Aboxes together. This is done by the combine_dependent_abox($\mathcal{A}$,depset) function. As a result we have $\mathcal{A}^g = \{\mathcal{A}_{123},\mathcal{A}_4\}$, where $\mathcal{A}_{123} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$.

At this stage, we create source description for each Abox in $\mathcal{A}^g$, using the create_source_description($\mathcal{A}^g$) procedure:

- $SD(\mathcal{A}_{123}) = (\ DailyPlanet, Australia, Inquisitor, ENG, DKE, Chloe, MetroNews05, Clark, USA, BKKDelight, John, UQ)$
- $SD(\mathcal{A}_4) = (\ Malaysia, TsuHelp, WorldHelp, QuakeHelp)$

Suppose, we have a boolean query $John : (Parent \sqcap \exists hasChild.Employee)$. For every $\mathcal{A}_h \in \mathcal{A}^g$, we determine relevancy with respect to the query, using the procedure query_relevancy($SD(\mathcal{A}_h)$,*query*). The procedure will extract "John" from the query, and search whether "John" is in $SD(\mathcal{A}_h)$ or not. In this case, "John" is in $SD(\mathcal{A}_{123})$, but not in $SD(\mathcal{A}_4)$. Consequently, we simply query $\mathcal{A}_{123}$, using the instance_checking($\mathcal{A}_h$, *query*) function. The result from instance checking test of the query in $\mathcal{A}_{123}$ is True. Thus, answer = False $\vee$ True = True, which is the same result as when we query the whole Abox $\mathcal{A}$.

6 Complexity Analysis

Each Abox Graph can be trivially generated in $O(n^2)$, where n is the number of assertions in Abox $\mathcal{A}$. The next part is Abox dependency determination. We need $k(k-1)/2$ comparisons (2-combinations) of unordered pair of Aboxes, where k is the number of Aboxes in $\mathcal{A}$. Each comparison needs UCONN algorithm. UCONN can be solved by DFS in $O(v^2)$, where v is the number of individuals in each Abox.

For source description, the create_source_description($\mathcal{A}^g$) procedure requires not more than $O(n^2)$ for all Aboxes, since it can be implemented using the quick sort algorithm. To determine the relevancy of an Abox to a query, we call the function query_relevancy($SD(\mathcal{A}_h)$,*query*) that operates in $O(n)$ for sequential search. Finally, we simply use the instance_checking($\mathcal{A}_h$,*query*) function to find the answer for each Abox, and simply combine the answer.

Till now, our space partitioning and reduction approach exploits at most PTIME algorithms in each part, i.e., the Abox dependency part and the source description part. Overall, our algorithm can be operated in PTIME, not including the instance checking part. Since the instance checking part is known to be solved in EXPTIME-complete, assuming P $\neq$ EXPTIME, the overall algorithm still operates in EXPTIME, but with a *reduced exponent*. The Abox dependency part will partition the search space, thus the exponent will be reduced if there are at least two partitions, e.g., the time complexity is reduced from 2^m to $2^n + 2^p$, where $m = n + p$. The source description part will further reduce the exponent if there are some Aboxes which can be eliminated from the process, e.g., the time complexity is reduced from 2^m to 2^q, where $q < m$.

7 The Extension

We intend to extend our work to non-boolean query answering (Abox retrieval) optimization. The space partitioning, Theorem 13, can be easily extended to cover Abox retrieval reasoning services. The main differences will be with the source description. Furthermore, we will investigate possible extensions of Theorem 13 to more expressive DLs.

In the nutshell, a source description of an Abox $\mathcal{A}$ for retrieval ($SD(\mathcal{A})$) consists of several types of source descriptions. In the basic setting, we shall have $SD^C(\mathcal{A})$ and $SD^R(\mathcal{A})$. $SD^C(\mathcal{A})$ is a set of *least common subsumer*s or *LCS*s of concepts appeared in concept membership assertions in the Abox $\mathcal{A}$. $SD^R(\mathcal{A})$ is a set of roles appeared in role membership assertions in the Abox $\mathcal{A}$. Let Q be a non-boolean query, C, D be concept expressions, and $\mathcal{A}_h$ be an Abox. The source description usage is different for each form of Q, for example:

- For the case $Q \leftarrow C$, $\mathcal{A}_h$ is relevant to Q if $\exists s \in SD^C(\mathcal{A}_h), Q \sqsubseteq s$
- For the case $Q \leftarrow C \sqcap D$, $\mathcal{A}_h$ is relevant to Q if $\mathcal{A}_h$ is relevant to C *and* $\mathcal{A}_h$ is relevant to D.
- For the case $Q \leftarrow C \sqcup D$, $\mathcal{A}_h$ is relevant to Q if $\mathcal{A}_h$ is relevant to C *or* $\mathcal{A}_h$ is relevant to D.
- For the case $Q \leftarrow \exists R.C$, $\mathcal{A}_h$ is relevant to Q if $R \in SD^R(\mathcal{A}_h)$ *and* $\mathcal{A}_h$ is relevant to C.
- For the case $Q \leftarrow \neg C$, $\mathcal{A}_h$ is relevant to Q if $\mathcal{A}_h$ is irrelevant to C.

These operations can be recursive. Hence, we can achieve a methodology for determine relevancy of the Abox $\mathcal{A}_h$ with respect to the arbitrary-formed non-boolean query Q. Note that the correctness of the above methodology follows from the independent Abox theorem, extended for non-boolean query.

8 Summary and Discussion

The optimization approach presented in this work is based on the procedure normally adopted for deduction over a description logic knowledge base (the query answering process over such knowledge base is a deduction process). In particular we refer to the tableaux algorithm. Traditionally, tableaux algorithm was designed to prove the satisfiability problem. The main idea behind this algorithm is based on a notational variant of the first order tableaux calculus. In fact, a tableaux algorithm tries to prove the satisfiability of a concept expression C by demonstrating a nonempty model of C. It constructively builds a model for a given concept. The process of constructing a model proceeds by completing a constraint system (Tessaris 2001), using a set of consistency-preserving completion (or expansion) rules. The process will continue if it can extend the existing constraint system. In $\mathcal{ALC}$ reasoning with $\mathcal{T}$ and $\mathcal{A}$, the process will proceed via a role membership assertion. The idea behind our work is to specify the condition where we guarantee that the reasoning process over $\mathcal{A}_1$ will never proceed to $\mathcal{A}_2$ if $\mathcal{A}_1$ and $\mathcal{A}_2$ are independent from each other. This optimization, in particular, the space partitioning part, can be seen as a divide-and-conquer technique. A general disadvantage of this kind of technique is the parts overlap. However, in this work we proposed a methodology to avoid overlapping part, thus, it does not suffer from such disadvantage of the divide-and-conquer technique.

Apparently, the major drawback of this approach is obviously the additional cost from Abox dependencies and source descriptions determination. But the cost is still in PTIME, as shown in previous section. One may argue that our space partitioning approach would support the reduction in apparent worst-case complexity for query answering, but at a high price in practice, in particular for a large knowledge base. However, this is not a scholarly argument, because the larger the knowledge base is, the less the relative cost is (recall that the normal query answering is in EXPTIME while the additional cost is in PTIME). Thus, our approach should behave well in practice. The only issue that we must give additional attention to is a design of effective Abox dependency information distribution, minimizing information exchange between nodes in the network, where each node represents an Abox. In addition, if the data pages do not change frequently, then there is no need to recompute the dependency of the Abox. In addition data source can be organised in indexes for fast retrieval.

One may also argue that this work is based on $\mathcal{ALC}$ description logic which is weaker than OWL. We accept this argument since at first we want to consider OWL-DL. However, OWL-DL is based on $\mathcal{SHOIQ}(\mathcal{D})$ description logic which is, in fact, the extension of $\mathcal{ALC}$. In the history of description logic, the least expressive language was investigated first. Then, the result were extended to cover more expressive languages consecutively. Consequently, we consider $\mathcal{ALC}$ first. We will, then, investigate the extension to cover OWL, in particular, OWL-DL, later.

This approach can be applied to a system which allows only one ontology (or Tbox). Though the Semantic Web technology tends to exploit multiple ontologies. In ontology-based integration of information area (Wache, Vögele, Visser, Stuckenschmidt, Schuster, Neumann & Hübner 2001), we can divide the exploitation of ontology into 3 approaches: single-ontology approach, e.g., SIMS (Arens, Hsu & Knoblock 1996), multiple-ontologies approach, e.g., OBSERVER (Mena, Kashyap, Sheth & Illarramendi 1996), and hybrid-ontology approach, e.g., COIN (Goh 1997). The single ontology approach allows only one ontology in the system while multiple-ontologies approach allows many ontologies in the system. The multiple-ontologies approach requires additional mapping specifications between each pair of ontologies. Since such mappings are infact ontologies themselves (Akahani, Hiramatsu & Kogure 2002), we need additional $n(n-1)/2$ ontologies for such an approach, where n is number of existing ontologies in the system. In hybrid-ontology approach, a global ontology and additional n mapping specifications (between global ontology and each local ontology) are required. Hence the single-ontology approach can be viewed as generalization of the other two approaches. Thus, we follow such approach. In addition, since the aim of our work is to study how to query multiple data sources, thus we do not need to add complexity arisen from ontology mapping in the last two approaches. Simple single-ontology approach, but not trivial for query answering, is enough. Note that we can extend our work to include multiple ontologies later when the research about ontology binding and ontology mapping and ontology integration is more mature.

This approach can be applied to a system which allows multiple data sources (or Aboxes). We can think of an Abox as an RDF document. In addition RDF databases try to partition RDF triples in disjoint graphs , where each graph can be understood as a data page of our approach. However, recent research has shown that there are several semantic problems when people tried to layer an ontology language, i.e., OWL, on top of the RDF layer (Pan & Horrocks 2003). Such problems stem from some features/limitations of RDF, e.g., no restriction on the use of built-in vocabularies, and no restriction on how an RDF statement can be constructed since it is just a triple. Thus, this implies that the ontology layer may be not compatible with the RDF layer of the Semantic Web. However, there is a work proposing additional layer on top of the RDF layer, i.e., RDF(FA) (Pan & Horrocks 2003). This layer corresponds directly to Aboxes, thus RDF(FA) may be very useful in the future.

There are few works related to our work, i.e., Instance Store (Horrocks et al. 2004) and RACER (Haarslev & Möller 2002). Both works propose retrieval optimization techniques. Hence, our approach seems to be the first approach for Abox instance checking optimization. Instance Store imposes an unnatural restriction on Abox, i.e., enforcing Abox to be role-free. This is a severe restriction since role names are included even for $\mathcal{FL}^-$ ($\mathcal{ALC}$ without atomic negation), a DL with limited expressive power. RACER proposes several innovative Abox reasoning optimization techniques. However, RACER allows single Abox, while our approach allows multiple Aboxes. Thus after we apply our techniques to reduce the reasoning search space, we can apply RACER techniques to reduce it further. Consequently, the approach taken in RACER seems to be complementary to ours. We will investigate the combination of our approach and RACER approach in the future.

References

Akahani, J., Hiramatsu, K. & Kogure, K. (2002), Coordinating heterogeneous information services based on approximate ontology translation, *in* 'Proceedings of AAMAS-2002 Workshop on Agentcities: Challenges in Open Agent Systems', pp. 10–14.

Arens, Y., Hsu, C. & Knoblock, C. A. (1996), 'Query processing in the sims information mediator', *Advanced Planning Technology* .

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. & Patel-Schneider, P., eds (2003), *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, New York.

Baader, F. & Hollunder, B. (1991), A terminological knowledge representation system with complete inference algorithm, *in* 'The Workshop on Processing Declarative Knowledge', pp. 67–86.

Berners-Lee, T. (1999), *Weaving the Web : the Original Design and Ultimate Destiny of the World Wide Web by its Inventor*, HarperSanFrancisco, San Francisco.

Donini, F. M., Lenzerini, M., Nardi, D. & Nutt, W. (1991), The complexity of concept languages, *in* J. Allen, R. Fikes & E. Sandewall, eds, 'Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR-91)', Massachusetts, pp. 151–162.

Donini, F. M., Lenzerini, M., Nardi, D. & Schaerf, A. (1996), 'Reasoning in description logics', *Foundation of Knowledge Representation* pp. 191–236.

Donini, F. M. & Massacci, F. (2000), 'Exptime tableaux for $\mathcal{ALC}$', *Artificial Intelligence* **124**(1), 87–138.

Doyle, J. & Patil, R. (1991), 'Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services', *Artificial Intelligence Journal* **48**, 261–297.

Goh, C. H. (1997), Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources, PhD thesis, MIT.

Goodchild, A. (1998), Database Discovery in the Organizational Environment, PhD thesis, University of Queensland.

Haarslev, V. & Möller, R. (2002), Optimization strategies for instance retrieval, *in* 'Proceedings of the International Workshop on Description Logics (DL 2002)'.

Horrocks, I. (1997), Optimising Tableaux Decision Procedures for Description Logics, PhD thesis, University of Manchester.

Horrocks, I., Li, L., Turi, D. & Bechhofer, S. (2004), The instance store: Description logic reasoning with large numbers of individuals, *in* 'International Workshop on Description Logics (DL 2004)', pp. 31–40.

McGuinness, D. L., Fikes, R., Stein, L. A. & Hendler, J. (2003), Daml-ont: An ontology language for the semantic web, *in* D. Fensel, J. Hendler, H. Lieberman & W. Wahlster, eds, 'Spinning the Semantic Web: Bringing the World Wide Web to its Full Potential', MIT Press.

Mena, E., Kashyap, V., Sheth, A. & Illarramendi, A. (1996), Observer: An approach for query processing in global information systems based on interoperability between pre-existing ontologies, *in* 'Proceedings of the 1^{st} IFCIS: International Conference on Cooperative Information Systems (CoopIS '96)'.

Pan, J. Z. & Horrocks, I. (2003), Rdfs(fa): A dl-ised sub-language of rdfs, *in* 'Proceedings of the 2003 International Workshop on Description Logics (DL2003)'.

Schmidt-Schauß, M. & Smolka, G. (1991), 'Attributive concept descriptions with complements', *Artificial Intelligence* **48**(1), 1–26.

Stuckenschmidt, H. (2003), 'Query processing on the semantic web', *Künstliche Intelligenz* **17**.

Tessaris, S. (2001), Questions and Answers: Reasoning and Querying in Description Logic, PhD thesis, University of Manchester.

Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H. & Hübner, S. (2001), Ontology-based integration of information - a survey of existing approaches, *in* 'Proceedings of the IJCAI-01 Workshop: Ontologies and Information Sharing', pp. 108–117.

A Multi-step Strategy for Approximate Similarity Search in Image Databases

Paul W.H. Kwan
School of Mathematics, Statistics and Computer Science
University of New England
Armidale, NSW 2351, Australia
kwan@mcs.une.edu.au

Junbin Gao
School of Information Technology
Charles Sturt University
Bathurst, NSW 2795, Australia
jbgao@csu.edu.au

Abstract

Many strategies for similarity search in image databases assume a metric and quadratic form-based similarity model where an optimal lower bounding distance function exists for filtering. These strategies are mainly two-step, with the initial "filter" step based on a spatial or metric access method followed by a "refine" step employing expensive computation. Recent research on robust matching methods for computer vision has discovered that similarity models behind human visual judgment are inherently non-metric. When applying such models to similarity search in image databases, one has to address the problem of non-metric distance functions that might not have an optimal lower bound for filtering. Here, we propose a novel three-step "prune-filter-refine" strategy for approximate similarity search on these models. First, the "prune" step adopts a spatial access method to roughly eliminate improbable matches via an adjustable distance threshold. Second, the "filter" step uses a quasi lower-bounding distance derived from the non-metric distance function of the similarity model. Third, the "refine" stage compares the query with the remaining candidates by a robust matching method for final ranking. Experimental results confirmed that the proposed strategy achieves more filtering than a two-step approach with close to no false drops in the final result.

Keywords: Multi-step Strategy, Similarity Search, Image Databases, Non-metric Distance, Lower Bound

1 Introduction

Researches on similarity search in image databases cover both data representation and query specifications. Images are usually represented as numerical vectors indexed by a spatial access method (Gaede and Günther 1998). In terms of query specifications, both range and nearest neighbour queries have been widely studied and analyzed (Weber, Schek and Blott 1998). In practice, a similarity query is often processed in two steps, namely the initial “filter” step employing a spatial index built on the distance function of the underlying similarity model, followed by a "refine" step performing exact but expensive distance calculations (Brinkhoff et al. 1994, Ankerst, Kriegel and Seidl 1998). However, most strategies assume a metric and quadratic form-based similarity model for which an optimal lower bounding distance function exists for filtering (Ciaccia and Patella 2002). Recent research on robust image matching methods for appearance-based vision has discovered that similarity models behind human visual judgment are often non-metric (Jacobs, Weinshall and Gdalyahu 2000). When applying such models, one has to address the problems of non-metric distance functions that might not have an optimal lower bound for filtering. Here, we proposed a novel three-step "prune-filter-refine" strategy for approximate similarity search for non-metric similarity models. First, the "prune" step adopts a spatial index to roughly eliminate improbable matches via an adjustable distance threshold. Second, the "filter" step uses a quasi lower-bounding distance derived from the non-metric distance function. Third, the "refine" stage compares the remaining candidates via a robust matching method for final ranking. Experimental evaluation confirmed that the proposed strategy achieves more filtering than a two-step approach with close to no false drops in the final result.

The rest of this paper is organized as follow. In Section 2, the proposed multi-step strategy is described. In Section 3, related works are mentioned. In Section 4, experimental evaluation performed on a database of traditional Japanese kamon images is presented. Lastly, Section 5 ends with concluding remarks.

2 The Multi-step Strategy

The proposed strategy consists of three successive steps, collectively known as the “prune-filter-refine” (or *PFR*) strategy. To facilitate explanation, Figure 1 illustrates the processing mechanisms behind the proposed strategy in the context of a shape-based image retrieval system that the authors developed based on a related research (Kwan, Kameyama and Toraichi 2003). This database application will serve as a guiding example throughout this paper.

2.1 The “Prune” Step

Similar to many two-step query processing strategies in the literature, the *prune* step of the proposed strategy uses a multi-dimensional spatial index to quickly eliminate irrelevant database objects from further matching. Each of the database objects is represented as a numerical vector whose values are derived from features of the image. Here, the set of numerical vectors is constructed by performing Discrete Fourier Transform (DFT) on both the horizontal and the vertical autocorrelation plots generated from each of the images by treating these plots as time sequence data.

Copyright © 2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

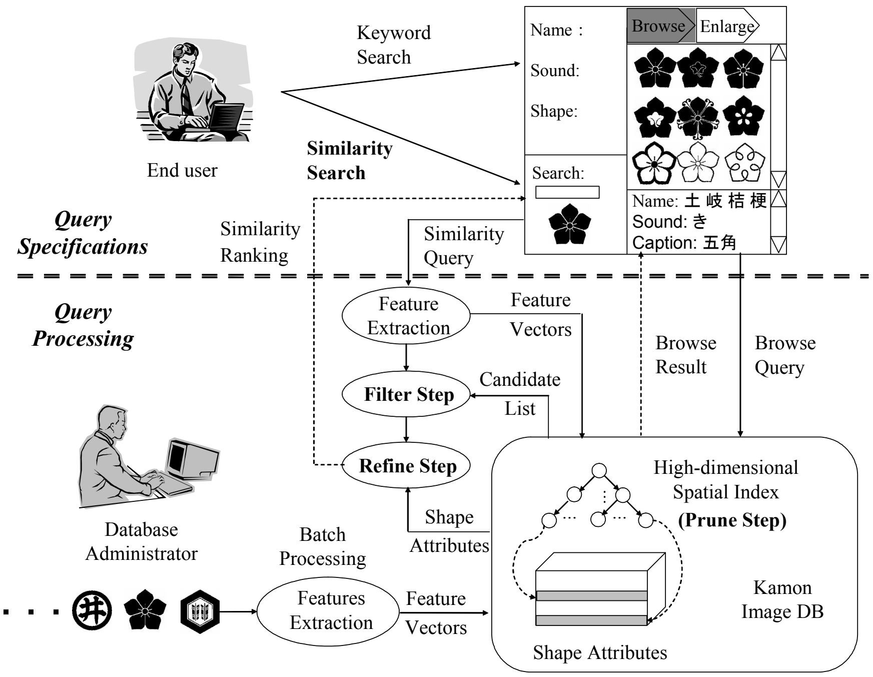


Figure 1: The Multi-step Strategy in relation to a Shape-based Image Retrieval System

The algorithm used in constructing these plots is similar to Nagashima, Tsubaki and Nakajima (2003). The idea is illustrated visually in Figure 2. Starting from a complete overlap of an image by a copy of itself, the horizontal autocorrelation is taken by shifting the image one pixel at a time from left to right while calculating for each shift the degree of autocorrelation, measured in terms of the number of non-background pixels that overlap. This is repeated until the image and its copy no longer overlap. Similar steps are taken when the vertical autocorrelation is measured by shifting from top to bottom instead.

More formally, let x and y be variables that denote the autocorrelation lengths measured in the horizontal and the vertical directions respectively. Then, the horizontal and vertical autocorrelations can be expressed as:

$$AC_h(x) = \frac{1}{l_Y}\sum_{Y=1}^{l_Y}\frac{1}{l_X - x}\sum_{X=1}^{l_X - x} f(X,Y)f(X+x,Y) \qquad (1)$$

$$AC_v(y) = \frac{1}{l_X}\sum_{X=1}^{l_X}\frac{1}{l_Y - y}\sum_{Y=1}^{l_Y - y} f(X,Y)f(X,Y+y) \qquad (2)$$

Here, f(·,·) is a function that returns 0 for a background pixel and a 1 for a non-background pixel. Based on these equations, the horizontal and vertical autocorrelation plots shown in Figures 2(c) and (d) can be generated.

Considering both horizontal and vertical autocorrelation plots as time sequences, the DFT generates for each a set of Fourier series coefficients as in Rafiei and Mendelzon (1997). Let a time sequence $\vec{x} = [x_t]$ for t = 0, 1, …, n-1 be a finite duration signal. The DFT of $\vec{x}$, denoted by $\vec{X} = [X_f]$, is given by:

$$X_f = \frac{1}{\sqrt{n}}\sum_{t=0}^{n-1} x_t e^{\frac{-j2\pi t f}{n}} \qquad f = 0,1,\ldots,n-1 \qquad (3)$$

Here, $j = \sqrt{-1}$ is the imaginary unit.

The inverse DFT of $\vec{X}$ returns the original time sequence by the following equation:

$$x_t = \frac{1}{\sqrt{n}}\sum_{f=0}^{n-1} X_f e^{\frac{j2\pi t f}{n}} \qquad t = 0,1,\ldots,n-1 \qquad (4)$$

For each database image, the two sets of coefficients are concatenated to produce a numerical vector. As illustrated in Table 1, for the experiments the first 15 Fourier series coefficients from each plot are used.

Horizontal Autocorrelation	Vertical Autocorrelation
163.334	212.053
28.528	49.417
-3.659	21.058
4.017	27.413
6.754	33.419
5.332	29.126
2.356	25.692
-1.210	21.002
-0.703	19.098
0.857	21.020
2.701	22.260
2.598	21.999
0.811	19.137
-0.557	15.825
-0.562	14.272

Table 1: The first 15 Fourier series coefficients of the autocorrelation plots shown in Figure 2(c) and (d)

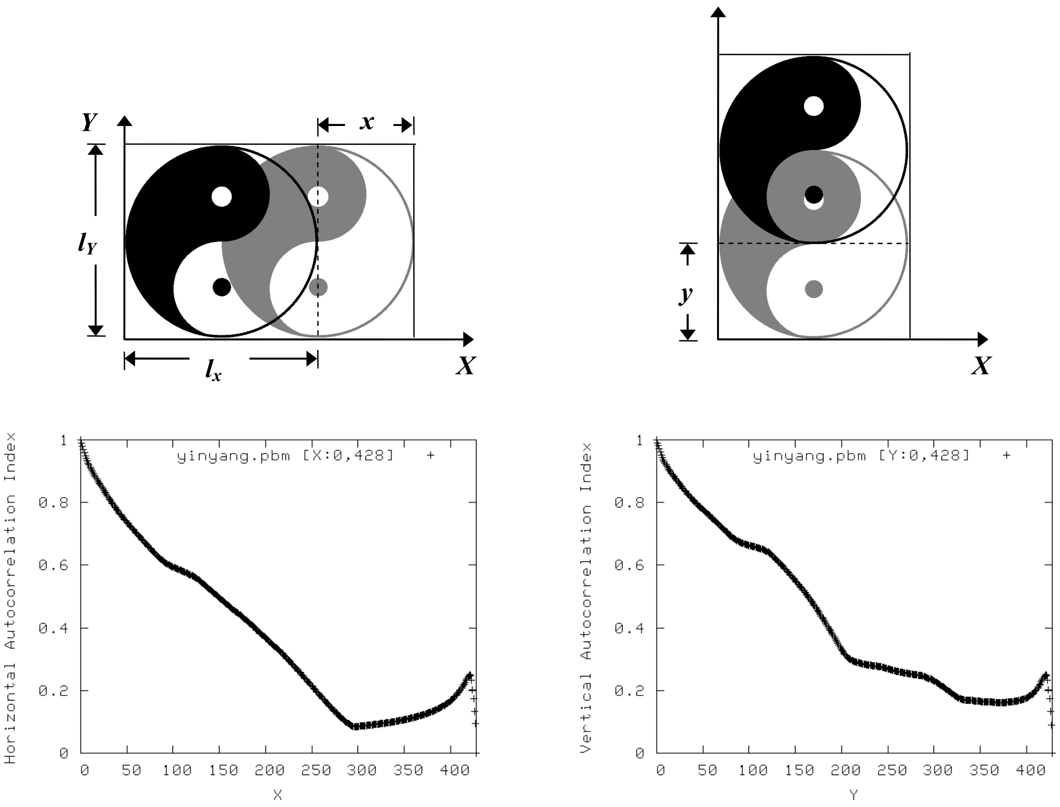


Figure 2: (a) and (b) illustrate how horizontal and vertical autocorrelations of an image is taken (c) and (d) are the generated horizontal and vertical autocorrelation plots respectively

To index the set of numerical vectors, a k-d-B tree based spatial index is chosen (Robinson 1981). The k-d-B tree combines the multi-dimensional search efficiency of k-d trees (Bentley and Friedman 1979) and the I/O efficiency of B-trees (Comer 1979) to handle multi-dimensional points. When applying the k-d-B tree in this work, the numerical vectors are *normalized* before embedded in a d-dimensional Euclidean space E^d. Let U be the universe of all such vectors. For any $O_i, O_j \in U$, their dissimilarity is defined by a distance metric $D(O_i, O_j) \in R^+$in E^d as:

$$D(O_i, O_j) = \sqrt{\left(O_i^1 - O_j^1\right)^2 + ... + \left(O_i^d - O_j^d\right)^2} \qquad (5)$$

Here, O_i^k and O_j^k denote the attribute values of O_i and O_j in the k^{th}-dimension respectively. Further, the condition below holds for the distance function:

$$0 < D(O_i, O_j) \leq \sqrt{d}\ , \ \ \forall O_i, O_j \in U \ \text{ and } \ i \neq j \qquad (6)$$

Based on the k-d-B tree based spatial index, two types of similarity queries, namely nearest neighbour and range queries could be possible. However, in the context of the proposed strategy, only range queries are relevant because the objective of the "prune" step is to reduce the search space quickly to a much smaller candidate set for further processing by the "filter and refine" steps. Given $O_q \in U$ as the query, by specifying $r \in R^+$ as the range, such that $0 \leq r \leq \sqrt{d}$ holds, the candidate set (denoted C) that meets the condition below is returned:

$$C = \left\{ O_p \middle| \ O_p \in U \ \ and \ \ D(O_p, O_q) \leq r \right\} \qquad (7)$$

Lastly, as seen from Figure 1, a *pointer* (or *ID*) is appended as the last element in each numerical vector to facilitate retrieving shape attributes of the corresponding image that will be used in the filter and refine steps.

2.2 The "Filter and Refine" Steps

The "filter and refine" steps of the proposed strategy are an application of Kwan et al. (2003). There, an *approximate query processing* approach for addressing the performance problem brought about by sequential matching in a related research was introduced (Kwan, Kameyama and Toraichi 2003). Kwan, Kameyama and Toraichi (2003) applied a robust image matching algorithm based on probabilistic relaxation labeling when comparing the query with every database image, and ranked them by a heuristic distance function. The features used were function approximated contour segments derived from closed contours of shapes extracted from the images. Figure 3 gives an example.

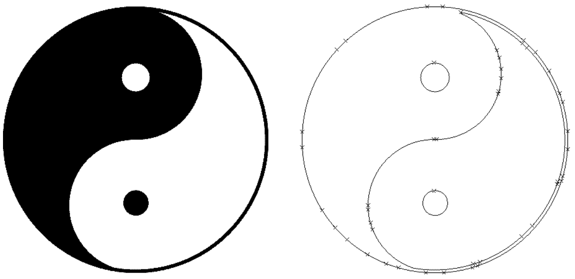


Figure 3: Function approximated image (on the right) showing joint points between contour segments used by Kwan, Kameyama and Toraichi (2003)

Because the heuristic distance function is defined using probabilistic variables whose final values are not known until after the matching has converged, the distances from the query to all other objects in the space cannot be accurately determined even when the query is entered. Further, the space of all objects is non-metric, in the sense of obeying the triangle inequality on distances, rendering it difficult to designate one database object as the vantage point for a possible index (Bozkaya and Ozsoyoglu 1997).

The approximate query processing approach of Kwan et al. (2003) and thereby, the "filter and refine" steps of the proposed strategy enables a close "approximation" of the retrieval result while simultaneously reducing the amount of computation required. At its centre is the concept of a lower bounding distance function for filtering improbable database objects from the need of computing expensive *query distances*. Provided that a provable lower bounding distance function exists and employed, no relevant objects should be discarded by the filtering step. Because some objects that remain after filtering might not belong to the final result, a further refining step based on computing the actual query distances is required to eliminate any false alarms (Ciaccia and Patella 2002).

Whereas related work used *provable* lower bounds due to the metric nature of their similarity models, in research like ours where non-metric distance functions are used, a provable lower bound might not be easily found. As one approach for addressing this problem, in our work a *quasi lower bounding distance* function was introduced. This function (which could be many) is defined using both the non-metric distance function and a confidence factor. In practice, it is computed right after the *initial state* of a robust matching algorithm is set, but before the process of advancing towards a final state has commenced.

In our formulation, the following notations are defined:

D_{initp} ≡ distance calculated after initial state located;

D_{query} ≡ actual query distance;

D_{quasi} ≡ the quasi lower bounding distance;

c_factor ≡ the confidence factor.

The following condition should be held during the entire retrieval process:

$$D_{query} \le D_{initp} \qquad (8)$$

The role of the confidence factor is to facilitate adjusting D_{initp} for the dual purpose of minimizing the chance of false drops while avoiding excessive false alarms. It is applied in the following equation:

$$D_{quasi} = c\_factor * D_{initp}\ , 0 < c\_factor \le 1 \qquad (9)$$

Here, a heuristic procedure is introduced to determine the value of the confidence factor dynamically by treating it as a discrete random variable that takes the value of the mean of the cumulative sum of the ratio, D_{query} / D_{initp}, averaged by the number of times that D_{query} has computed. In other words, the confidence factor represents a running average of the ratio of D_{query} and D_{initp}. Taken over an infinite time interval, it approximates the expected mean, E[*c_factor*].

Changes to the query processing strategy of Kwan et al. (2003), likewise for other robust matching methods such as based on deformable template matching, hierarchical neural network, etc) were made in two places. The first change was made where the initial state of the relaxation labeling algorithm is set and the second concerned the filtering step that was enabled by D_{quasi}, calculated by the product of D_{inip} and *c_factor*. An important assumption is that the amount of computation spent in setting the initial system state is much less than that of computing the actual query distances. In the context of Kwan et al. (2003), the changes made are summarized as follows:

1. Rough matching between the query and *all* database images is performed all at once initially. For each match, an initial D_{initp} is calculated. This differs from Kwan, Kameyama and Toraichi (2003) in that rough matching between every (query, database image) pair is followed by actual distance calculation at once.
2. In the filtering step, D_{quasi} is computed by taking the product of D_{initp} and the running value of c_factor. A database image is a candidate for *refining* wherever less than k nearest neighbours have been found so far or when the following criterion is satisfied:

$$D_{quasi}(\mathrm{db}[i]) \le D_{query}(\text{current } k\text{th-}NN) \qquad (10)$$

Here, db([i]) refers to the current database image being compared, and kth-*NN* the kth element in the nearest neighbour list accumulated so far. Only those that satisfy (10) at their respective turn of comparison will have their actual D_{query} calculated. The D_{query} computed is used in updating the *k-NN* list. Further, the value of D_{query} / D_{initp} is used in updating the running value of c_factor to be used in matching the next database image.

Pseudo code for "filter and refine" steps is given below:

```
[BEGIN]
expected_c_factor = 0.0;
cumulative_c_factor = 0.0;
count = 0;

for (i : [1,NUMBER_DB_IMAGES]) {
   D_initp[i] = rough_matching(Query, DB_Image[i]);
}

for (i : [1,NUMBER_DB_IMAGES]) {
  if (Less than k images in NN-List) {
    D_query = fine_matching(Query, DB_Image[i]);
    Update the NN-List;
    count = count + 1;
    cumulative_c_factor += (D_query / D_initp[i]);
    expected_c_factor = cumulative_c_factor / count;
  } else {
    D_quasi = expected_c_factor * D_initp[i];
    if (D_quasi <= distance of kth-NN) {
       D_query = fine_matching(Query, DB_Image[i]);
       Update the NN-List;
       count = count + 1;
       expected_c_factor += (D_query / D_initp[i]);
       expected_c_factor = cumulative_c_factor / count;
    }
  }
}
Result ← The NN-List
[END]
```

The “filter” step works to produce a significant reduction in the number of expensive distance calculations for the “refine” step, which is supported by experimental results.

3 Related Work

First, the autocorrelations have been previously employed as features for 1*D* or 2*D* signal classification in a wide range of applications, like texture classification, face detection and recognition, EEG signal classification (McLaughlin and Raviv 1968, Kurita, Otsu and Sato 1992, Kreutz, Volpel and Janssen 1996). For example, in Kurita, Otsu and Sato (1992), a set of local autocorrelation kernels defined on a neighbourhood of 3 x 3 pixels is used for computing 25 local autocorrelation coefficients from a digital facial image by spatial convolution. The set of coefficients is used as values for the numerical vectors in a multi-dimensional feature space. On the other hand, in this work global rather than local autocorrelation between an image and itself is taken, resulting in horizontal and vertical autocorrelation plots which are then transformed into the frequency domain by the DFT to obtain the Fourier series coefficients used in the numerical vectors.

Second, many two-step “filter and refine” approaches have been reported in the literature, like Brinkhoff et al. (1994), Hafner et al. (1995), Korn et al. (1996), Ankerst et al. (1998) and Ciaccia and Patella (2002). Most of them used a provable lower bound on the query distance to filter out irrelevant database objects from further matching. Since a “true” lower bounding distance was used, no relevant objects would be discarded by the filtering step.

In our approach, because the non-metric distance function used involves probabilistic values that might not allow for a “true” lower bound be accurately determined without having to go through the expensive iterative probability updating until convergence, the concept of a *quasi lower bounding distance* is introduced. The quasi lower bounding distance is computed using the *same* non-metric distance function weighted by a confidence factor which resembles the *scaling factor* of Ciaccia and Patella (2002).

4 Experimental Evaluation

4.1 Evaluation Criteria

A number of experiments were performed on a database of 2,000 Japanese kamon images (System Product 2003). Kamons are family emblems traditionally that have both cultural and commercial values. Because their meanings are largely conveyed by shapes, they were chosen as the objects for our shape-based image retrieval experiments. There are three primary evaluation criteria as follows:

1. The effectiveness of using Fourier series coefficients derived from both the horizontal and the vertical autocorrelation plots as values of numerical vectors for the multi-dimensional index in the “prune” step.
2. The effectiveness of the combination of quasi lower bounding distance and confidence factor in reducing the number of distance calculations in the “filter and refine” steps while not compromising the recall.
3. The advantage of the proposed strategy over the 2-step “filter and refine” approach in terms of the overall reduction in the number of images matched.

4.2 Experimental Results

First, the results presented in Tables 1 and 2 collectively addresses evaluation criterion (1). The values in Table 1 are distances measured in the normalized Euclidean space E^d between each member of the group of 6 kamon images shown in Table 2. In this experiment, one pair of visually similar images from 3 of the 9 shape categories is chosen.

It is clear from Table 1 that the distance is closer between images that are visually similar in Table 2, namely they belong to the same group. This is supported by the shape of the autocorrelation plots generated from these images in Table 2, underlying the effectiveness of employing the set of Fourier series coefficients in the index for pruning.

Second, to address evaluation criterion (2), the result of a *k*-nearest neighbour query ($k = 20$) by using the method of Kwan, Kameyama and Toraichi (2003) is shown in Table 3. It will be used as the benchmark for recall for the rest of our experiments. Note that in Table 3, D_{query} refers to the query distance computed by using the original non-metric distance function while D_{index} is the distance measured in the indexed E^d space. While they do not exhibit full correlation, it is noteworthy that D_{index} alone could still be effective in approximate similarity ranking.

Because the second set of experiments is meant to verify how effective the filtering by the quasi lower bounding distance is achieved based on the number of database images that have to go through actual expensive distance calculations, two metrics are defined.

The first of these is a reduction ratio defined as follow:

$Reduction_I$ = (# images actually match / # total database images) ∗ 100% (11)

The second metric is the recall defined as follow:

$Recall_I$ = (# correct responses in the “filtered” result / # responses returned) ∗ 100% (12)

	K1	K2	K3	K4	K5	K6
K1	0	0.64	2.37	2.40	3.09	3.31
K2	0.64	0	2.43	2.38	3.00	3.07
K3	2.37	2.43	0	1.99	3.29	3.40
K4	2.40	2.38	1.99	0	3.08	3.09
K5	3.09	3.00	3.29	3.08	0	1.08
K6	3.31	3.07	3.40	3.09	1.08	0

Table 1: Distance in E^d space between the set of 6 images shown in Table 2

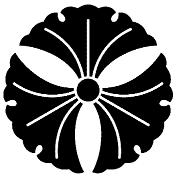
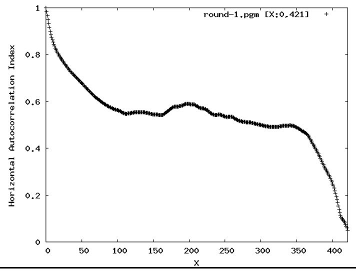
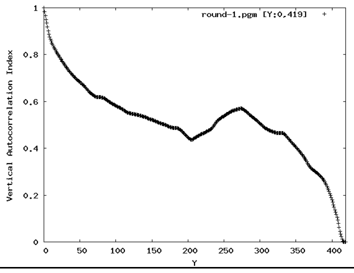
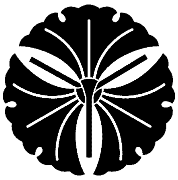
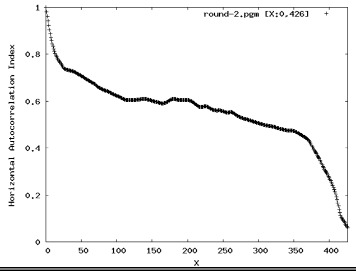
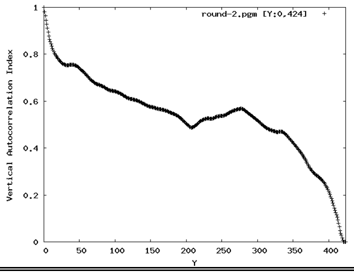
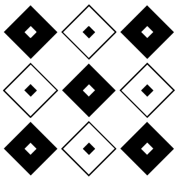
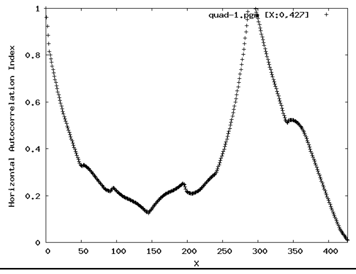
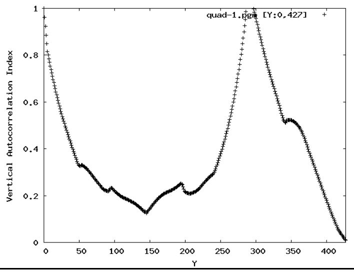
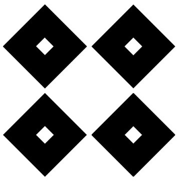
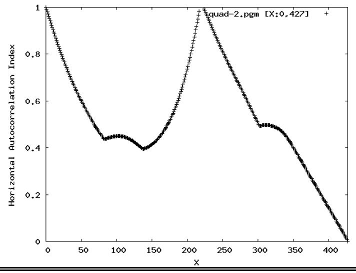
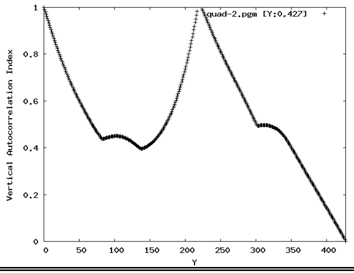
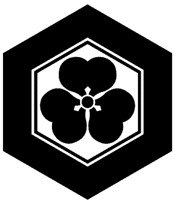
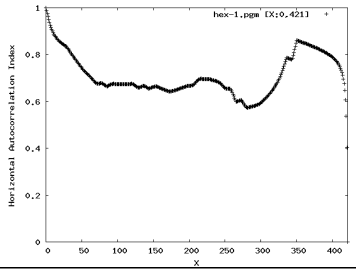
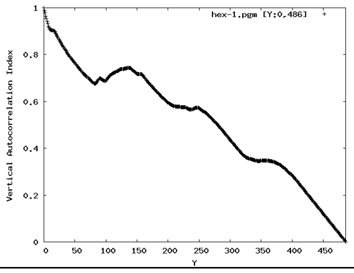
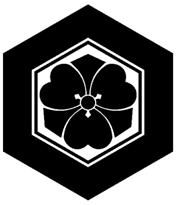
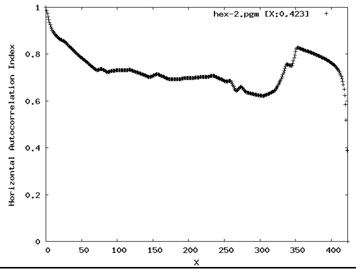
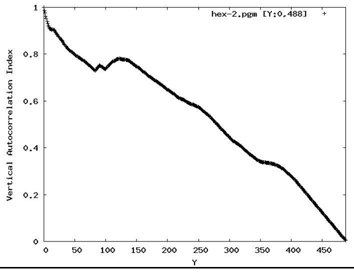
Shape Category		Horizontal Autocorrelation Plot	Vertical Autocorrelation Plot
Round	 (K1)		
	 (K2)		
Quadrangle	 (K3)		
	 (K4)		
Hexagon	 (K5)		
	 (K6)		

Table 2: Horizontal and vertical autocorrelation plots for pairs of similar images in 3 of 9 shape categories

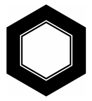
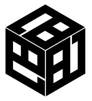
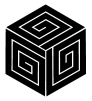
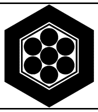
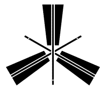
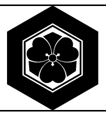
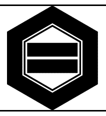
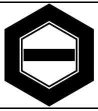
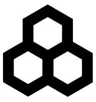
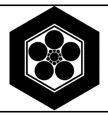
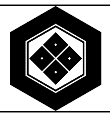
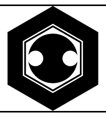
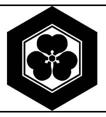
									
Rank: 1	2	3	4	5	6	7	8	9	10
D_{query}:0.0	0.61	0.66	0.71	0.74	0.75	0.77	0.78	0.79	0.80
D_{index}:0.0	0.754	0.558	1.526	1.272	2.819	2.211	1.351	1.492	1.623
									
11	12	13	14	15	16	17	18	19	20
0.81	0.82	0.84	0.86	0.866	0.869	0.872	0.875	0.883	0.886
1.630	1.592	1.937	1.548	1.429	1.347	1.491	1.374	1.339	1.573

Table 3: Result of a *k*-nearest neighbour query (*k* = 20) by using Kwan, Kameyama and Toraichi (2003)

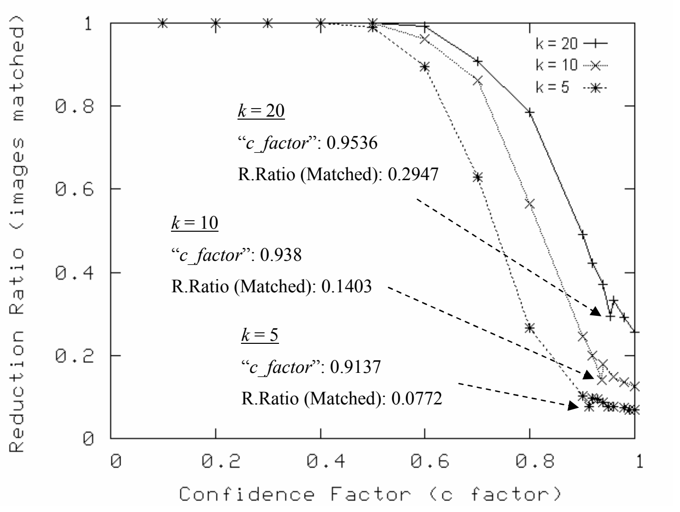


Figure 4: Reduction achieved by filter-and-refine

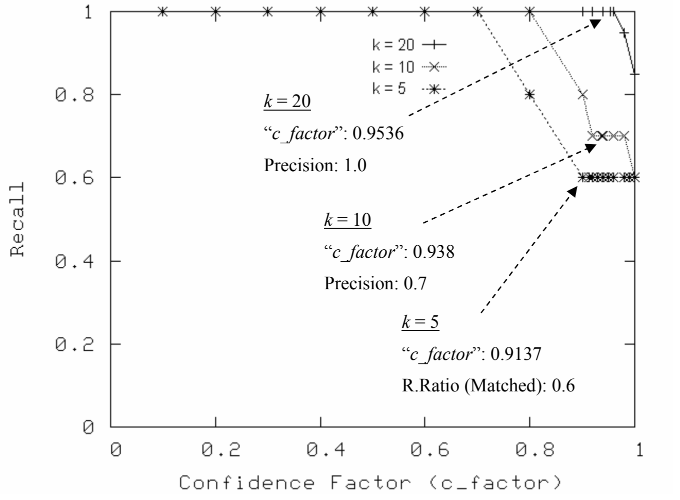


Figure 5: Relation between Recall and Reduction

Here, the denominator of $Recall_l$ is k, which is the number of nearest neighbours to return. $Recall_l$ is the percentage of correct responses that are included in the "filtered" result. In this group of experiments, for each setting of the c_factor (both manual and automatic), the $Reduction_l$ and $Recall_l$ are computed. Results are given in Figure 4 and 5.

For k = 20, one can notice that the "filter and refine" steps achieved a significant reduction (more than 70%) in number of actual distance calculations while the recall is maintained (that is, $Recall_l$ = 1.0) at the point where the confidence factor is deduced automatically. Compared to this, both for k = 10 and k = 5, although the reductions are greater, $Recall_l$ suffered. Nonetheless, in applications where either the number of nearest neighbours to return is not overly small (about 20 in our case) or that approximate results can be accepted, the savings in computation by the "filter and refine" steps are highly significant.

Range (r)	**Recall** ($Recall_2$)	**Reduction Ratio** ($Reduction_2$)
5.0	1.0	1.0
4.0	1.0	0.99
3.0	1.0	0.94
2.9	1.0	0.92
2.8	0.95	0.90
2.6	0.95	0.84
2.4	0.95	0.74
2.2	0.90	0.65
2.0	0.90	0.55
1.8	0.85	0.42
1.6	0.75	0.28
1.4	0.4	0.13

Table 4: Recall versus reduction ratio by varying *r*

Finally, to address criterion (3), we combine the previous results and that of the following experiment. Although at the moment there is no effective way to deduce an optimal value for range *r* as a function of the query that can provide maximum pruning while minimizing the number of false drops, it is still possible to heuristically deduce an approximate value for the database of our experiment via simulation. This result summarized in Table 4.

Two additional metrics are defined for evaluating the performance, namely $Recall_2$ and $Reduction_2$. Their definitions are given as follows:

$$Recall_2 = k' / k \qquad (13)$$

Here, *k'* is the number of correct answers in Table 3 that remain after pruning. That is, $0 \leq Recall_2 \leq 1$ holds.

$$Reduction_2 = n' / n \qquad (14)$$

Here, n and n' are the total size of the database and the number of images remained after "pruning".

From Table 4, it is reasonable to conclude that the optimal value *r'* should satisfy this condition, $2.8 < r' \leq 2.9$ for the query and the database that are used. At $r = 2.9$, pruning is about 8% while the recall is 1.0. When we assumed that pruning using a value of $r = 2.9$ has been done before the "filter and refine" steps are performed in the proposed multi-step strategy, the overall reduction in the actual distance calculations can be close to 80% for $k = 20$.

5 Conclusion

In this paper, a novel three-step "prune-filter-refine" query processing strategy for approximate similarity search in image databases is described. First, the "prune" step adopts a k-d-B tree based multi-dimensional spatial index to eliminate improbable matches via an adjustable distance threshold. Second, the "filter" step makes use of a quasi lower-bounding distance derived from the original non-metric distance function of the underlying similarity model to further reduce the number of candidates for detailed matching. Third, the "refine" stage evaluates the remaining candidates by a robust matching method to return the final similarity ranking. Experimental results on a shape-based retrieval system for Japanese kamon images verified that the proposed strategy achieves larger overall reduction in actual distance calculations than two-step approaches with close to no false drops in the final result.

6 References

Ankerst, M., Kriegel, H.-P. and Seidl, T. (1998): A multistep approach for shape similarity search in image databases. *IEEE Trans. Knowl. Data Eng.* **10** (6):996-1004.

Bentley, J. L. and Friedman, J. H. (1979): Data structures for range searching. *ACM Computing Surveys* **11** (4): 397-409.

Bozkaya, T. and Ozsoyoglu, M. (1997): Distance-Based Indexing for High-Dimensional Metric Spaces. In *Proc. ACM SIGMOD International Conference on Management of Data*, 357-368.

Brinkhoff, T., Kriegel, H.-P., Schneider, R. and Seeger, B. (1994): Multi-step processing of spatial joins. In *Proc. ACM SIGMOD International Conference on Management of Data,* Minneapolis, Minn., **23**:197-208, ACM Press.

Ciaccia, P. and Patella, M. (2002): Searching in Metric Spaces with User-Defined and Approximate Distances. *ACM Trans. Database Systems.* **27** (4):398-437.

Comer, D. (1979): The ubiquitous B-tree. *ACM Computing Surveys* **11** (2): 121-138.

Gaede, V. and Günther, O. (1998): Multidimensional Access Methods. *ACM Computing Surveys* **30** (2):170-231.

Hafner, J., Sawhney, H.S., Equitz, W., Flickner, M. and Niblack, W. (1995): Efficient Color Histogram Indexing for Quadratic Form Distance Functions. *IEEE Trans. Patt. Anal. Mach. Intell.* **17** (7):729-736.

Jacobs, D., Weinshall, D. and Gdalyahu, Y. (2000): Classification with Nonmetric Distances: Image Retrieval and Class Representation. *IEEE Trans. Patt. Anal. Mach. Intell.* **22** (6):583-600.

Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E. and Protopapas, Z. (1996): Fast Nearest Neighbor Search in Medical Image Databases. In *Proc. of the 22nd VLDB Conference*, Mumbai, India, 215-226.

Kreutz, M., Volpel, B. and Janssen, H. (1996): Scale-Invariant Image Recognition Based on Higher Order Autocorrelation Features. *Pattern Recognition* **29** (1).

Kurita, T., Otsu, N. and Sato, T. (1992): A Face Recognition Method Using Higher Order Local Autocorrelation and Multivariate Analysis. In *Proc. 11th IAPR International Conference on Pattern Recognition*, The Hague, 213-216.

Kwan, P., Kameyama, K. and Toraichi, K. (2003): On a relaxation-labeling algorithm for real-time contour-based image similarity retrieval. *Image and Vision Computing*. **21** (3):285-294.

Kwan, P., Toraichi, K., Kitagawa, H. and Kameyama, K. (2003): Approximate Query Processing for a Content-Based Image Retrieval Method. In: V. Malik et al.(Eds.): *DEXA 2003*, LNCS 2736, Springer-Verlag, 517-526.

McLaughlin, J. A. and Raviv, J. (1968): Nth-order autocorrelations in pattern recognition. *Information and Control* **12**:121-142.

Nagashima, H., Tsubaki, S. and Nakajima, J. (2003): A Classification for Trademark Images Using the Auto-correlation Function Graph Figure. *IEEJ Trans. EIS*. **123** (9):1547-1554.

Rafiei, D. and Mendelzon, A. (1997): Similarity-Based Queries for Time Series Data. In *Proc. ACM SIGMOD International Conference on Management of Data*, Tucson, Ariz., USA, 13-24.

Robinson, J. T. (1981): The K-D-B-tree: A search structure for large multidimensional dynamic indexes. In *Proc. ACM SIGMOD International Conference on Management of Data*, New Jersey, USA, 322-331.

System Product Co. Ltd. (2003), "Home Page", http://www.e-spc.co.jp/hp/product_3_3.htm

Weber, R., Schek, H-J. and Blott, S. (1998): A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *Proc. of the 24th VLDB Conference*, New York, USA, 194-205.

A New Approach to Intelligent Text Filtering Based on Novelty Detection

Randa Kassab **Jean-Charles Lamirel**

LORIA - INRIA Lorraine
Campus scientifique - BP. 239
54506 Vandoeuvre-lès-Nancy
France
Email: {kassabr,lamirel}@loria.fr

Abstract

This paper presents an original approach to modelling user's information need in text filtering environment. This approach relies on a specific novelty detection model which allows both accurate learning of user's profile and evaluation of the coherency of user's behaviour during his interaction with the system. Thanks to an online learning algorithm, the novelty detection model is also able to track changes in user's interests over time.

The proposed approach has been successfully tested on the Reuters-21578 benchmark. The experimental results prove that this approach significantly outperforms the well-known Rocchio's learning algorithm.

Keywords: novelty detection, information filtering, personalization, profile, online learning.

1 Introduction

Consequently to the constant increase of information available on the Web, in digital libraries and in other similar resources, new techniques for personalized information access have become more and more important. Information filtering is one of the most useful and challenging tasks for effective information access. It is concerned with dynamically adapting the distribution of information where both evolving user's interests and new incoming information are taken into account. For building up a model of user's information need, also called user's profile, information filtering relies on user's positive examples, represented by documents he likes, and possibly on user's negative examples, represented by documents he dislikes. This profile is furthermore used to automatically separate relevant from irrelevant documents in an incoming stream.

A wide range of machine learning algorithms and information retrieval techniques have been applied to text filtering task, including the Rocchio's linear classifier, k-nearest neighbours, Bayesian classifiers, neural networks, Support Vector Machines and boosting (Ault & Yang 2001, Schapire & Singer 1998, Schutze, Hull & Pedersen 1995, Dumais, Platt, Heckerman & Sahami 1998, Shankar & Karypis 2000). All these techniques focus on exactly learning user's profile in order to filter the content that would be interesting for the user as accurately as possible. This practice is very suitable considering the fact that the performance of the filtering process is extensively dependent on the accuracy of the user's profile. However, these techniques could not provide any information about the user's behaviour during his interaction with the filtering system. In our opinion, user's information need can be of different types, e.g. precise, exploratory or thematic (Lamirel & Créhange 1994); and according to these types the system must adapt the filtering results in a specific way. Moreover, most of the existing methods operate in an off-line mode where all the training documents have to be stored. Consequently, at each time a new training example need to be added, the system has to restart the training from the beginning. Such learning mode is not appropriate in online applications where memory space is limited and real-time filtering response is crucial.

This paper presents a new approach to text filtering based on the novelty detection principle. The main objective of novelty detection is to emphasize the novelty in yet unseen document with respect to previously learned ones. The basic idea is to learn a model of available documents and to use it for identifying the dissimilar documents (novelty). In filtering context, the novelty detection principle is mainly applied in a reverse way, i.e. the documents that are similar to a model learned from positive examples of user's need will be selected. The specific novelty detector filter (NDF) we use, is an adaptation of a former novelty detector model proposed by Kohonen (kohonen 1984) and based on the orthogonal projection operators. The most powerful feature of this filter is its ability to accurately learn user's profile and to evaluate, in a parallel way, the coherency of user's behaviour during his interaction with the system. Thanks to an online learning algorithm, the novelty filter is also able to track changes in user's interests over time.

The novelty detector filter is investigated on the Reuters-21578 benchmark. We compare experimentally its performance with the widely used Rocchio's algorithm; our experiments clearly highlight that the novelty detector filter is more effective than this latter algorithm.

The paper is organized as follows: Section 2 introduces the basic concepts of the novelty detector filter. Section 3 describes our adaptation of this filter for the text filtering task. Section 4 reports on our experiments. The paper ends with the description of our future work.

2 The Novelty Detector Filter

The novelty detector filter is a linear adaptive system which acts, after its learning on a reference data, as a projection operator in a vector space that is orthogonal to the vector space spanned by the reference data. Consider figure 1; the NDF only passes

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

through the "novelty" component of a data with respect to the previously learned reference data. The residual component which is orthogonal to the novelty component is known as the "habituation" component. Such a system has a transfer function equivalent to a square matrix : $\emptyset_k = I - X_k X_k{}^+$ where $X_k = [x_1, x_2, \ldots x_k]$ is the reference data matrix in which each x_i is a n-dimensional vector, $X_k{}^+$ is the penrose pseudo-inverse of X_k, I denotes the identity matrix. The inputs and the outputs are thus related by $x' = \emptyset_k \cdot x$.

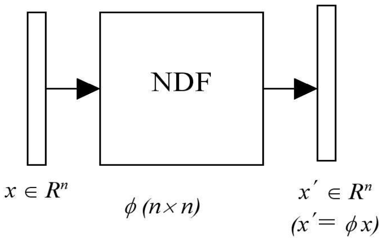


Figure 1: The model of the novelty detector filter

The mathematical learning model of the novelty detector is based on the theorem of Greville (kohonen 1984) which yields a recursive expression for calculating the transfer function of the filter. After simplification, the theorem can be expressed as:

$$\emptyset_k = \emptyset_{k-1} - \frac{x\prime_k x\prime_k^T}{\|x\prime_k\|^2} \tag{1}$$

where $x_k{}' = \emptyset_{k-1} \cdot x_k$ represents the orthogonal projection of the vector x_k on a space that is orthogonal to the space spanned by the $k-1$ learned vectors; $\|x\|$ represents the length of the vector x; and the recursion starts with $\emptyset_0 = I$.

Once the learning phase described above is over, the filter becomes habituated to the reference data. Thus, if one of the reference data or their arbitrary linear combination is applied to the filter input, the novelty output will be zero. On the other hand, if a novel data not belonging to the space spanned by the reference data is chosen as an input, the corresponding output will be nonzero and can be seen as representative of the new features extracted from the input data with respect to the reference data.

3 The Novelty Detector Filter in Text Filtering

The principle of novelty detection is particularly interesting for text filtering. In this context, a data corresponds to a text document which is itself represented by a weighted feature vector in a n-dimensional description space, where n is the total number of features extracted from the training documents (i.e. all the documents currently available in the system). The reference data are the documents marked by the user as examples of his information need, i.e. the positive training examples.

Our previous evaluations of the NDF for text filtering (Kassab, Lamirel & Nauer 2005, Kassab, Lamirel & Nauer 2005) enable us to observe that the performance of the NDF is quite effective in the case of single-label datasets where each document belongs exactly to one category. However, it fails in the multi-label case where a document may be belong to more than one category. Although the latter case is relatively more difficult to process for all the existing approaches, since high correlation between relevant and irrelevant documents is strongly probable, it has proved much harder than expected for the NDF. This can be mainly explained by the fact that the learning rule (Eq.1), which is essentially designed for novelty detection, is not directly applicable to filtering task. In fact, the novelty detection learning rule is mainly intended to distinguish the novelty parts from the old or habituated parts in the input data with respect to the previously learned reference data, regardless of whether a learned data is more seen in the reference data than another or not. Hence, such a learning rule does not allow cumulative learning of the user's need. In other words, both training examples which are regarded as redundant (i.e. those that do not carry novelty, that is, those for which x' is null) and the features that have been totally learned (i.e. those for which the novelty vector is null when applying the unit vector associated to the feature as input) are no more taken into account during training. Therefore, it is often possible to learn the discriminating and non discriminating features with the same degree of relevancy. Accordingly, a good separation between relevant and irrelevant documents will not always be easy to achieve. An adaptation of the learning rule to filtering is then required.

Our proposal is to introduce the identity matrix in the learning formula for considering separately all training examples, and consequently all their features, during the learning phase. The new learning rule is defined as:

$$\emptyset_k = I + \emptyset_{k-1} - \frac{x\prime_k x\prime_k^T}{\|x\prime_k\|^2} \tag{2}$$

where $x_k{}' = (I + \emptyset_{k-1})x_k$ and $\emptyset_0$ is a zero, or null, matrix.

As learning progresses, features which frequently appear in the training documents become more and more habituated as compared to the less frequent ones. This typically helps to discriminate more accurately the relevant and irrelevant documents. An example and further details about the defects of the original learning rule of the NDF (Eq.1) when applied to filtering task and how our modified rule (Eq.2) can improve performance are provided in the appendix.

Since learning from only positive examples is suitable for several applications, two strategies are considered and tested for using the novelty detector filter in text filtering system according to the kind of available training examples.

3.1 Positive training examples only

The learning of a NDF on a set of positive training examples permits to calculate the transfer function $\emptyset$ of the filter. This function is a projection matrix that represents a space that is orthogonal to the space spanned by the positive training examples. Then, the projection of each yet unseen document, say d, on the filter matrix $\emptyset$ will generate a novelty vector $d' = \emptyset \cdot d$. Two proportions can be thus computed:

- The novelty proportion which quantifies the amount of novelty in the document under consideration with respect to the documents that have been seen during training.

$$N_d = \frac{\|d'\|}{n \times \|d\|} \tag{3}$$

where n is the number of training documents.

- The habituation proportion which quantifies the similarity of the document with the previously learned ones.

$$H_d = 1 - \frac{\|d'\|}{n \times \|d\|} \tag{4}$$

This later proportion could be considered as the relevance score of the document d and thus be used for ranking documents: the higher the habituation proportion is, the more relevant the document will be.

3.2 Positive and negative training examples

When training set consists of positive and negative examples of user's need, two novelty detector filters should be used. The acceptance filter $\emptyset_A$ which is learned from the positive examples and the rejection filter $\emptyset_R$ learned from the negative examples. After learning, the relevance score of each new document d is computed using the following formula:

$$R_d = \beta \cdot H_{Ad} - \gamma \cdot H_{Rd} \tag{5}$$

where H_{Ad} is the habituation proportion of the document d using the acceptance filter;
H_{Rd} is the habituation proportion of the document d using the rejection filter;
β, γ are positive parameters which control the relative importance of the positive and negative examples respectively.

Concept of saturation

Since most filtering systems use all features assigned to positive or negative examples for modelling user's profile, this could naturally lead to an increased noise in this profile and consequently to reduce filtering accuracy. Nevertheless, if the accuracy becomes too low this phenomenon can surely be imputed to the user who has not been able to correctly formulate his need. Unfortunately, to the best of our knowledge, this fact is not taken into account in most existing systems despite its great importance for choosing a well-suited strategy for adjusting a dissemination threshold.

The saturation of a NDF (S_{NDF}) may be understood as the inability of this filter to extract new features with respect to the learned documents. In other words, it corresponds to the learning by the filter of all features of the description space. This case can occur if the number of the learned documents is such as it generates a subspace whose dimension is equal to the dimension of the description space. We define the saturation of a NDF as the ratio of the number of learned features[1] to the total number of features in the description space.

The saturation value is useful for assessing the type of user's information need. This later can be considered as precise when the saturation is quite low and thus the filtering threshold must be set to a high value, while the user's need can be regarded as exploratory when the saturation is very high, or maximal, and in this case the filtering threshold must be set to a low value[2] (see figure 2).

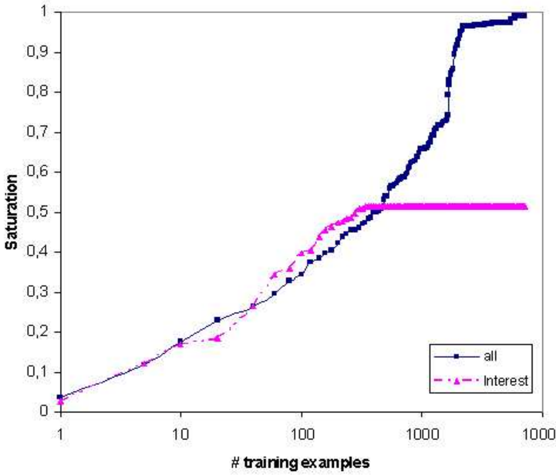


Figure 2: Evolution of the saturation when learning from all 10 categories and when learning only from one category (e.g. Interest)

4 The Rocchio Algorithm

Rocchio's Algorithm is a relevance feedback algorithm that has been widely used for improving the performance of information retrieval systems (Rocchio 1971) and that has been then adapted to text filtering task (Schapire et al. 1998, Ault et al. 2001). It allows computing of a prototype profile P_c as the weighted difference of the centroid vectors of the positive and the negative examples:

$$P_c = \beta \cdot \frac{\sum_{d \in R} d}{|R|} - \gamma \cdot \frac{\sum_{d \in N} d}{|N|} \tag{6}$$

The parameters β and γ control the relative impact of the positive and negative examples; $|R|$ and $|N|$ are respectively the number of positive and negative examples. The profile P_c is restricted to non negative values.

Ranking is then achieved by performing a cosine similarity between the prototype profile and each new document.

5 Experiments

In this section, we describe our experiments for testing the performance of the novelty detector filter in text filtering environment. Throughout the present study we focus our attention on the evaluation of the quality of the learned profile, i.e. the filter's matrix, for representing user's information need and thus for ranking documents by relevancy. Hence, we did not use any dissemination threshold but we plan to investigate this strategy in near future.

Experimental results on the Reuters-21578 collection are presented, demonstrating that our approach is more effective than the Rocchio's learning algorithm.

5.1 Reuters collection

We conducted our experiments on the Reuters-21578 collection[3]. The documents of this collection are divided into training and test sets and they are labelled by 135 categories. Our experimental results are reported for the set of the top 10 categories with the highest number of positive training documents.

[1] We mean here by learned features, the features for which the habituation proportion is higher than zero or a predefined threshold.

[2] We expect that a threshold set to the value $(1 - S_{NDF})$ would be suitable, but we have not yet compared it with other thresholding strategies.

[3] The Reuters-21578 collection is publicly available at : http://www.daviddlewis.com/resources/testcollections/reuters21578/

Table 1: # training and test examples in the top 10 categories of Reuters collection

Category name	# train	# test
acq	1650	719
corn	181	56
crude	389	189
earn	2877	1087
grain	433	149
interest	347	131
money-fx	538	179
ship	197	89
trade	369	117
wheat	212	71

Table1 gives the number of training and test documents in each of the categories.

Usually, the training documents assigned to each category are used as positive examples of user's need and the rest of training documents in the other categories as negative examples. Due to the relative high quantity of the negative examples in Reuters collection, we have selected the negative examples using the query zoning method (Singhal, Mitra & Buckley 1997, Schapire et al. 1998), where only the $|R|$ top ranking documents retrieved from the negative training examples by the centroid vector of the positive examples are used. This method has proved to be efficient for Rocchio's algorithm.

5.2 Preprocessing

We performed standard preprocessing steps: document parsing, tokenization, stop words removal. Only single words were used for content representation. The highly discriminating word features were selected using the chi-square statistic (Yang & Pedersen 1997) as follows. Given a set of candidate features, we compute the feature goodness for each category as:

$$x(f,c) = \frac{\sqrt{N} \times (AD - CB)}{\sqrt{(A+C) \times (B+D) \times (A+B) \times (C+D)}} \qquad (7)$$

where N is the total number of training documents; A is the number of times f and c co-occur; B is the number of times f occurs without c; C is the number of times c occurs without c; D is the number of times neither f nor c occurs.

Then, the values were sorted in descending order and the top 50 features were chosen for each category[4] and used for forming a global description space of 379 features. These later were thus used for representing both training and test documents by vectors of feature-weights. These weights are calculated using TF×IDF weighting (Salton & Buckley 1988) for training examples and only TF weights for test examples[5]. The vectors are normalized using the cosine normalization method (Salton et al. 1988), so that each document has a length of 1.

5.3 Performance measure

Evaluation is achieved using average uninterpolated precision metric, which is widely used for TREC[6] routing tasks without dissemination thresholds(Robertson & Soboroff 2001). It is defined as the sum of the precision value at each point where a relevant document appears in the ranked list, divided by the total number of relevant documents. Relevant documents which are not retrieved within the top 1000 receive a precision of zero.

Table 2: Comparison of performance results for the original learning rule (NDRule) and our modified learning rule (FRule) using only positive examples

Category name	NDRule	FRule
acq	74.50	95.65
corn	61.09	81.34
crude	40.49	91.65
earn	84.45	91.95
grain	64.48	97.82
interest	58.16	78.83
money-fx	50.95	76.70
ship	54.60	88.10
trade	33.30	93.35
wheat	59.34	87.61
mean AvgP	58.14	88.30

5.4 Results and discussion

For the purpose of demonstration, we firstly present the results from the original learning rule (Eq.1) and our modified learning rule (Eq.1) for the 10 most frequent categories from the Reuters collection. As it can be seen in Table2, our modified learning rule (FRule) substantially outperforms the original learning rule (NDRule). More specifically, NDRule yields 58.14% accuracy over the ten categories, while our modified learning rule FRule yields 88.30%, resulting in an overall improvement of 30%. The explanation of the substantial difference between the two learning rules has to do again with the fact that the original learning rule is not able to distinguish between discriminating and non discriminating features in training documents (see section 3). Consequentely, the failure of the NDRule becomes obvious when documents are represented by a few highly discriminating features against too many non discriminating ones. Hence, when we looked at the ranking performed by the NDRule we found a considerable confusion between the categories. As an example, the NDRule produces particularly bad results for trade because of its confusion with earn. As soon as trade shares almost all its features with earn, some of these features are certainly more relevant to trade than to earn. Nevertheless, the inability of the NDRule to discriminate such features led it to consider most earn documents as relevant and even more relevant to trade than its proper documents. On the other hand, earn is less sensitive to this phenomenon because of the higher number of the test documents associated to this category as compared to trade.

In the rest of this discussion we evaluate the performance of the NDF using our modified learning rule (FRule) with and without negative examples. Table3 summarizes our results using only positive training examples. Compared to Rocchio's method, NDF produces superior average precision on almost all categories with +3.25% overall performance. Although Rocchio is slightly better than NDF for some categories (viz. ship, trade, wheat), we believe that NDF learning is constantly better than Rocchio learning especially for distinguishing between discriminating and non discriminating features in training examples. This situation is especially obvious for the crude category whose documents are represented with a high ratio of discriminating features. In our opinion, one

[4] Selecting the top 50 feautures is solely based on the fact that the average length of the training documents after preprocessing is 47 words.

[5] The TF (term frequency) formula used in our study is: $TF = 1 + log_2(tf)$ where tf is the feature frequency within a document. The IDF (inverse document frequency) is defined as: $IDF = log_2(\frac{N+1}{n})$ where N is the total number of documents in the training collection, n the number of documents containing the feature.

[6] Text REtrieval Conference, TREC. http://trec.nist.gov/

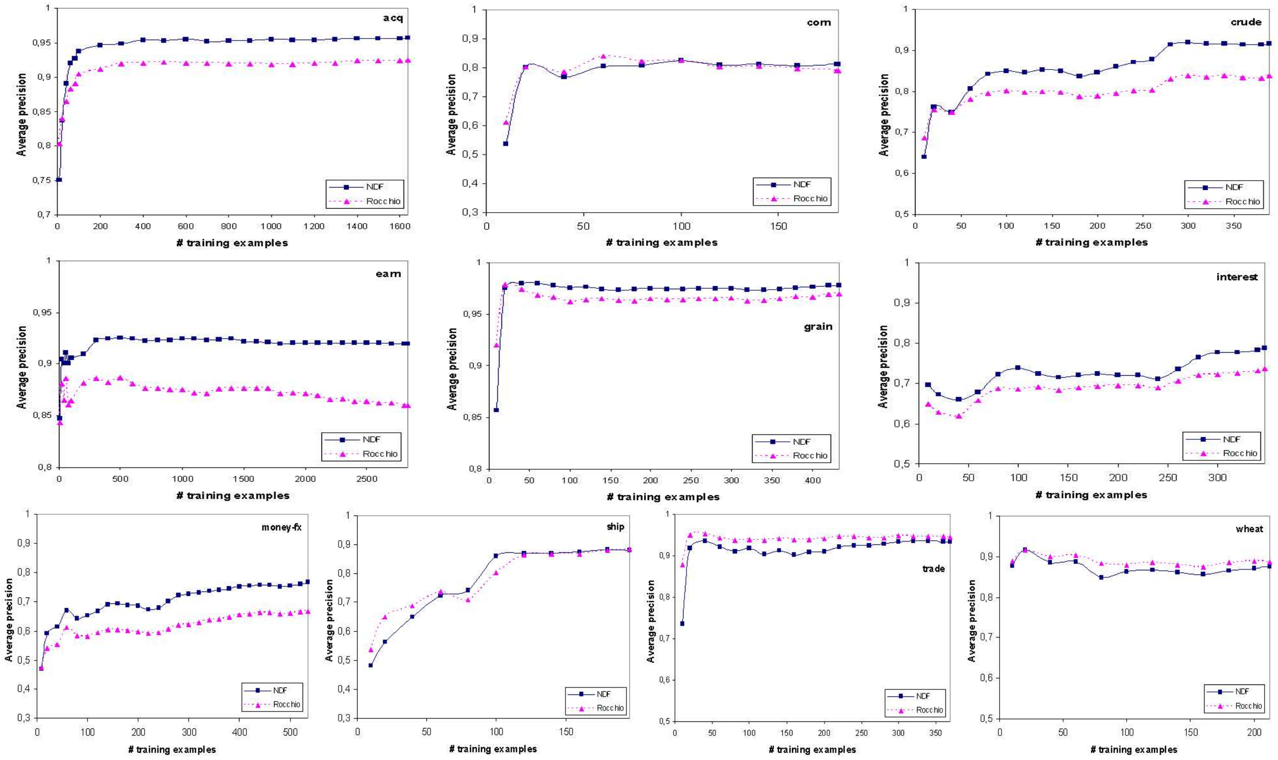


Figure 3: Comparison of NDF and Rocchio on the Reuters categories using only positive examples

reason of the small advantage of the Rocchio's method for few categories might be the presence of some outliers among the test documents which are extreme in comparison with the training documents, i.e. their discriminating features are quite different from those of the training documents. Thus, the NDF will advantage the outliers of the other categories that are similar to the positive training documents over the outliers of the training category. As a consequence, these outliers might disturb the ranking of the last relevant documents. On the other hand, Rocchio avoids this problem because its separation between discriminating and non discriminating features is less precise. We would also like to mention that Rocchio achieved much better effectiveness in our experiments than the current state-of-the-art Rocchio's method on the Reuters collection, see (Dumais et al. 1998, Shankar et al. 2000). We think that the results are quite sensitive to the indexing and feature selection methods. For example, unlike the claims reported in (Ault et al. 2001), we found that chi-square test leads to a considerable improvement in performance on the Reuters collection. Nevertheless, this improvement depends on how the selection criterion is applied. A general practice is to use a global measure that averages the chi values over the number of categories (Liu, Liu, Chen & Ma 2003, Yang et al. 1997). This strategy enables to eliminate the most common features, but not to extract the most discriminating features between categories. For this reason, we have measured individually the relevancy of a feature for each category and then selected the top 50 features of each category in order to form our global feature space.

To get a better understanding of the behavior of the NDF, we have calculated the average uninterpolated precision in relation to the number of positive training examples for the 10 categories. Considering the results we obtained, shown in figure 3, we observe that NDF and Rocchio are closely competitive when training from very few postive examples. However, as the number of positive training examples increases, NDF is consistently superior to Rocchio for almost all categories.

Table 3: AvgP using only positive examples

Category name	Rocchio	NDF
acq	92.46	95.65
corn	78.88	81.34
crude	83.88	91.65
earn	86.04	91.95
grain	97.02	97.82
interest	73.73	78.83
money-fx	66.75	76.70
ship	88.55	88.10
trade	94.48	93.35
wheat	88.72	87.61
mean AvgP	85.05	88.30

As expected, exploiting negative examples yields better results for both Rocchio's and NDF approaches. Table4 shows the average precision for $\beta = 2, \gamma = 1$ (see equations 5,6). Despite the significant improvement in the performance of Rocchio's method, we observe that the NDF remains more effective through all categories.

In order to explore the sensitivity of our proposed approach to the control parameters setting, we conduct experiments using different values of these parameters. The mean uninterpolated average precision over the categories is depicted in table5. Conversely to (Singhal et al. 1997), our results show that $\beta = \gamma$ is unsuitable for multi-label collection, especially when both query zoning and feature selection techniques are jointly applied. Our hypothesis is that for the most similar categories, like corn, grain and wheat, which have many features in common, such parameter settings could yield the elimination of the most expressive features of the relevant category and thus the irrelevant documents would be favoured over the relevant ones. This effect becomes evident when fea-

Table 4: AvgP using positive and negative examples

Category name	Rocchio	NDF
acq	95.29	97.27
corn	95.45	95.94
crude	92.96	93.27
earn	90.02	92.66
grain	97.80	98.50
interest	84.04	90.92
money-fx	80.41	88.99
ship	88.38	89.32
trade	95.73	96.87
wheat	91.75	91.95
mean AvgP	91.18	93.57

Table 5: Mean average precision for different β, γ

β	γ	Rocchio	NDF
1	1	78.57	58.01
16	4	88.90	91.62
0.75	0.25	89.77	92.50
2	1	91.18	**9**3.57

ture selection method is applied. In a concrete example, assuming that a document d has a positive similarity 0.8 with the acceptance filter (respectively, the centroid of positive examples in Rocchio's method) and a negative similarity 0.9 with the rejection filter (respectively, the centroid of negative examples) the relevance score of this document will be -0.1. Thus, this document which is a common document between relevant category and one or more of the negative categories would be ranked under the irrelevant documents which have very low or zero similarity with the positive and negative examples. In this case, Rocchio will perform better than NDF because it prohibits this effect by arbitrarily setting the negative components to zero.

6 Conclusion

We have presented a promising approach to modelling user's information need in intelligent text filtering environment. This approach is based on a specific novelty detection model, the NDF, which allows to accurately learn user's profile and which provides, in a parallel way, a more comprehensive capture of the type of user's information need. Experiments carried out on the Reuters collection showed the effectiveness of the NDF as compared to Rocchio's method. Moreover, when combining our method with a suitable feature selection approach, the results seem to be better than the ones ever reported for the best offline learning method, like KNN, SVM and bootstrap. Hence, in a further step, we plan to more thoroughly compare our approach with these latter methods.

Nevertheless, a more extensive evaluation is needed, concerning especially the dissemination thresholds setting and the performance of the NDF for adaptive filtering task (see definition in (Robertson et al. 2001)). Last but not least, we plan to exploit the ability of our approach to detect the novelty for controlling the amount of redundancy in filtering results according to the type of user's information need (Zhang, Callan & Minka 2002). Hence, the filtering result should exactly response to user's need when this later is precise, whereas, if the type of the user's information need is exploratory, the filtering result should firstly recover as much as possible the user's need. In this latter case redundancy could be present and new information could also be recommended to the user.

References

Ault, T. & Yang, Y. (2001), kNN, Rocchio and Metrics for Information Filtering at TREC-10, *in* 'The Tenth Text REtrieval Conference (TREC 10)'

Dumais, S., Platt, J., Heckerman, D.,& Sahami, M (1998), Inductive Learning Algorithms and Representations for Text Categorization, *in* 'Proceedings of the Seventh International Conference on Information and Knowledge Management CIKM', pp. 148–155.

Kassab, R., Lamirel, J.C., & Nauer, E. (2005), Novelty Detection for Modeling User's Profile, *in* 'Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference (FLAIRS 05)', Clearwater Beach, Florida, AAAI Press. pp. 830–831.

Kassab, R., Lamirel, J.C., & Nauer, E. (2005), Une nouvelle approche pour la modélisation du profil de l'utilisateur dans les systèmes de filtrage d'information : le modèle de filtre détecteur de nouveauté, *in* 'The Deuxième Confrence en Recherche d'information et Applications, CORIA'05', France, pp. 185–200.

Kohonen, T. (1984), *Self organisation and associative memory*, Springer Verlag, New York, USA.

Lamirel, J.C., & Créhange, M. (1994), Application of a Symbolico-Connectionist Approach for the Design of a Highly Interactive Documentary Database Interrogation System with On-Line Learning Capabilities, *in* 'Proceedings of the Third International Conference on Information and Knowledge Management (CIKM 94)', pp. 155–163.

Liu, T., Liu, S., Chen, Z. & Ma, W. (2003), An Evaluation on Feature Selection for Text Clustering, *in* 'Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)'.

Robertson, S.E. & Soboroff, I. (2001), The TREC-9 Filtering Track Final Report, *in* 'Proceedings of the 9th Text REtrieval Conference (TREC 9)', pp. 25–40.

Rocchio, J J (1971), *Relevance feedback in information retrieval, In The SMART Retrieval System : Experiments in Automatic Document Processing*, Prentice Hall Inc., Englewood Cliffs, New Jersey.

Salton, G. & Buckley, C. (1988), Term weighting approaches in automatic text retrieval, *in* 'Information Processing and Management', 24(5), 513–523.

Schapire, R., Singer, Y. & Singhal, A. (1998), Boosting and Rocchio Applied to Text Filtering, *in* 'Proceedings of the Twenty-first Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', pp. 215–223.

Schutze, H., Hull, David A. & Pedersen, Jan O. (1995), A Comparison of Classifiers and Document Representations for the Routing Problem, *in* 'Proceedings of SIGIR'95, the International Conference on Research and Development in Information Retrieval (1995)', ACM Press, pp. 229–237.

Shankar, S. & Karypis, George(2000), Weight Adjustment Schemes for a Centroid Based Classifier, 'Computer Science Technical Report (TR00-035)', Department of Computer Science, University of Minnesota, Minneapolis, Minnesota.

Singhal, A., Mitra, M.& Buckley, C. (1997), Learning routing queries in a query zone, *in* 'Proceedings SIGIR'97, 20th ACM International Conference on Research and Development in Information Retrieval', pp. 25–32.

Yang, Y. & Pedersen, Jan O. (1997), A Comparative Study on Feature Selection in Text Categorization, *in* 'Proceedings of ICML-97, 14th International Conference on Machine Learning', pp. 412–420.

Zhang, Y., Callan, J.& Minka, T. (2002), Novelty and redundancy detection in adaptive filtering, *in* 'Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval', pp. 81–88.

Appendix

Illustrative example of the NDF learning model

In sections 2 and 3 we have outlined the basic concepts of the NDF model and our adaptation of this model to text filtering task. This appendix presents a simple example showing the main defect of the original learning rule of the NDF (Eq.1) when applied to filtering task and why a modification of this rule is required. The example also illustrates how the modified rule (Eq.2) can enhance the efficiency of the filtering process, in both learning user's profile and ranking the filtering result. We will refer to the original learning rule as NDRule and to the modified learning rule as FRule.

Table 6: Sample feature-by-document matrix (5 features × 7 documents)

	Train				Test		
	d_1	d_2	d_3	d_4	d_5	d_6	d_7
f_1	1	1	0	0	1	0	1
f_2	0	0	0	1	0	0	0
f_3	0	1	1	1	0	1	0
f_4	0	0	1	1	0	1	0
f_5	0	0	0	0	1	0	0

Let us now consider the set of documents presented in table6 which contains four training documents and three new testing documents. Let us suppose that the two documents d_1, d_2 are selected by the user as examples of his information need among the four available documents. The system has to learn the user's profile using these two examples in order to rank the incoming testing documents by relevancy for the user.

As a result of running the NDRule on the reference documents d_1, d_2 we obtain the transfer matrix $\emptyset_2$ of the NDF:

1.

$$\emptyset_1 = \emptyset_0 - \frac{d\prime_1 d\prime_1^T}{||d\prime_1||^2}$$

where $\emptyset_0 = I$ and $d_1{}' = \emptyset_0 \cdot d_1$

2.

$$\emptyset_2 = \emptyset_1 - \frac{d\prime_2 d\prime_2^T}{||d\prime_2||^2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

with $d_2{}' = \emptyset_1 \cdot d_2$

On the basis of the above matrix, it is possible to calculate the habituation proportion of all features by projecting the unit vector U_f associated with each feature on the space spanned by the NDF; using the following formula:

$$H_f = 1 - \frac{|\emptyset_2 U_f|}{|U_f|}$$

we get:

$$H_{f1} = H_{f3} = 1, H_{f2} = H_{f4} = H_{f5} = 0$$

This means that the features f_1, f_3 have been totally learned and will have the same relevancy for representing the training documents, despite the fact that f_1 is more relevant for discriminating user's need than f_3 which could be a common feature (i.e. a non discriminating feature) among documents in collection. This is due to the fast learning of features that is characteristic to the NDRule. Hence once a feature is totally learned it will no more be considered since only the novelty vector is used during learning (i.e. the presence or absence of such a feature in the ensuing training documents will not affect learning in the next steps). This increases the learning possibility of non discriminating features.

Calculating now the relevance score of each new document which corresponds to the habituation proportion of each document, we get:

Rank	Document	Relevance score
1	d_7	1
2	d_5,d_6	0.29

Note that although d_5 is more relevant than d_6, they were ranked as having the same degree of relevancy[7]. The main reason for this is the inability of the NDRule to correctly distinguish between discriminating and non discriminating features. Modifying the learning strategy is then imperative to avoid this problem. In the following discussion we are going to show how our modification of the learning rule (FRule) can improve the filtering efficiency.

Starting from the modified learning rule FRule, the transfer matrix $\emptyset_2$ we obtain is:

$$\emptyset_2 = \begin{pmatrix} 0.8 & 0 & -0.4 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ -0.4 & 0 & 1.2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

If we now calculate the habituation proportions of the features according to Eq.4, we get:

[7] In more complex cases where there is a high similarity between the relevant and non relevant training and/or test documents, many relevant documents may have lower rank than non relevant ones.

$H_{f1} = 0.55, H_{f3} = 0.37, H_{f2} = H_{f4} = H_{f5} = 0$

As it can be seen, f_1 is now more habituated than f_3 and accordingly, f_1 will be considered as more relevant than f_3 for representing the user's need. The ranking of the new documents is shown in the following table:

Rank	Document	Relevance score
1	d_7	0.55
2	d_5	0.23
3	d_6	0.16

As it can be expected, the non relevant document d_6 is ranked with the lowest relevance score.

Using a Temporal Constraint Network for Business Process Execution[1]

Ruopeng Lu, Shazia Sadiq, Vineet Padmanabhan, Guido Governatori

School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, Australia
{ruopeng, shazia, vnair, guido}@itee.uq.edu.au

Abstract

Business process management (BPM) has emerged as a dominant technology in current enterprise systems and business solutions. However, the technology continues to face challenges in coping with dynamic business environments where requirements and goals are constantly changing. In this paper, we present a modelling framework for business processes that is conducive to dynamic change and the need for flexibility in execution. This framework is based on the notion of process constraints. Process constraints may be specified for any aspect of the process, such as task selection, control flow, resource allocation, etc. Our focus in this paper is on a set of scheduling constraints that are specified through a temporal constraint network. We will demonstrate how this specification can lead to increased flexibility in process execution, while maintaining a desired level of control. A key feature and strength of the approach is to use the power of constraints, while still preserving the intuition and visual appeal of graphical languages for process modelling.

Keywords: Process modelling; Workflows; Temporal constraints; Constraint Satisfaction

1 Introduction

It has been long established that automation of specific functions of enterprises will not provide the productivity gains for businesses unless support is provided for overall business process control and monitoring. Workflow systems have delivered effectively in this area for a class of business processes, but typical workflow systems have been under fire due to their lack of flexibility, i.e., their limited ability to adapt to changing business conditions. In the dynamic environment of e-business today, it is essential that technology supports the business to adapt to changing conditions, where different process models should be derived from existing ones to tailor individual process instances. However, this flexibility cannot come at the price of process control, which remains an essential requirement of process enforcement technologies.

Providing a workable balance between flexibility and control is indeed a challenge, especially if generic solutions are to be offered. Clearly, there are parts of the process which need to be strictly controlled through fully predefined models. There can also be parts of the same process for which some level of flexibility must b e offered, often because the process cannot be fully predefined due to lack of data at process design time. For example, in call centre responses, where customer inquiries and appropriate response cannot be completely pre-defined, or in health systems, where patient care procedures resulting from individual patient diagnosis cannot be anticipated.

In general, a process model needs to be capable of capturing multiple perspectives (Jablonski and Bussler 1996), in order to fully capture the business process. There are a number of proposals from academia and industry on the modelling environment (language) that allow these perspectives to be adequately described. Different proposals offer different level of expressiveness in terms of these perspectives. Basically these perspectives are intended to express the constraints under which the business process can be executed such that the targeted business goals can be effectively met.

We see three essential classes of constraints:

- **selection** constraints that define *what* activities constitute the process,
- **scheduling** constraints that define *when* these activities are to be performed, both in terms of ordering as well as temporal dependencies, and lastly
- **resource** constraints that define *which* resources are required to perform the activities.

These constraints are applicable at two different levels, *process* level and *activity* level. Process level constraints specify what activities must be included within the process, and the flow dependencies within these activities including the control dependencies (such as sequence, alternative, parallel etc.) and inter-activity temporal dependencies (such as relative deadlines). Activity level constraints constitute the specification of various properties of the individual activities within the process, including activity resources (applications, roles and performers, data), and time (duration and deadline constraints).

Although, the various constraints are inter-related, in this paper, we primarily focus on process level scheduling constraints. In typical process specifications, such constraints are specified using rigid control flow

[1] This work is partly supported by the Australian Research Council funded Project DP0558854

Copyright © 2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

dependencies. One such specification approach is introduced in section 2. Although such approaches have had significant success for a large class of processes due to their intuitive and visual appeal, their appropriateness is debatable for processes that require much greater flexibility in execution. As an example, consider the following scenario:

In a Telco servicing organization, customer requests are received through a web portal. Requests are then assigned to supervising engineers. These supervising engineers are considered domain experts capable of diagnosing service requests and preparing a customized *service plan*. The service plan essentially consists of several diagnostic tests and subsequently one or more actions. This service plan is then executed and results of the services rendered are compiled into a service report and logged into the system. Actual execution of the service plan may be long duration and involve delegation to several field workers.

In this scenario, consider specifically the task that prepares the service plan. Suppose that a number of diagnostic tests, (say 5 tests, T1, T2, … T5), are available. Any number of these tests can be prescribed for a given request, and in a given order. The supervising engineer has the flexibility to design a plan that best suits the customer request. However, there are certain restrictions on the scheduling of these tests. For example, T4 and T5 must be performed at the same time, and T2 must always be performed before T3. Providing a complete specification of all valid configurations of these tests is clearly not feasible, but would be necessary in typical control flow based graphical languages.

In this paper we target the modelling and execution of processes which have requirements as identified in the above scenario. We propose a framework which **firstly** allows scheduling constraints to be captured through a temporal constraint network. Temporal Constraint Networks (TCN) have been widely studied (Allen 1983; Vilain, Kautz et al. 1989; van Beek 1990; Dechter, Merir et al. 1991; van Beek 1992; Meiri 1995; Nebel and Buckert 1995; Drakengren and Jonsson 1996). Essentially TCNs are defined through 13 interval relations (Allen 1983) describing the relative positions between each pair of objects, including *before, meets, during, overlaps, starts, finishes*, the inverse of these relations *after, met-by, contains, started-by, finished-by* and a special relation *equals*. Temporal knowledge of multiple time intervals can be expressed by these relations and reasoned about in such a TCN. Using well established results from literature, we will present a discussion on the properties of such networks, showing that they not only provide a highly expressible and succinct specification to meet advanced requirements as described above, but also viable reasoning techniques for determining network consistency (i.e. ensuring executable processes). We will cover this discussion in section 3.

The proposed framework **secondly** also provides an execution environment in which individual instances can be customized according to specific needs, but still conform to process constraints. Instance customization is offered in an intuitive graphical language, whereas analysis on the correctness of the instance template is provided through TCN reasoning. In section 4, we will deliberate on how this is achieved, by illustrating the concepts through the above scenario. A review of related literature is provided in section 5, and finally conclusions and potential extensions are presented in section 6.

2 Background Concepts

In this section we provide essential concepts necessary for subsequent discussion. These concepts relate to typical business process modelling and execution, constraint satisfaction in general and temporal constraint networks in specific.

A substantial segment of the BPM space endorses the use of graphical models due to their intuitive and visual appeal (see e.g. (van der Aalst 1996; Coalition 1998; WfMC 1998; Sadiq and Orlowska 1999; Sadiq and Orlowska 2000)). An example of such a modelling language is given in figure 1.

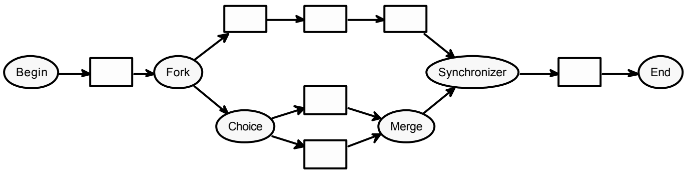


Figure 1. Graphical Modelling Language

The language consists of basic constructs such as sequence, fork, choice etc. Further details of this language can be found in (Sadiq and Orlowska 1999). We will use this simple notation to illustrate various examples in this paper. For example, figure 2 provides three acceptable process models for the telco scenario.

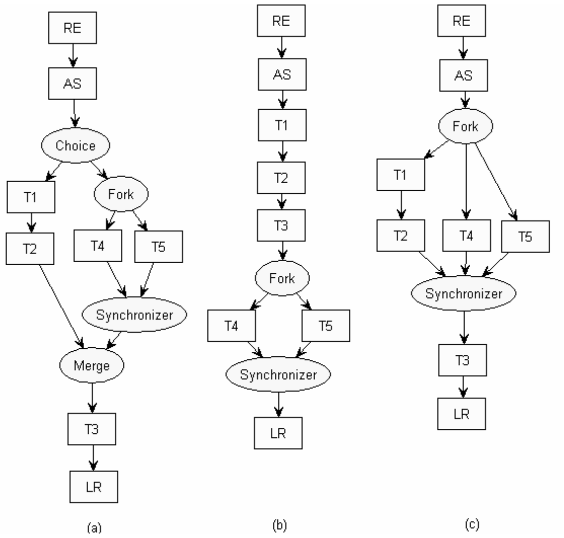


Figure 2: Process Models for Telco Scenario

Tasks RE, AS, T1, T2, T3, T4, T5, LR in figure 2 correspond to the following process activities in the scenario (respectively) - customer Request Enter, Assess Situation and preparation of service plan by supervisor engineer, Test 1 to Test 5, and finally Logging service Report.

Constraint satisfaction is a well known problem solving approach where a problem is formulated as a constraint satisfaction problem (CSP) andsearching for solutions and reasoning about some hypotheses in a restricted domain of knowledge can be performed. The process of problem formulation is called *constraint modelling* and the process of knowledge reasoning and solution searching is called *constraint processing*. The problem to be solved is modelled and represented in a *constraint network* N, where N *is* a triple $< X, D, C >$[1]. X is a finite set of *variables* $X = \{X_1, X_2, \ldots X_n\}$ with respective *domains* $D = \{D_1, D_2, \ldots D_n\}$, which contain the possible values for each variable. C is a set of *constraints* $C = \{C_1, C_2, \ldots C_t\}$ where each constraint C_i is a relation that imposes a limitation on the values a variable, or a combination of variables may be assigned to. A constraint can be specified on single variable (unary constraint), or on a pair of variables (binary constraint). (Jeavons 1999; Dechter 2003). However, practical CSPs with higher order constraints are generally NP-complete (Cook and Mitchell 1997), since modelling of real world problems often requires a large number of variables with large domains. e.g. to determine the satisfiability of a formula containing three literals (3-SAT problem) is NP-complete.

A simple CSP can be given as finding an assignment of values from domain $\{1, 2, 3\}$ to variables x, y and z such that x > y and y > z hold, where $X=\{x, y, z\}$, $D = \{D_x, D_y, D_z\}$, $D_x = D_y = D_z = \{1, 2, 3\}$ and $C=\{C_{xy}, C_{yz}\}$, $C_{xy} = x > y$, $C_{yz} = y > z$.

Figure 3 shows the constraint graph of this problem. A vertex in the constraint graph represents a variable and the arc between two vertices represents the constraint between the two variables.

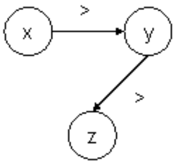
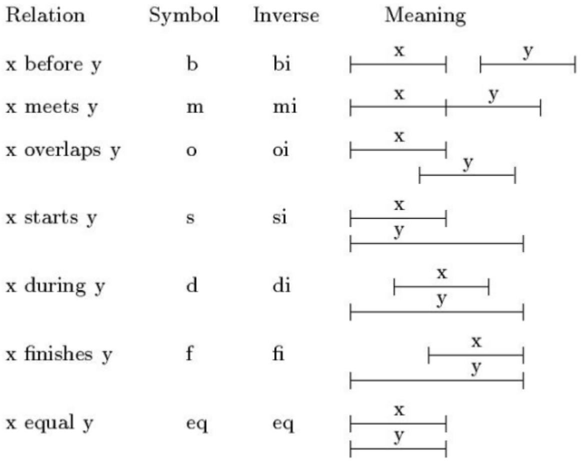


Figure 3: Constraint Graph

A solution of CPS is an assignment of a single value from its domain to each variable such that no constraint is violated. A problem may have one, many or no solution. A problem that has one or more solutions is *satisfiable* or *consistent*. The only possible solution to the previous example is x = 3, y = 2, z =1. Constraint satisfaction in general is a well-studied area and many techniques are available for reasoning and solving different class of CSPs.

The requirements to represent and reason about scheduling constraints in business processes necessitate a formal framework to capture temporal relations between process activities. Such temporal information is often incomplete and indefinite. (Allen 1983) has proposed a framework, called Interval Algebra (**IA**) network, for representing and reasoning about such information.



Relation	Symbol	Inverse	Meaning
x before y	b	bi	
x meets y	m	mi	
x overlaps y	o	oi	
x starts y	s	si	
x during y	d	di	
x finishes y	f	fi	
x equal y	eq	eq	

Figure 4: Basic Interval Relations (Allen 1983)

In (Allen 1983), 13 basic relations are given (see figure 4), which can hold between two intervals. In order to represent indefinite information, the relations between two intervals are allowed to be a disjunction of the basic relations. For example, the relation {*before, meets*} between intervals x and y restricts that x either finishes before y starts or x finishes immediately before y.

A restricted class of IA networks (Vilain, Kautz et al. 1989), denoted **SA**, can be translated into the Point Algebra framework, called Point Algebra (**PA**) network in polynomial time without loss of information. A point algebra network is a network of binary relations where the variables represent time points, and the binary relations between variables are disjunctions of the basic point relations $\{<, =, >\}$. In SA networks, the allowed relations between two intervals are the subsets of IA that can be represented using the relations $\{<, =, >\}$ into conjunctions of relation between the endpoints (start and finish points) of the intervals (van Beek 1990). For example, let the (T_1^-, T_1^+) and (T_2^-, T_2^+) denote the start and finish points of interval T_1 and T_2 respectively, an IA relation $T_1\{meets\}T_2$ can be translated into SA as $(T_1^- < T_2^-) \wedge (T_1^- < T_2^+) \wedge (T_1^+ = T_2^-) \wedge (T_1^+ < T_2^+)$. The complete description of SA can be found in (van Beek 1990).

These concepts have been utilized in temporal constraint networks (TCN). A temporal constraint network is a subclass of constraint networks where the representations of temporal information can be viewed as binary constraint networks and constraint satisfaction techniques can be used to reason about the temporal information. The variables in TCNs represent time intervals and constraints represent sets of allowed temporal relations between them. A solution or *consistent instantiation* of the network is an instantiation of the variables such that all the constraints between the variables are satisfied (van Beek 1992; Dechter 2003)

There are two fundamental reasoning problems in temporal constraint networks (van Beek 1992; Nebel and Buckert 1995; Dechter 2003). Given a temporal constraint network *N*,

[1] To be more precise, we can define $N = < X, D, d, C >$, where d is a function $d : x \rightarrow 2^D$ which maps a variable to some value in the corresponding domain.

- decide whether there exist a consistent assignment of values to all variables such that no constraint is violated, also known as the SAT problem, and
- find the minimal network of *N*.

The first problem is to reason about whether the set of temporal relations modelled in *N* is *valid* by determining whether the given temporal information is consistent, that is, whether it is possible to find a scenario where the intervals can be arranged along the time line according to the given information.

The second problem is to find (if the information is consistent) the feasible relations between all pairs of intervals, that is find one, some or all arrangements of the intervals along the time line, each corresponding to a possible scenario.

The major advantage of the constraint satisfaction approach to solve process modelling problems is that all a process designer has to do is to provide an appropriate formulation of the CSP. Well established techniques from constraint processing can be utilized to determine network consistency, as well as to find solutions. As such, we apply temporal constraints tomodelling scheduling requirements for ordering of process activities in business processes in a constraint network. In the following section, we will provide formal specifications to such a framework.

3 Business Process Constraint Network

Informally, we consider the business process as a set of tasks, where a task is either an atomic activity (a unit of work to be done) or a sub-process that contains one or more activities. The ordering of the tasks is specified by the temporal relations between the tasks.

3.1 Definition of BPCN

We follow Allen's IA network approach (Allen 1983) to represent and reason about temporal information of business process. A task *T* in a business process is modelled as a time interval, which is an ordered pair (T^-, T^+) such that $T^- < T^+$, where T^- and T^+ are interpreted as points on the time line. In particular, T^- is the point of time when task *T* starts execution, and T^+ is the point of time when *T* finishes execution. Henceforth, we refer to a task *T* as a time interval interpreted by the endpoints T^- and T^+.

The scheduling constraints between the tasks constituting a business process can be expressed by some combination of the 13 pair-wise interval relations (figurer 4) where each relation can be defined in terms of endpoint relations.

One or more relations can be defined on each pair of tasks. If more than one relation is defined on the same pair of tasks, we take the disjunction of the relations, which requires at least one relation must hold for all instances of the process. These relations describe a *partial order* of the tasks while a *total order* can be given if we assign exactly one relation between each pair of tasks. A process instance is a *totally ordered* instance if for every pair of tasks in the process either one of the 13 relations holds. The characteristic of partial order relations corresponds to the uncertain relationship between tasks, which allows for a large number of possibilities in which execution of tasks can be ordered according to different instances of a process.

Given a set of interval relations defined on the tasks of a business process, we can determine through logical inference whether a satisfiable ordering of task can be constructed. We define a Business Process Constraint Network (BPCN) based on a temporal constraint network adapted to represent scheduling constraints between tasks in the business process.

More specifically, a **BPCN** is a temporal constraint network $N = <X, D, C>$ where the set of variables $X = \{T_1, \ldots, T_n\}$ is the set of all tasks in the business process represented as time intervals, the set of domains $D = \{D_1, \ldots, D_n\}$ is the set of ordered pairs of discrete time values $\{(s, e) \mid s < e\}$, representing the start (s) and end (e) points of the corresponding task intervals. Binary constraints between pairs of interval variables are given as IA relations. The constraint C_{ij} between task T_i and T_j is defined as $C_{ij} \subseteq R = \{b, bi, m, mi, o, oi, s, si, f, fi, eq\}$, which describes the allowed relative locations of paired tasks in the discrete time line. A subset of basic relations corresponds to an ambiguous, disjunctive relationship between intervals. As a result, the set of constraints C in a BPCN defines the partial-order of the process execution model.

A *solution* to a BPCN is an assignment of a pair of values to each variable such that no constraint is violated. A solution can be established by assigning a single relation to each pair of tasks that is consistent with the constraint definition. A solution defines a total order of the process execution model.

3.2 Consistent BPCN

Consistency is used to describe the quality of the constraints defined in the constraint network. If conflicts exist between the constraints, or the inferred constraints, then we can conclude that the constraint network is inconsistent and hence no solution exists. The problem of determining whether a given BPCN is consistent can be mapped to the SAT problem.

It is desirable that given a BPCN, one can derive multiple process instances to suit different process requirements. Thus, we need to make sure that at least one satisfiable instance can be found, i.e. to determine the given BPCN is consistent (satisfiable).

Since the set of 13 interval relations are totally disjoint, and in Allen's IA network we allow multiple relations defined on the same pair of variables, conflicts of constraints in the BPCN can only exist between different pairs of variables. For example, we have a network of three variables, $X = \{T1, T2, T3\}$ and the constraints $C = \{C_{12}, C_{13}, C_{23}\}$, where C_{12}= T1{*b*, *m*}T2, C_{13}= T1{*s*, *eq*} T3 and C_{23} = T2{*b*, *m*} T3. The definition for each C_{ij} does not cause conflicts since for a particular process instance we only require one relation to hold for each pair of variables, i.e. T1{*b*}T2. However, the network is inconsistent since from C_{12} and C_{23} we can infer C'_{13} = T1{*b*}T3, but $C'_{13} \cap C_{13} = \emptyset$, which means we cannot find a scenario that satisfies C_{12}, C_{23} and C_{13} at the same time. Hence the network is inconsistent.

The technique to determine consistency for BPCN is based on enforcing local consistency on the network. Before defining local consistency, we first give a formal definition for a consistent BPCN.

A BPCN $N = < X, D, C >$ is said to be *consistent* if we can find a consistent scenario of network *N*. A network N' is a consistent scenario of a network *N* if and only if (iff):

1. there exists exactly one relation between each pair of variables (T_i, T_j) in N', namely, $|R'_{ij}| = 1$; and
2. every such relation R'_{ij} in N' is a subset of the relation between the same pair of variables in *N*, namely, $R'_{ij} \subseteq R_{ij}$; and
3. there exists a consistent instantiation of N'.

We assume that each variables have sufficient large domains, as such when condition 1 and 2 hold, we can determine N' is a consistent scenario of *N*. To find a consistent scenario we simply search through the different possible *Ns* that satisfy conditions 1 and 2 (van Beek 1992).

Given a BPCN *N* where the set of relations *C* is restricted by SA subclass, to determine the consistency of *N* only requires to determine whether *N* is *path-consistent* (Vilain, Kautz et al. 1989; van Beek 1990). To define path-consistency in BPCN, we need to define the following operations.

IA describes all possible relations between two intervals, as such the universal relation between two intervals (which means there is no constraint defined on them) is the set of basic relations *R*.

Being part of Allen's IA, the inverse, intersection and composition operations on pairs of variables are also defined, which are given as follows (Allen 1983; Dechter 2003):

The *inverse* $R^{\cup}$ of the relation *R* is the relation $R^{\cup} = \{(a,b) \mid (b,a) \in R\}$.

The *intersection* of two IA relations R' and R'', denoted by $R' \cap R''$, is the set theoretic intersection of R' and R''. For example, given $R' = \{o, s, f, m\}$ and $R'' = \{s, f, d\}$, $R' \cap R'' = \{s, f\}$.

The *composition* of two basic IA relations r' and r'', denoted by $r' \otimes r''$, is defined by the transitive table (see figure 5). For example, the basic relations T1 *meets* T2, T2 *before* T3 induce a new (single or composite) relation T1 *before* T3. The composition of two composite relations R' and R'', denoted by $R' \otimes R''$, is the composition of the constituent basic relations:

$$R' \otimes R'' = \{r' \otimes r'' \mid r' \in R', r'' \in R''\}$$

r'' r'	**b**	**s**	**d**	**o**	**m**
b	b	b	b o m d s	b	b
s	b	s	d	b o m	b
d	b	d	d	b o m d s	b
o	b	o	o d s	b o m	b
m	b	m	o d s	b	b

Figure 5: Portion of the transitivity table defined by (Allen 1983)

A binary constraint C_{ij} is path-consistent relative to a variable T_k iff $(C_{ij} \cap (C_{ik} \otimes C_{kj})) \neq \emptyset$. A BPCN is a path-consistent BPCN iff for every relation R_{ij} (including universal relations) and for every $k \neq i, j$, R_{ij} is path-consistent relative to T_k.

Validation of the constraint definition on a BPCN can be achieved by applying a generic path-consistency algorithm.

```
Input: An IA network N
Output: A path-consistent IA network
for k:= 1 to n do
    for i,j:=1 to n do begin
        $C_{ij} \leftarrow C_{ij} \cap (C_{ik} \otimes C_{kj})$
        if $C_{ij} = \emptyset$ then break
```

Figure 6: Path-Consistency Algorithm (Dechter 2003)

We repeatedly apply the above algorithm to the network *N* until no further changes can be made to the current constraints or some constraint becomes empty, indicating inconsistency. The operation $C_{ij} \leftarrow C_{ij} \cap (C_{ik} \otimes C_{kj})$ applied to each constraint is called the relaxation operation (Dechter 2003). For some class of IA relations, this algorithm is guaranteed to determine consistency in $O(n^3)$ time, where *n* is the number of intervals in *N*. Further discussions can be found in section 4.

4 Execution Framework

As explained in the previous section, process definition consists of a pool of activities and a small number of constraints defined on those activities. However, the process instances are allowed to follow a very large number of execution paths. As long as the given constraints are met, any execution path dynamically constructed at runtime is considered legal. This ensures flexible execution while maintaining a desired level of control through the specified constraints. The key feature and strength of the approach is to use the power of constraints, while still preserving the advantages of graphical languages.

Below we explain the core functions of the process management system based on the concepts presented above. The discussion is presented as a series of steps in the specification and deployment of the Telco example process. Figure 7 provides an overview diagram of these steps and associated functions.

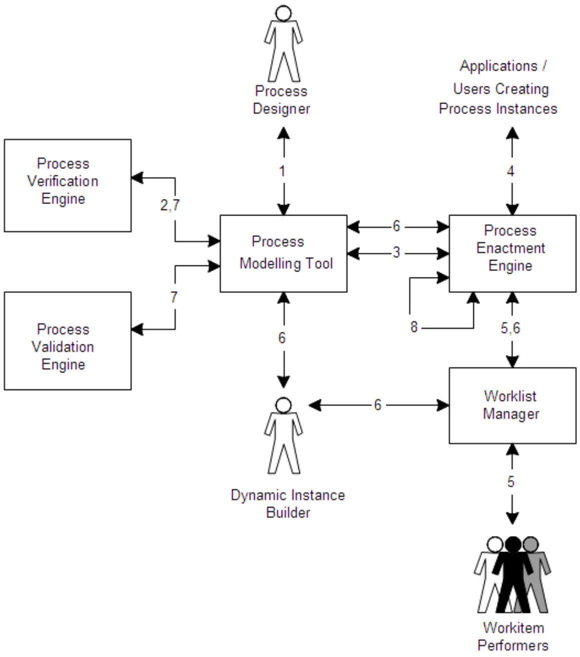


Figure 7: Framework Overview

- **Step 1**: The definition of the (flexible) process model takes place. The pool of activities and associated constraints are defined.

We specify the telco business process using BPCN. The process is represented as $N = <X, D, C>$. There are 8 tasks in this business process, X = {*RE, AS, T1, T2, T3, T4, T5, LR*}, which correspond to customer request, situation assessment by Supervisor Engineer, test 1 to test 5, and logging service report, respectively.

The set of constraints C is defined according to the scheduling requirements. Consider the following restrictions on the scheduling of the tests: Test 1 must start before test 2 starts. If both tests do not solve the problem then further test 3 is ordered. Test 3 must not start before test 2 finishes. Test 4 and test 5 must start at the same time. Besides, no tests can start before Supervisor Engineer starts assessing the report, and no tests can be started after service report is logged. One can define the set of scheduling constraints C = $\{C_{RE\text{-}AS}, C_{AS\text{-}T1}, C_{AS\text{-}T4}, C_{T1\text{-}T2}, C_{T2\text{-}T3}, C_{T4\text{-}T5}, C_{AS\text{-}LR},, C_{T2\text{-}LR}, C_{T3\text{-}LR}, C_{T5\text{-}LR}\}$, where

$C_{RE\text{-}AS}$= RE{*b, m, o*}AS

$C_{AS\text{-}T1}$= AS{*b, m, o*}T1

$C_{AS\text{-}T4}$= AS{*b, m, o*}T4

$C_{T1\text{-}T2}$= T1 {*b,m,o,di,fi*}T2

$C_{T2\text{-}T3}$ = T2{*b, m*}T3

$C_{T4\text{-}T5}$ = T4{*s, si, eq*}T5

$C_{AS\text{-}LR}$ = AS{*b, m*}LR

$C_{T2\text{-}LR}$ = T2 {*b, m*}LR

$C_{T3\text{-}LR}$ = T3 {*b, m*}LR

$C_{T5\text{-}LR}$ = T5 {*b, m*}LR

For example, constraint $C_{T2\text{-}T3}$ = T2{*b, m*}T3 defines a precedence order on the executions of T2 and T3, which requires T2 must finish execution before or meets T3 (Figure 8).

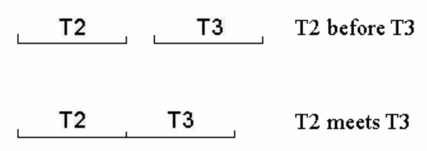


Figure 8: Valid relations between T2 and T3

Figure 9 shows some of the **many** valid orders of execution for T2 and T3 in some instance templates based on graphical model. The unnamed tasks represent any valid tasks in the process.

On the graphical model, relations {*o, oi, s, si, d, di, f, fi, eq*} correspond to concurrent execution pattern, i.e., there is no path between two tasks. Relations {*b, bi, m, mi*} correspond to either concurrent or serial execution pattern. The interval relation between two tasks in concurrent execution pattern requires consideration of the execution duration of the tasks, i.e. for each task, its estimated maximum duration must be provided. T2 and T3 in figure9 (a) (b) (c) execute in serial, while in figure9 (d) execute in concurrent threads of control.

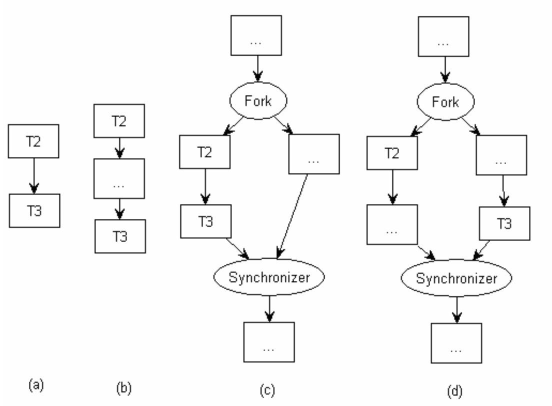


Figure 9: Valid execution order of T2 and T3. (a)(c) and (d) correspond to T2{*meets*}T3, (b) corresponds to T2{*before*}T3

If no constraint is defined on a pair of tasks, then universal constraint applies, which means any 13 relations can be assigned to this pair of tasks.

Figure 10 shows the constraint graph of the BPCN network *N*.

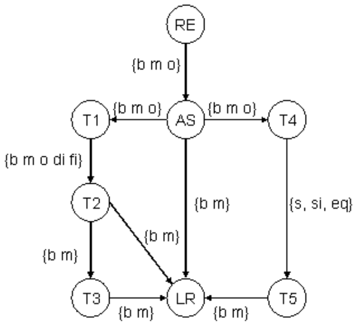


Figure 10: Constraint graph of *N*

- **Step 2**: The process definition is verified for structural errors (Sadiq and Orlowska 2000). The validation of the given constraint set may take place at this time.

This step is to determine whether the BPCN is consistent. We apply the algorithm shown in figure 6 to the BPCN, the resulting consistent network is shown in figure 11.

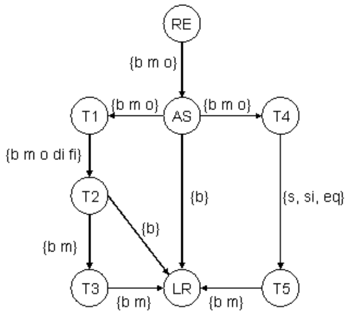


Figure 11 Path-Consistent Network of *N*

In the original network, T2{*m*}LR is not consistent with respect to C_{T2-T3}. and C_{T3-LR} since from T2{*b*,*m*}T3 and T3{*b*,*m*}LR we can infer through transitive table T2{*b*}LR but not T2{*m*}LR. Thus T2{*m*}LR has been deleted from C_{T2-LR}. Similarly AS{*m*}LR has been deleted from C_{AS-LR}.

It is important to point out that the choice of constraints that will be removed as a result of conflicts is a design issue. The framework will only identify which constraints have conflicts. Process designers then have to make a decision based on process requirements, as to which constraint can be removed.

- **Step 3**: The definition created above is uploaded to the process engine. This process model is now ready for deployment.

- **Step 4**: For each case of the process model,the user or application would create an instance of the process model. On instantiation, the engine creates a copy of the process definition and stores it as an instance template. This process instance is now ready for execution.

- **Step 5**: The available process activities of the newly created instance are assigned to performers through work lists and activity execution takes place as usual, until the instance needs to be dynamically adapted to particular requirements arising at runtime (as shown in next step).

- **Step 6**: The knowledge worker or expert user, shown as the dynamic instance builder, will invoke a special build function, and undertake the task of dynamically adapting the instance template with available pool of activities, while guided by the specified constraint set. This revises the instance template. The build function is thus the key feature of this approach and requires the capability to load and revise instance templates for active instances.

An instance template is a particular customization of a given instance to suit runtime requirements, e.g. a particular configuration of tests prescribed by a service plan. Instance templates define total order of task execution. Figure 2(b)(c) are examples of valid instance templates for the Telco scenario.

- **Step 7**: The next step is to validate the new template, to ensure that it conforms to the correctness properties of the language as well as the given constraints.

An instance template is a totally ordered process instance, where for each pair of tasks, there must be exactly one relation between them. The total ordering in the given template is defined by the assignment of values to the endpoints of each task instance and visualised in the graph-based model. The template validation service first translates the total ordering of task instances from graph-based model into the interval model and checks whether the given sequence of task execution conforms to the constraints defined in the network. The objective of the translation is to find out the implicit temporal relations between tasks and check against the process constraints defined as interval relations on the tasks.

Take the instance template shown in figure 2(c) as the instance template to be validated. Through the PA-IA translation table (van Beek 1990) given in figure 12, we can work out the interval relation between each pair of tasks (as shown in figure 13). Then we check for each translated relation r'_{ij}. If r'_{ij} belongs to R_{ij} of the consistent relations given in figure 11, then r_{ij} is a valid relation between T_i and T_j. If r_{ij} is valid for all T_i and T_j in the instance template, then this template is a valid process instance according to *N* (a consistent scenario of *N*).

PA IA	$T_i^- T_j^-$	$T_i^- T_j^+$	$T_i^+ T_j^-$	$T_i^+ T_j^+$
{*eq*}	=	<	>	=
{*b*}	<	<	<	<
{*d*}	>	<	>	<
{*o*}	<	<	>	<
{*m*}	<	<	=	<
{*s*}	=	<	>	<
{*f*}	>	<	>	=
{*di*}	<	<	>	>
{*oi*}	>	<	>	>
(*fi*)	<	<	>	=

Figure 12: PA-IA translation for basic IA relations (van Beek 1992)

The validation procedure corresponds to determining whether a given network instance is a consistent scenario of the original network, as discussed in section 3.2.

PA IA	$T_i^- T_j^-$	$T_i^- T_j^+$	$T_i^+ T_j^-$	$T_i^+ T_j^+$	Translated Relation $R_{ij}^{'}$
RE-AS	$<$	$<$	$=$	$<$	$\{m\}$
AS-T1	$<$	$<$	$=$	$<$	$\{m\}$
AS-T4	$<$	$<$	$<$	$<$	$\{b\}$
T1-T2	$<$	$<$	$=$	$<$	$\{m\}$
T2-T3	$<$	$<$	$=$	$<$	$\{m\}$
T4-T5	$=$	$<$	$>$	$=$	$\{eq\}$
AS-LR	$<$	$<$	$<$	$<$	$\{b\}$
T2-LR	$<$	$<$	$<$	$<$	$\{b\}$
T3-LR	$<$	$<$	$=$	$<$	$\{b\}$
T5-LR	$<$	$<$	$<$	$<$	$\{b\}$

Figure 13: PA-IA translation for the given instance template

Since every translated relation r_{ij} in the given instance template belongs to R_{ij}, we can determine that the instance template is valid. We can say that the instance template as a totally ordered process instance is consistent with regard to the constraint definition in *N*.

- **Step 8**: On satisfactory validation results the newly defined (or revised) instance template resumes execution. Execution will now continue as normal, until completion or until re-invocation of the build function, in which case steps 6-8 will be performed again.

Discussion

The execution framework presented above is based on the fact that consistency of BPCN can be determined by algorithm shown in figure 6, and the translations between IA and PA are made possible without loss of information.

It has been shown (Allen 1983; Vilain, Kautz et al. 1989) that the algorithm applied in step 2 is sound but incomplete for the full IA network. It is sound because it does not introduce invalid relations to the network. It is incomplete because in some cases consistency cannot be determined. Determining the minimal network for some class of IA network is known to be NP-complete. However, in the execution framework, we only consider the *SA*, a subclass of *IA* network that can be translated into *PA*, because we also need such translations when verifying instance templates given in graphical process models. *SA* networks are tractable (Vilain, Kautz et al. 1989; Schwalb and Vila 1998), where enforcing path-consistency correctly decides consistency of the network (SAT) in $O(n^3)$ where *n* is the number of intervals.

Besides, many tractable subclasses of IA network have been identified, including pointisable IA by (Vilain, Kautz et al. 1989; van Beek 1990), tractable subclass of the Point-Interval algebra by (Drakengren and Jonsson 1996; Jonsson, Drakengren et al. 1996), ORD-Horn subclass of IA (Nebel and Buckert 1995) and Interval-point algebra (IPA) network by (Meiri 1995). A complete classification of tractability in Allen's Algebra has been given by (Drakengren and Jonsson 1997). On the other hand, approximation can also be made to express intractable subclass of relations by the tractable counterpart(van Beek 1989).

Furthermore, it is also worthwhile to consider how many relations are sufficient to describe scheduling relations for certain classes of business processes. If no constraint is specified on a BPCN, we permit any combinations of ordering of the tasks in any process instance. If too many constraints are specified, the constraint network is too rigid, and corresponds to the over-constraint problem (Beaumont, Sattar et al. 2001). The minimal requirement is that any constraint definition should permit at least one consistent scenario (to make constraint network consistent). The practical requirement however is that the constraint definition should permit a *large* number of consistent scenarios. This is the case when the full potential of the proposed framework will be realized (as illustrated previously in this section).

5 Related Work

Constraints have been incorporated with business process modelling. (Crampton 2004) identifies a generic class of constraints, called entailment constraints, which restrict the execution order of process tasks with respect to authorisation. e.g."Task 2 must be performed by a role that is more senior than the role that performed task 1"(Crampton 2004).

In (Tsang 2003), constraint satisfaction in business process modelling is aligned with Distributed Constraint Satisfaction (DCSP), a branch of CSP in a collaborative agent environment. An additional set of constraints, *E*, called open constraints is used to capture external and uncertain information. Many studies on DCSP are available in literature (Yokoo 2001).

Planning and scheduling are major applications in constraint satisfaction. (Barták 1999) provides a classification for resource allocation and temporal constraints, as well as dynamic models for reasoning about such information. Particularly, time is modelled in either discrete model or event-based model. In discrete time model, the timeline is divided into a sequence of discrete time intervals with some duration. The variables are time intervals describing durations of process activities. This model is applicable for modelling processes where time intervals represent individual tasks. The event-based model only capture time points when change takes place. This model associate mostly with resources constraints. In our execution framework, the temporal information is modelled as discrete time intervals.

Temporal reasoning techniques have been applied to business process modelling, mostly to capture relative and absolute deadline constraints (Marjanovic and Orlowska 1999; Marjanovic 2000; Li and Yang 2004; Eder, Panagos et al. 1999; Eder, Gruber et al. 2000; Combi and Pozzi 2003). In particular, (Marjanovic 2000) provides a dynamic verification algorithm for absolute and relative deadline constraints in workflow, where the algorithm is based on execution durations represented by metric points. (Combi and Pozzi 2003) describe absolute and relative deadline constraints based on endpoints of intervals, as well as some considerations to represent fork and merge operators. It is obvious that these constraints can be described by a small subset of *PA*, but there is no constraint validation algorithm given in this approach. (Eder, Panagos et al. 1999) present a timed workflow graph approach to express the upper and lower bound constraints of task execution, where the constraint semantics is based on the execution durations and relative deadlines of process tasks. (Bettini, Wang et al. 2002) present a quantitative temporal constraints model which supports multiple time granularity. In this model, constraints are defined on quantitative time points (i.e. seconds, hours), such time points are regarded as variables, and the constraints are defined as temporal distances. In some process modelling approaches, scheduling constraints are incorporated with resource allocation constraints, such as (Li and Yang 2004; Li and Yang 2004; Tan, Crampton et al. 2004; Tan, Crampton et al. 2004).

The distinctions between the framework proposed in the paper and the previous work can be made as follows: Firstly, we have shown that a large subset of full IA network, called the SA network can be used to represent temporal relations in business processes within the BPCN. Secondly, we have shown through a case study that using generic constraint propagation techniques (path-consistency algorithm shown in figure 6) is sufficient to provide validation for such information in BPCN. Last but not least, we have shown translations between graph-based process description to interval-based process description where the former enables intuitive model expression and the later provides a wealth of reasoning techniques.

6 Summary and Outlook

In summary, we have presented a framework that allows for flexible business process execution. The framework is based on the notion of process constraints, and in this paper a particular sub class of process constraints has been considered. In general, we see the level of definition of these constraints along a continuum of specification. There is the completely predefined model on one end, and the model with no predefinition on the other. Thus the former only has strong constraints (e.g. X and Y are activities of a given process, and Y must immediately follow X), and the latter no constraints at all. The former extreme is too prescriptive and not conducive to dynamic business environments; and the latter extreme defeats the purpose of process enforcement, i.e. with insufficient constraints, the process goals may be compromised and quality of service for the process cannot be guaranteed. Finding the exact level of specificity along this continuum will mostly be domain dependent. However, technology support must be offered at a generic level. This work has accordingly attempted to address the need to provide a modelling environment wherein the level of specification can be chosen by the process designer such that the right balance between flexibility and control can be achieved.

We see significant potential in expanding this framework to incorporate other classes of constraints, and especially to study the interplay between them. For example if two tasks X and Y have a scheduling constraint on them, defined by an overlap relation $X\{o\}Y$, and then a resource constraint is also defined on them, say by the *binding of duty* (Li and Yang 2004) relation (i.e. X and Y must be performed by the same resource), what impact does this have on the overall constraint network. There can be several interpretations of this problem, which need to be analysed to formulate workable solutions. However, the essence of the framework will still hold true, that is, a small number of constraints can potentially be specified to realize a very large number of valid instances at runtime.

Another interesting problem is to augment the template construction (see section 4) with an intelligent search function for *best* template. This requires at a minimum a facility to build an objective function into the *BPCN* and furthermore a facility to search the solution space of the *BPCN* for solutions meeting the objective. Example of such an objective function can be minimum time span of part or whole of the process, minimum consumption of a given resource, maximal concurrent execution of process activities etc. Such a service could greatly enhance the productivity of the knowledge worker who is dynamically building the template, by not only allowing them to incorporate domain experience in to the template construction, but also providing guidelines on best practice.

7 References

Allen, J. F. (1983): Maintaining knowledge about temporal intervals. *Communications of the ACM* **26**: 832 - 843.

Barták, R. (1999): Dynamic Constraint Models for Planning and Scheduling Problems. *Lecture Notes In Computer Science*; Vol. 1865, Selected papers from the Joint ERCIM/Compulog Net Workshop on New Trends in Constraints, Springer-Verlag London, UK**:** 237 - 255.

Beaumont, M., A. Sattar, et al. (2001): Solving Overconstrained Temporal Reasoning Problems. *14th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, Springer-Verlag.

Bettini, C., X. S. Wang, et al. (2002): Temporal Reasoning in Workflow Systems. *Distributed and Parallel Databases* **11**(3 (May 2002)): 269 - 307\6.

Combi, C. and G. Pozzi (2003): Temporal Conceptual Modelling of Workflows. *Lecture Notes in Computer Science*, Springer-Verlag. **2813:** 59 - 76.

Cook, S. A. and D. G. Mitchell (1997): Finding Hard Instances of the Satisfiability Problem: A Survey. *The DIMACS Workshop on Satisfiability Problems*, American Mathematical Society.

Crampton, J. (2004): On the satisfiability of authorization constraints in workflow systems. RHUL--MA--2004--1, Department of Mathematics, Royal Holloway, University of London.

Dechter, R. (2003): *Constraint Processing*, Morgan Kaufmann Publishers.

Dechter, R., I. Merir, et al. (1991): Temporal Constraint Networks. *Artificial Intelligence* **49**: 61 - 95.

Drakengren, T. and P. Jonsson (1996): Maximal Tractable Subclasses of Allen's Interval Algebra: Preliminary Report. AAAI/IAAI. **Vol. 1:** 389-394.

Drakengren, T. and P. Jonsson (1997): Towards a Complete Classification of Tractability in Allen's Algebra. *IJCAI:* 1466-1475.

Eder, J., W. Gruber, et al. (2000): Temporal Modeling of Workflows with Conditional Execution Paths. *Database and Expert Systems Applications: 11th International Conference, DEXA 2000*, London, UK, Springer-Verlag GmbH.

Eder, J., E. Panagos, et al. (1999): Time Constraints in Workflow Systems. *Advanced Information Systems Engineering: 11th International Conference, CAiSE'99*, Heidelberg, Germany,, Springer-Verlag GmbH.

Jablonski, S. and C. Bussler (1996): *Workflow Management - Modeling Concepts, Architecture and Implementation*, International Thomson Computer Press.

Jeavons, P. (1999): Constructing Constraints. *The 4th International Conference on Principles and Practice of Constraint Programming*, Springer-Verlag.

Jonsson, P., T. Drakengren, et al. (1996): Tractable subclasses of the point-interval algebra: A complete classification. *KR'96*, AAAI Press/The MIT Press.

Li, H. and Y. Yang (2004): Dynamic checking of temporal constraints for concurrent workflow. *Electronic Commerce Research and Applications* **4**(2005): 124-142.

Li, H. and Y. Yang (2004): Verification of Temporal Constraints for Concurrent Workflows. *Advanced Web Technologies and Applications, 6th Asia-Pacific Web Conference, APWeb 2004*, Hangzhou, China, Springer.

Marjanovic, O. (2000): Dynamic verification of temporal constraints in production workflows. *Proceedings of 11th Australasian Database Conference, 2000. ADC 2000.*

Marjanovic, O. and M. E. Orlowska (1999): Dynamic Verification of Absolute and Relative Deadline Constraints in Production Workflows, Technical Report, No. 446, Department of Computer Science and Electrical Engineering, University of Queensland, Australia.

Meiri, I. (1995): "Combining qualitative and quantitative constraints in temporal reasoning." *Artificial Intelligence* **87**: 343 - 385.

Nebel, B. and H.-J. Buckert (1995): "Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra." *Journal of ACM* **42**(1): 43 - 66.

Sadiq, W. and M. E. Orlowska (1999): On Capturing Process Requirements of Workflow Based Business Information System. *3rd International Conference on Business Information Systems (BIS '99)*, Poznan, Poland, Springer-Verlag.

Sadiq, W. and M. E. Orlowska (2000): "Analyzing Process Models using Graph Reduction Techniques." *Information Systems* **25**(2): 117 - 134.

Schwalb, E. and L. Vila (1998): Temporal Constraints: A Survey. *Constraints* **3**(2 - 3): 129-149.

Tan, K., J. Crampton, et al. (2004): The consistency of task-based authorization constraints in workflow systems. *Proceedings of 17th IEEE Computer Security Foundations Workshop, 2004*.

Tsang, E. P. K. (2003): Constraint Satisfaction in Business Process Modelling, http://www.econ.uba.ar/servicios/publicaciones/journal7/tsang.htm, Accessed 2005.

van Beek, P. (1989): Approximation algorithms for temporal reasoning. *The 11th International Joint Conference on Artificial Intelligence*, Detroit, Mich, Morgan Kaufmann.

van Beek, P. (1990): Exact and Approximate Reasoning about Qualitative Temporal Relations. Dept. of Computer Science, University of Alberta, PhD Thesis.

van Beek, P. (1992): "Reasoning about Qualitative Temporal Information." *Artificial Intelligence* **58**: 297 - 326.

van der Aalst, W. M. P. (1996): Three Good reasons for Using a Petri-net-based Workflow Management System. *Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96):* S. Navathe and T. Wakayama. Camebridge, Massachusetts**:** 179 - 201.

Vilain, M., H. Kautz, et al. (1989): Constraint propagation algorithms for temporal reasoning: a revised report. *Readings in qualitative reasoning about physical systems*. San Francisco, CA, USA, Morgan Kaufmann Publishers Inc**:** 373--381.

WfMC (1998): Interface 1: Process Definition Interchange, Process Model, Workflow Management Coalition.

Yokoo, M. (2001): *Distributed Constraint Satisfaction*, Springer.

Discovering Task-Oriented Usage Pattern for Web Recommendation

Guandong Xu[1], Yanchun Zhang[1], Xiaofang Zhou[2]

[1]School of Computer Science and Mathematics
Victoria University, PO Box 14428, VIC 8001, Australia
[2] School of Information Technology & Electrical Engineering
University of Queensland, Brisbane QLD 4072, Australia

{xu,yzhang}@csm.vu.edu.au
zxf@itee.uq.edu.au

Abstract

Web transaction data usually convey user task-oriented behaviour pattern. Web usage mining technique is able to capture such informative knowledge about user task pattern from usage data. With the discovered usage pattern information, it is possible to recommend Web user more preferred content or customized presentation according to the derived task preference. In this paper, we propose a Web recommendation framework based on discovering task-oriented usage pattern with *Probabilistic Latent Semantic Analysis* (PLSA) model. The user intended tasks are characterized by the latent factors through probabilistic inference, to represent the user navigational interests. Moreover, the active user's intuitive task-oriented preference is quantized by the probabilities, by which pages visited in current user session are associated with various tasks as well. Combining the identified task preference of current user with the discovered usage-based Web page categories, we can present user more potentially interested or preferred Web content. The preliminary experiments performed on real world data sets demonstrate the usability and effectiveness of the proposed approach.

Keywords: Task-Oriented Usage Pattern, Web Usage mining, Web Recommendation.

1 Introduction

With the rapid development of a variety of Internet applications, Web has recently become not only a powerful platform and tool for retrieving information, but also a large repository for discovering knowledge. However, how to find needed and related information from the Web is a big challenge that Web information search domain is facing. Among much work addressed to such so-called information overload problem, Web recommendation is one of the instrumental means to help users locate more preferred information. Basically, Web recommendation is considered as the process of identifying user's preference and adapting service to satisfy user's need based on referring the historical behaviour of current user or others who share similar interest to this user.

To-date, there are two kinds of approaches and techniques commonly used in Web recommendation, namely content-based filtering agents and collaborative filtering systems (Dunja 1996; Herlocker, Konstan et al. 2004). Content-based filtering systems, such as WebWatcher (Joachims, Freitag et al. 1997) and client-side agent Letizia (Lieberman 1995), generally generate recommendation based on the pre-constructed user profiles by measuring the similarity of Web content to these profiles. In contrast, Collaborative filtering systems make recommendation by utilizing the rating of current user for objects via referring other users' preference that is closely similar to current one. Since collaborative filtering technique refers common interest of user group instead of individual's and is capable of presenting more preferable content to users, it has recently been widely adopted in Web recommendation applications and have achieved great success as well (Shardanand and Maes 1995; Konstan, Miller et al. 1997; Herlocker, KONSTAN et al. 1999). In addition, Web usage mining (WUM) has been proposed as an alternative method for not only revealing user access pattern, but also making Web recommendation in recent year (Mobasher, Dai et al. 2002). WUM is an application of data mining to discover usage pattern from Web log files and identify the underlying user functional interests that lead to common navigational activity, and has become an active topic of research and commercialization. Existing WUM techniques, which are well studied and developed in data mining domain, include collaborative filtering based on the k-Nearest Neighbor algorithm (*kNN*) (Shardanand and Maes 1995; Konstan, Miller et al. 1997; Herlocker, KONSTAN et al. 1999), Web clustering (Han, Karypis et al. 1998; Perkowitz and Etzioni 1998; Mobasher, Dai et al. 2002), association rule mining (Agrawal and Srikant 1994; Agarwal, Aggarwal et al. 1999) and sequential pattern mining technique (Agrawal and Srikant 1995). Amongst these methods, Web clustering is an important topic that engages in clustering not only Web users but also pages - discovering clusters of users that exhibit similar access pattern and categories of pages that share close functionality to users. By making use of the discovered knowledge from user clusters or page categories, Web designer may understand the users better, capture the unobservable relationships among pages from user's view ponit deeply, thus, can improve Web structure design and provide more preferable and customized service to the users.

In our previous work (Xu, Zhang et al. 2004; Xu, Zhang et al. 2005), Web user clustering and page grouping

Copyright (c) 2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

techniques are well investigated to reveal such informative knowledge with regard to user behaviour and page functionality based on mining usage data. Especially, a so-called *Probabilistic Latent Semantic Analysis* (PLSA) model is proposed to address the topic of Web clustering. Different from other existing *Latent Semantic Analysis* (LSA) method, PLSA is to capture not only the underlying relationships among Web users as well as pages, but also reveal the hidden task-oriented pattern derived form WUM with probability inference approach. The main idea of this paper is to extend the above work to Web recommendation by identifying user task-oriented access pattern and integrating usage-based Web page category into Web recommendation process to improve the efficiency of recommendation. Moreover, approaches based on PLSA has been successfully applied in collaborative filtering (Hofmann 2004) Web usage mining (Jin, Zhou et al. 2004; Xu, Zhang et al. 2004; Xu, Zhang et al. 2005), text learning and mining (Cohn and Chang 2000; Hofmann 2001), co-citation analysis (Cohn and Hofmann 2001; Hofmann 2001) and related topics.

In this paper, we propose a Web recommendation framework based on discovering task-oriented usage pattern with PLSA model. The Web recommendation process exploits the usage pattern derived from Web usage mining to predict user preferred content or customized presentation. At data preparation stage, we collect Web transaction data from Web server log files, and construct user session collection and Web page corpus respectively. Conceptually, each user session can be expressed as a weighted Web page vector, in which the element reflects the relative significance contributed by the corresponding Web page in same user session. After integrating all user sessions, the Web access observation (i.e. usage data), on which the mining task is performed, is ultimately constructed in the form of page-based weight matrix. By employing PLSA model, we can not only characterize the underlying relationships among Web access observation but also identify the latent semantic factors that are considered to represent the navigational tasks of users during their browsing period. Such relationships are determined by the estimated probabilities, and then are utilized to discover the task-oriented usage patterns in the form of a dominant task sequence. Furthermore, we make use of this discovered knowledge of usage pattern for Web recommendation by combining the task-oriented usage pattern and Web page category into Web recommendation to predict the more potentially interested or preferred content to user.

The main contributions we have done in this work are described as follows: firstly, we present a Web usage mining and Web recommendation integrated framework based on PLSA model. Secondly, we investigate the discovery of user access pattern and latent factor related to these patterns via employing probability inference process, in turn, make use of the discovered usage knowledge for Web recommendation. Particularly, we develop an algorithm for identifying task-oriented usage pattern and predicting user potentially visited pages based on Bayesian updating approach and incorporating Web page category into Web recommendation. Finally, we demonstrate the usability and effectiveness of the proposed model by conducting experiments on two real world datasets.

The rest of the paper is organized as follows. In section 2, we introduce Web usage mining technique with PLSA model, especially we discuss how to identify user access session and achieve probability estimations via Expectation-Maximization (EM) algorithm. We present the algorithms for discovering Web page categories and identifying task-oriented access pattern in section 3. In section 4, we concentrate on how to develop Web recommendation framework upon the discovered usage knowledge. To validate the proposed approach, we conduct preliminary experiments on two real world datasets, present evaluation results in section 5, and conclude the paper and outline future work in section 6.

2 Web Usage Mining with PLSA

Web usage mining usually consists of three steps, i.e. data collection and pre-processing, pattern mining as well as knowledge application. As a result, Web recommendation is actually the ultimate stage of the Web usage mining, i.e. application stage. The overview of Web usage mining and Web recommendation is depicted in Figure 1.

2.1 Usage Data Model

Prior to introducing Web usage mining technique, we briefly discuss the issue with respect to construction of usage data. In general, the exhibited user access interests may be reflected by the varying degrees of visits on different Web pages during one session. Thus, we can represent a user session as a weighted page vector visited by the user during a period. In this paper, we use the following notations to model the co-occurrence activities of Web users and pages:

- $S = \{s_1, s_2, \cdots s_m\}$: a set of m user sessions.
- $P = \{p_1, p_2, \cdots p_n\}$: a set of n Web pages.
- For each user, the navigational session is represented as a sequence of visited pages with corresponding weights: $s_i = \{a_{i,1}, a_{i,2}, \cdots a_{i,n}\}$, where a_{ij} denotes the weight for page p_j visited in s_i user session. The corresponding weight is usually determined by the number of hit or the amount time spent on the specific page. Here, we use both of them to construct usage data from two real world data sets.
- $SP_{m\times n} = \{a_{i,j}\}$: the ultimate usage data in the form of weight matrix with dimensionality of $m \times n$.

Generally, the element in the session-page matrix, a_{ij}, is the normalized weight associated with the page p_j in the user session s_i, which is usually determined by the number of hit or the amount time spent on the specific page. The session normalization is able to capture the relative significance of a page within one user session with respect to others pages accessed by same user. For example, Figure 2 depicts an usage snapshot from log file (Shahabi, Zarkesh et al. 1997; Xiao, Zhang et al. 2001).

The element in the normalized session-page matrix is determined by the ratio of the visiting time on corresponding page to total visiting time, e.g. $a_{11} = 15/(15+43+52+31+44)*100 = 9.7$ ... and so on.

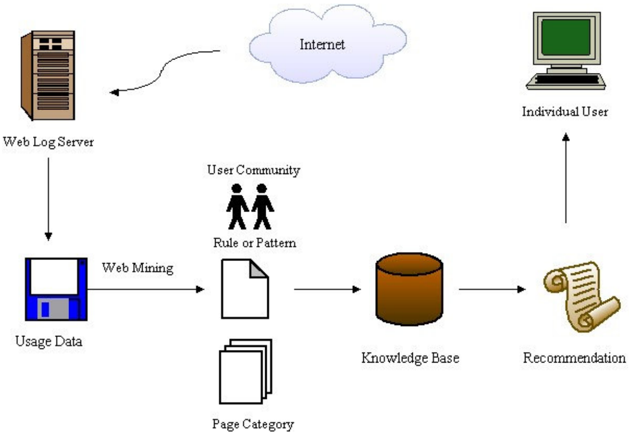


Fig. 1. The overview of Web Mining and Web Recommendation system

1) Main Movies: 20sec *Movies* News: 15sec *News*Box: 43sec *Box-Office* Evita: 52sec *News* Argentina:31 sec *Evita: 44sec*

2) Music Box: llsec *Box-Office* Crucible: 12sec *Crucible* Book: 13sec *Books: 19sec*

3) Main Movies: 33sec *Movies* Box: 21sec *Boxoffice* Evita: 44sec *News* Box: 53sec *Box-office* Evita: 61 sec *Evita : 31sec*

4) Main Movies: 19sec *Movies* News: 21sec *News b*ox: 38sec Box-*Office* Evita:61 sec *News* Evita:24sec *Evita* News: 31 sec *News* Argentina: 19sec *Evita: 39sec*

5) Movies Box: 32sec *Box-Office* News: 17sec *News* Jordan: 64sec *Box-Office* Evita: 19sec *Evita: 50sec*

6) Main Box: 17sec *Box-Office* Evita: 33sec *News* Box: 41 sec *Box-Office* Evita: 54sec *Evita* News: 56sec *News: 47sec*

$$SP_{ex} = \begin{bmatrix} 9.76 & 7.32 & 36.1 & 25.4 & 21.5 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 21.8 & 0.00 & 20.0 & 23.6 & 34.6 \\ 13.6 & 8.64 & 21.8 & 43.2 & 12.8 & 0.00 & 0.00 & 0.00 \\ 7.54 & 8.33 & 32.1 & 34.2 & 27.8 & 0.00 & 0.00 & 0.00 \\ 0.00 & 17.6 & 35.2 & 19.8 & 27.5 & 0.00 & 0.00 & 0.00 \\ 6.85 & 0.00 & 35.5 & 35.1 & 22.6 & 0.00 & 0.00 & 0.00 \end{bmatrix}$$

Fig. 2. A usage snapshot and its normalized session-page matrix expression

2.2 PLSA Model

The PLSA model is based on a statistic model called aspect model, which can be utilized to identify the hidden semantic relationships among general co-occurrence activities (Hofmann 1999). Similarly, we can conceptually view the user sessions over Web pages space as co-occurrence activities in the context of Web usage mining to discover the latent usage pattern. For the given aspect model, suppose that there is a latent factor space $Z = \{z_1, z_2, \cdots z_k\}$ and each co-occurrence observation data (s_i, p_j) is associated with the factor $z_k \in Z$ by varying degree to z_k. In this manner, each usage data (s_i, p_j) can convey the user navigational interest by mapping the observation data into the *k*-dimensional latent factor space. The degrees to which such relationships are "explained" by each factor are represented by the factor-conditional probabilities. Below is some background of PLSA. We use

following probability definitions to model usage data:

- $P(s_i)$ denotes the probability that a particular user session s_i will be observed in the occurrences data,
- $P(z_k | s_i)$ denotes a user session-specific probability distribution on the unobserved class factor z_k explained above,
- $P(p_j | z_k)$ denotes the class-conditional probability distribution of pages over the latent variable z_k.

Based on these definitions, we calculate probability of an observed pair (s_i, p_j) by adopting the latent factor variable z_k as:

$$P(s_i, p_j) = P(s_i) \bullet P(p_j | s_i) \qquad (1)$$

$$P(p_j | s_i) = \sum_{z \in Z} P(P_j | z) \bullet P(z | s_i) \qquad (2)$$

By applying Bayesian formula, we obtain the probability of an observation data associated with the latent factor as:

$$P(s_i, p_j) = \sum_{z \in Z} P(z) \bullet P(s_i | z) \bullet P(P_j | z) \qquad (3)$$

Following the likelihood principle, the total likelihood *Li* is determined as

$$L_i = \sum_{s_i \in S, p_j \in P} m(s_i, p_j) \bullet \log P(s_i, p_j) \qquad (4)$$

where $m(s_i, p_j)$ is the element of the session-page matrix corresponding to session s_i and page p_j.

From knowledge of statistics, Expectation Maximization (EM) algorithm is an effective way to perform maximum likelihood estimation in latent variable model (Dempster, Laird et al. 1977). Usually, two steps namely Expectation (E) and Maximization (M) step are iterating in this algorithm, i.e. E step leads to calculate the posterior probabilities for the latent factors based on the current estimates of conditional probability; whereas M step results in updating the estimated conditional probabilities and maximizing the likelihood based on the posterior probabilities computed in the previous E-step, i.e.

(1) In the E-step, we can simply apply Bayesian formula to generate following variable based on usage observation:

$$P(z_k | s_i, p_j) = \frac{P(z_k) \bullet P(s_i | z_k) \bullet P(p_j | z_k)}{\sum_{z_k \in Z} P(z_k) \bullet P(s_i | z_k) \bullet P(p_j | z_k)} \qquad (5)$$

(2) In M-step, we can compute:

$$P(p_j | z_k) = \frac{\sum_{s_i \in S} m(s_i, p_j) \bullet P(z_k | s_i, p_j)}{\sum_{s_i \in S, p_j \in P} m(s_i, p_j') \bullet P(z_k | s_i, p_j')} \qquad (6)$$

$$P(s_i \mid z_k) = \frac{\sum_{p_{j\in} \in P} m(s_i, p_j) \bullet P(z_k \mid s_i, p_j)}{\sum_{s_i^{'} \in S, p_j \in P} m(s_i^{'}, p_j) \bullet P(z_k \mid s_i^{'}, p_j)} \quad \textbf{(7)}$$

$$P(z_k) = \frac{1}{R} \sum_{s_i \in S, p_j \in P} m(s_i, p_j) \bullet P(z_k \mid s_i, p_j) \quad \textbf{(8)}$$

where

$$R = \sum_{s_i \in S, p_j \in P} m(s_{i,}, p_j) \quad \textbf{(9)}$$

Substituting equation (6)-(8) into (3) and (4) will result in the monotonically increasing of total likelihood Li of the observation data. The executing of E-step and M-step is repeating until Li is converging to a local optimal limit, which means the estimated results can represent the final probabilities of observation data.

It is easily found that the computational complexity of this algorithm is $O(mnk)$, where m is the number of user session, n is the number of page, and k is the number of factors.

3 Identifying Web Page Category and Task-Oriented Access Pattern

As we discussed in section 2, note that each latent factor z_k do really represent specific aspect associated with co-occurrence in nature. For each factor, the degree related to the co-occurrence is expressed by the factor-based probability estimate. From this viewing point, we, thus, can utilize the class-conditional probability estimates generated by the PLSA model and clustering algorithm to partition Web pages into various usage-based groups. Meanwhile, we can infer the latent factors by interpreting the meaning of "dominant" Web pages whose probabilities are exceeding the predefined threshold.

3.1 Discovering Web Page Category

Note that the set of $P(z_k \mid p_j)$ is conceptually representing the probability distribution over the latent factor space for a specific Web page p_j, we, thus, construct the page-factor matrix based on the calculated probability estimates, to reflect the relationship between Web pages and latent factors, which is expressed as follows:

$$vp_j = (c_{j,1}, c_{j,2}, \ldots, c_{j,k}) \quad \textbf{(10)}$$

Where $c_{j,s}$ is the occurrence probability of page p_j on factor z_s. In this way, the distance between two page vectors may reflect the functionality similarity exhibited by them. We, therefore, define their similarity by applying well-known cosine similarity as:

$$sim(p_i, p_j) = \left(vp_i, vp_j\right) \big/ \left(\|vp_i\|_2 \bullet \|vp_j\|_2\right) \quad \textbf{(11)}$$

where $\left(vp_i, vp_j\right) = \sum_{m=1}^{k} c_{i,m} c_{j,m}$, $\|vp_i\|_2 = \sqrt{\sum_{l=1}^{k} C_{i,l}^2}$

With the page similarity measurement (11), we propose a modified k-means clustering algorithm to partition Web pages into corresponding categories. The detail of the clustering algorithm is described as follows:

Algorithm 1 Clustering Web Page

Input: the set of $P(z_k \mid p_j)$, predefined threshold μ

Output: A set of Web page categories and centroids $PCL = \{PCL_1, \cdots, PCL_P\}$

1. Select the first page p_1 as the initial cluster PCL_1 and the centroid of this cluster: $PCL_1 = \{p_1\}$ and $Cid_1 = p_1$.
2. For each page p_j, measure the similarity between p_j and the centroid of each existing cluster $sim(p_j, Cid_i)$
3. If $sim\left(p_j, Cid_t\right) = \max_i(sim(p_j, Cid_i)) > \mu$, then insert p_j into the cluster PCL_t and update the centroid of PCL_t as

   $$Cid_t = 1/\left|PCL_t\right| \bullet \sum_{j \in SCL_t} vp_j \quad \textbf{(12)}$$

   where $\left|PCL_t\right|$ is the number of sessions in the cluster

   Otherwise, p_j will create a new cluster itself and is the centroid of the new cluster.
4. If there are still sessions to be classified into one of existing clusters or a session that itself is a cluster, go back to step 2 iteratively until it converges (i.e. all clusters' centroid are no longer changed)
5. Output $PCL = \{PCL_p\}$

In addition, note that $P(p_j \mid z_k)$ represents the conditional occurrence probability over the page space corresponding to a specific factor, whereas $P(z_k \mid p_j)$ represents the conditional probability distribution over the factor space corresponding to a specific page, which is expressed in the form of:

$$P(z_k \mid p_j) = \frac{P(p_j \mid z_k) \bullet P(z_k)}{\sum_{z_k \in Z} P(p_j \mid z_k) \bullet P(z_k)} \quad \textbf{(13)}$$

In such expression, we may consider that the pages whose conditional probabilities $P(p_j \mid z_k)$ and $P(z_k \mid p_j)$ are both greater than a predefined threshold μ can be viewed to contribute to one particular functionality related to the latent factor. Furthermore, we choose all pages satisfying aforementioned condition to form "dominant" page sets to characterize the latent factor.

3.2 Identifying Task-Oriented Access Pattern

Suppose that the conditional probability estimates are derived from the PLSA model as described above, we, in turn, utilize them to identify the user's underlying access task and to predict the potentially interested Web content to user in recommendation process.

Since the user session is represented as a sequence of visited pages, we can capture the task sequence derived from clicked pages within the session accordingly. This aim is accomplished by computing the posterior probability of each task based on Bayesian updating approach, given that pages are independent on tasks. These posterior probabilities associated with the various tasks indicate the likelihood of user's underlying intention. The usage pattern, therefore, is characterized as a sequence of tasks with corresponding probabilities. By presetting an appropriate threshold, we choose all tasks whose posterior probabilities are greater than the preset value as a dominant task collection to reflect the user's initial intention. In section 5, we present two examples of task-oriented access pattern to illustrate how such task sequences are represented in terms of probability weights.

4 Web Recommendation Based on Task-Oriented access pattern

The discovered task-oriented access pattern can actually reveal the user's intrinsic access intend associated with latent task factors. As a result, incorporating the identified sequence of dominant tasks with the task-based page categories derived from previous section will lead to discover the potential pages more likely to be visited or interested by the user in following period. The detailed algorithm of Web recommendation is described as follows:

Algorithm 2 Web Recommendation

Input: the active user session $s_i =< p_1^i, p_2^i, \cdots, p_t^i >, p_j^i \in P$, a set of estimated conditional probabilities $P(p_j \mid z_k)$ and threshold.

Output: the dominant task sequence corresponding to the user session $TL = \{z_1^i, \cdots, z_t^i\}$ and the top-N recommendation pages $RS = \{p_j^r\}$.

1. For each task $z_k \in Z$, which is independent on the pages, calculate the posterior probability of z_s given all pages in s_i by employing Bayesian updating method (Russell and Norvig 1995):

   $$P(z_k \mid s_i) = \alpha P(z_k) \prod_{p_j^i \in s_i} P(p_j^i \mid z_k)$$

   where α is a constant.

2. Choose all tasks whose conditional probabilities are greater than a preset threshold as the dominant task sequence corresponding to the user session.

   $$TL = \{z_k \mid z_k \in Z, P(z_k \mid s_i) > \mu\}$$

3. For each z_k in *TL*, incorporate it with the corresponding task-based page category, then compute the recommendation score for each page p_j as

   $$rs(p_j) = \sqrt{\sum_{z_k, p_j} P(z_k \mid s_i) \cdot P(p_j \mid z_k)}, p_j \in P, z_k \in TL$$

   Note that the recommendation score will be 0 if the page is already visited in the current session

4. Sort the computed recommendation scores from step 3 in a descending order, i.e. $rs = (rs(p_1^r), \cdots, rs(p_n^r))$, and choose the N pages with the highest scores to construct the top-N recommendation set.

   $$RS = \{p_j^r \mid rs(p_j^r) > rs(p_{j+1}^r), j = 1, 2, \cdots, N-1\}$$

5 Experiments and Evaluations

In order to evaluate the effectiveness of the proposed method based on PLSA model and explore the discovered latent semantic factor, we have conducted preliminary experiments on two real world data sets.

5.1 Data Sets

The first data set we used is downloaded from KDDCUP (www.ecn.purdue.edu/KDDCUP/). After data preparation, we have setup an evaluation data set including 9308 user sessions and 69 pages, where every session consists of 11.88 pages in average. We refer this data set to "KDDCUP data". In this data set, the numbers of Web page hits by the given user determines the elements in session-page matrix associated with the specific page in the given session.

The second data set is from a academic Website log files (Mobasher 2004). The data is based on a 2-week Web log file during April of 2002. After data pre-processing stage, the filtered data contains 13745 sessions and 683 pages. The entries in the table correspond to the amount of time (in seconds) spent on pages during a given session. For convenience, we refer this data as "CTI data".

Discovery of latent task factor with PLSA model has been investigated in our previous work (Xu, Zhang et al. 2004). In this part, we just present the experimental results in terms of Web page category, task-oriented access pattern as well as the evaluation of recommendation.

5.2 Examples of Web Page Categories

At this stage, we utilize aforementioned clustering algorithm to partition the Web pages into various clusters. By analysing the discovered clusters, we may conclude that many of groups do really reflect the single user access task; whereas others may cover two or more tasks, which may be relevant in nature. As indicated above, the former can be considered to correspond to the intuitive latent factors, and the latter may reveal the "overlapping" relationships in content among Web pages.

In Table 1, we list three Web page groups out of total generated groups from KDDCUP data set, which is expressed by top ranked page information such as page numbers and their relative URLs as well. It shows that each of these three page groups reflects sole usage task, which is consistent with the corresponding factor depicted in Table 1 of (Xu, Zhang et al. 2005). Table 2 illustrates two Web page groups from CTI data set correspondingly. In this table, the upper row lists the top ranked pages and their corresponding content from one of the generated page clusters, which reflect the task regarding searching postgraduate program information, and it is easily to conclude that these pages are all contributed to factor #13 displayed in Table 2 of (Xu, Zhang et al. 2005). On the other hand, the listed significative pages in lower row in the table involve in the "overlapping" of two dominant tasks, which are corresponding to factor #3 and #15 depicted in Table 2 of (Xu, Zhang et al. 2005).

Table 1. Examples of Web page groups from KDDCUP

Page	Content	Page	Content
10	main/vendor	38	articles/dpt_payment
28	articles/dpt_privacy	39	articles/dpt_shipping
37	articles/dpt_contact	40	articles/dpt_returns
27	main/login2	50	account/past_orders
32	main/registration	52	account/credit_info
42	account/your_account	60	checkout/thankyou
44	checkout/expresCheckout	64	account/create_credit
45	checkout/confirm_order	65	main/welcome
47	account/address	66	account/edit_credit
12	dpt_about	20	dpt_affiliate
13	dpt_about_mgmtteam	21	new_security
14	dpt_about_boarddirectors	22	new_shipping
15	dpt_about_healthwellness	23	new_returns
16	dpt_about_careers	24	dpt_terms
17	dpt_about_investor	57	dpt_about_the_press
18	dpt_about_pressrelease	58	dpt_about_advisoryboard
19	dpt_refer		

Table 2. Examples of Web page groups from CTI

Page	Content	Page	Content
386	/News	588	/Prog/2002/Gradect2002
575	/Programs	590	/Prog/2002/Gradis2002
586	/Prog/2002/Gradcs2002	591	/Prog/2002/Gradmis2002
587	/Prog/2002/Gradds2002	592	/Prog/2002/Gradse2002
65	/course/internship	406	/pdf/forms/assistantship
70	/course/studyabroad	666	/program/master
352	/cti/…/applicant_login	678	/resource/default
353	/cti/…/assistantship_form	679	/resource/tutoring
355	/cti/…/assistsubmit		

Note that with these generated Web page categories, we may make use of these intrinsic relationships among Web pages to reinforce the improvement of Web organization or functionality design. For example, the instrumental and suggestive task list based on the discovered page groups can be added into the original Web page as the means of *Adaptive Web Site Design*, to provide better service to users.

5.3 Examples of Task-Oriented Usage Patterns

As described in section 4, we exploit the posterior probability derived from PLSA model to identify the task-oriented usage pattern and predict the user's potentially visited Web pages by combining the task-oriented page categories into recommendation process. In the following table, we demonstrate two examples of derived task-based usage patterns through employing algorithm 2 on two real user sessions from KDDCUP and CTI dataset respectively. We list the active user sessions as well as tasks model derived from their sessions.

From the table, it is easily found that the two users have visited 10 and 11 pages respectively during their browsing period. The task-based usage patterns as well as their corresponding probabilities are depicted in the third and fourth column of the table. The upper part of the table shows that the user's activity actually involves in multiple purposes. However, the user's main intention is to perform online shopping as the probability of task #6 is significantly greater than the occurrence probabilities of other tasks. Therefore, we conclude that the dominant theme of first user's behaviour is actually locating on task #4.

Table 3. Examples of Task-Oriented Usage Pattern

#	Real user session	Task # & title	Prob.
1	1. boutique 2. search-result 3. ProductDetailLegcare 4. shopping_cart 5. login2 6. Welcome 7. expressCheckout 8. your_account 9. confirm_order 10. vendor	Online shopping (#6) Product Legcare (#2) Boutique (#9) Department search (#1) Vendor info (3)	0.94 0.02 0.02 0.01 0.01
2	1. admissions/ 2. admissions/requirements 3. admissions/mailrequest 4. admissions/orientation 5. gradapp/appmain_right 6. /news/default 7. /programs/ 8. programs/gradcs2002 9. programs/gradect2002 10. /programs/gradhci2002 11. /programs/core_guide	Admission (#4) Postgrad Program (#13)	0.63 0.37

For another user, we can find that the user was mainly conducting two tasks, i.e. task #4 and task #13. Incorporating the derived task model in table 1, we can further identify that task #4 represents prospective students searching for admission information, such as requirement, orientation etc., whereas task #13 reflects the activity of those students who are particularly interested the postgraduate programs in IT discipline. Unlike the first user, the second user clearly exhibits the cross-interest as the difference of the two corresponding probabilities is not quite significant.

Once the task model of user is identified, it is further utilized to recommend user preferred content accurately

5.4 Evaluation Metric for Web Recommendation

From the view of the user, the effectiveness of the proposed approach is evaluated by the precision of recommendation. Here, we exploit a metric called *hit precision* (Mobasher, Dai et al. 2002) to measure the effectiveness in the context of top-*N* recommendation. Given a user session in the test set, we extract the first *j* pages as an active session to generate a top-*N* recommendation set via the procedure described in section 4. Since the recommendation set is in descending order, we then obtain the rank of $j+1$ page in the sorted recommendation list. Furthermore, for each rank $r > 0$, we sum the number of test data that exactly rank the rth as $Nb(r)$. Let $S(r)=\sum_{i=1}^{r} Nb(i)$, and $hitp = S(N)/|T|$, where $|T|$ represents the number of testing data in the whole test set. Thus, $hitp$ stands for the hit precision of Web recommendation process.

Table 4 gives the effectiveness of recommendation in terms of hit precision. From the table, it is shown that bigger the N number is, higher the $hitp$ value is. In most case, the hit precision parameters are larger than 30%.

Table 4. The Results of Recommendation Hit Precision

N	5	6	7	8	9	10	11	12
hitp	0.17	0.19	0.20	0.22	0.27	0.30	0.32	0.33
N	13	14	15	16	17	18	19	20
hitp	0.34	0.35	0.36	0.36	0.37	0.37	0.37	0.38

6 Conclusion and Future Work

Web transaction data between Web visitors and Web functionalities usually convey user task-oriented behavior pattern. As a result, there is an increasing demand to develop techniques that can not only discover user task-oriented usage patterns, but also provide more benefits for recommend user more interested or preferred content In this paper, we have developed a Web recommendation technique by exploiting the knowledge of usage pattern from Web usage mining process based on PLSA model. With the proposed probabilistic method, we can measure the co-occurrence activities (i.e. user session) in terms of probability estimations to capture the underlying relationships among users and pages. Analysis of the estimated probabilities leads to build up task-oriented usage patterns and Web page categories, identify the hidden factors conceptually representing user interests or tasks. The discovered usage patterns can result in improvement of Web recommendation. We demonstrate the usability and effectiveness of our technique through experiments performed on the real world datasets.

Our future work will focus on the following issues: we intend to conduct experimental work on more datasets to validate the scalability of our approach. Meanwhile we plan to develop other machine learning algorithms to improve the accuracy of Web recommendation.

Acknowledgement

This research has been partly supported through ARC Discovery Project Grant DP0345710 and National Natural Science Foundation of China (No 60403002).

7 Reference

Agarwal, R., C. Aggarwal, et al. (1999): A Tree Projection Algorithm for Generation of Frequent Itemsets. *Journal of Parallel and Distributed Computing* **61**(3): 350-371.

Agrawal, R. and R. Srikant (1994): Jorge B. Bocca and Matthias Jarke and Carlo Zaniolo. *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, Santiago, Chile, 487-499, Morgan Kaufmann.

Agrawal, R. and R. Srikant (1995): Mining Sequential Patterns. *Proceedings of the International Conference on Data Engineering (ICDE)*, Taipei, Taiwan, 3-14, IEEE Computer Society Press.

Cohn, D. and H. Chang (2000): Learning to probabilistically identify authoritative documents. *Proc. of the 17th International Conference on Machine Learning*, San Francisco, CA, 167-174, Morgan Kaufmann.

Cohn, D. and T. Hofmann (2001): The missing link: A probabilistic model of document content and hypertext connectivity: an in *Advances in Neural Information Processing Systems*. T. G. D. Todd K. Leen, and Tresp, V.(eds). MIT Press.

Dempster, A. P., N. M. Laird, et al. (1977): Maximum likelihood from incomplete data via the EM algorithm. *Journal Royal Statist. Soc. B* **39**(2): 1-38.

Dunja, M. (1996). Personal Web Watcher: design and implementation, Department of Intelligent Systems, J. Stefan Institute, Slovenia.

Han, E., G. Karypis, et al. (1998): Hypergraph Based Clustering in High-Dimensional Data Sets: A Summary of Results. *IEEE Data Engineering Bulletin* **21**(1): 15-22.

Herlocker, J., J. KONSTAN, et al. (1999): An Algorithmic Framework for Performing Collaborative Filtering. *Proceedings of the 22nd ACM Conference on Researchand Development in Information Retrieval (SIGIR'99)*, Berkeley, CA.

Herlocker, J. L., J. A. Konstan, et al. (2004): Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* **22**(1): 5 - 53.

Hofmann, T. (1999): Probabilistic Latent Semantic Analysis. *Proc. of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*, Berkeley, California, 50-57, ACM Press.

Hofmann, T. (2001): Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Machine Learning Journal* **42**(1): 177-196.

Hofmann, T. (2004): Latent Semantic Models for Collaborative Filtering. *ACM Transactions on Information Systems* **22**(1): 89-115.

Jin, X., Y. Zhou, et al. (2004): A Unified Approach to Personalization Based on Probabilistic Latent Semantic Models of Web Usage and Content. *Proceedings of the AAAI 2004 Workshop on Semantic Web Personalization (SWP'04)*, San Jose.

Joachims, T., D. Freitag, et al. (1997): WebWatcher: A Tour Guide for the World Wide Web. *Proceedings of the International Joint Conference in AI (IJCAI97)*, Los Angeles.

Konstan, J., B. Miller, et al. (1997): Grouplens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM* **40**: 77-87.

Lieberman, H. (1995): Letizia: An agent that assists web browsing. *Proc. of the 1995 International Joint Conference on Artificial Intelligence*, Montreal, Canada, 924-929, Morgan Kaufmann.

Mobasher, B. (2004): Web Usage Mining and Personalization: an in *Practical Handbook of Internet Computing*. M. P. Singh(eds). CRC Press.

Mobasher, B., H. Dai, et al. (2002): Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization. *Data Mining and Knowledge Discovery* **6**(1): 61-82.

Perkowitz, M. and O. Etzioni (1998): Adaptive Web Sites: Automatically Synthesizing Web Pages. *Proceedings of the 15th National Conference on Artificial Intelligence*, Madison, WI, 727-732, AAAI.

Russell, S. J. and P. Norvig (1995): *Artificial Intelligence, A Modern Approach*, Prentice Hall.

Shahabi, C., A. Zarkesh, et al. (1997): Knowledge discovery from user web-page navigational. *Proceedings of the 7th International Workshop on Research Issues in Data Engineering (RIDE '97)*, 20-29, IEEE Computer Society.

Shardanand, U. and P. Maes (1995): Social Information Filtering: Algorithms for Automating 'Word of Mouth'. *Proceedings of the Computer-Human Interaction Conference (CHI95)*, Denver, CO.

Xiao, J., Y. Zhang, et al. (2001): Measuring similarity of interests for clustering web-users. *Proceedings of the 12th Australasian Database conference (ADC2001)*, Queensland, Australia, **35:** 107-114, ACS Inc.

Xu, G., Y. Zhang, et al. (2004): Discovering User Access Pattern Based on Probabilistic Latent Factor Model. *Proceeding of 16th Australasian Database Conference*, Newcastle, Australia, **39:** ACS Inc.

Xu, G., Y. Zhang, et al. (2005): Using Probabilistic Semantic Latent Analysis for Web Page Grouping. *15th International Workshop on Research Issues on Data Engineering: Stream Data Mining and Applications (RIDE-SDMA'2005)*, Tokyo, Japan.

Handling of Current Time in Native XML Databases

Bela Stantic[1] **Guido Governatori**[2] **Abdul Sattar**[1]

[1] Institute for Integrated and Intelligent Systems,
Griffith University, Brisbane Australia
[2] School of Information Technology and Electrical Engineering,
The University of Queensland, Brisbane Australia,
Email: B.Stantic@griffith.edu.au, guido@itee.uq.edu.au, A.Sattar@griffith.edu.au

Abstract

The introduction of Native XML databases opens many research questions related to the data models used to represent and manipulate data, including temporal data, in XML. Increasing use of XML for Valid Web pages warrants an adequate treatment of *now* in Native XML databases. In this study, we examined how to represent and manipulate *now-relative* temporal data. We identify different approaches being used to represent current time in XML temporal databases, and introduce the notion of storing variables such as 'now' or 'UC' as strings in XML native databases. All approaches are empirically evaluated on a query that time-slices the timeline at the current time. The experimental results indicate that the proposed extension offers several advantages over other approaches: better semantics, less storage space and better response time.

Keywords: Native XML Databases, Temporal Databases, current time

1 Introduction

There has been a lot of research into adding time to different data models, for example to Semantic data model, Knowledge-based data model and Entity Relationship model. But most of the literature in temporal databases is related to Relational and Object-Oriented data model (Jensen 2000). A large number of temporal data models were studied and the design space for the relational data model has been exhaustively explored (Date, Darwen & Lorentzos 2002). Clifford et al. (Clifford, Croker & Tuzhilin 1994) classified them as two main categories: temporally ungrouped and temporally grouped. Although temporally grouped models have long been known to be more expressive and appealing to intuition (Clifford, Croker, Grandi & Tuzhilin 1995), they cannot be supported easily in the framework of flat relations and SQL, and therefore they have not been actually implemented in temporal database projects and prototypes (Ozsoyoglu & Snodgrass 1995).

Recently research in temporal representtion and reasonng has been extended to XML. Research on adding temporal features to XML has taken into account change, versioning, evolution and also explicitly temporal aspects. Some extensions of the XML, such as τXQuery language, have been proposed to extend XQuery for temporal support (Chawathe, Abiteboul & Widom 1999), (Gao & Snodgrass 2003). Recently, database researchers, vendors and SQL standardization groups started work toward extensions of SQL with XML capabilities (*http://www.sqlx.org* 2004) and to support languages such as XQuery (*XQuery 1.0: An XML query language* 2004) on XML data (Carey, Kiernan, Shanmugasundaram & et al 2000) (Funderburk, Kiernan, Shanmugasundaram, Shekita & Wei 2002). XML and XQuery can be viewed as a new powerful data model and query language providing a better basis for representing and querying temporal data. In contrast to relational databases temporally grouped data model is supported well by XML and its query languages.

Most modern database applications involve a significant amount of time dependent data and a substantial proportion of this data is *now-relative*, i.e. the end time of their validity follows the current time. *Now-relative* data are natural and meaningful part of every temporal database as well as being the focus of most queries. It has been shown that different semantics for *now* in temporal relational environment significantly influence performance (Torp, Jensen & Bohlen 1999), (Stantic, Khanna & Thornton 2004). As XML is used for Valid Web, which has temporal features and is associated with current validity of Web pages, handling *now* in XML is even more important than in relational databases. While significant research has been oriented toward adding different temporal dimensions to XML and querying such data with XQuery and on different extensions to XQuery, handling current time or *now* has received only a little attention. The majority of the proposals for simplicity consider only closed intervals. For closed intervals exact starting and ending point must be known up front. This is obviously unrealistic in real application domains. These proposals do not support data where ending time of validity follows the current time, *now-relative* data. Proposals that mentioned 'now' usually do so only briefly, in line with temporal relational research, recommending the *MAX* approach to represent current time or suggesting the usage of user defined functions. These recommendations are usually made without any further explanation or empirical results that show the efficiency and support the recommendation (Wang & Zaniolo 2003).

For the above reasons we decided to investigate how different semantic for *now* influence not only the performance but also the accuracy of the queries. Also, we decided to investigate whether temporally groping of data offers any advantage and better performance than the direct conversion of relations to XML. In order to do this testing we decided to use data set generated in relational environment.

At first, we identify available options for representing *now* in XML-Temporal. Interestingly, the options ruled out as not suitable for relational databases can be considered as viable options for XML and Native XML databases. We introduce the notion of storing variables such as 'now' or 'UC' as strings in XML native databases. All approaches are empirically eval-

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

uated on a query that time-slices the timeline at the current time. The experimental results indicate that the proposed extension offers several advantages over other approaches: better semantics, less storage space and better response time.

The remainder of this paper is organised as follows, in the next section we look more closely at temporal dimensions of interest in XML. In Section 3 we demonstrate that any temporal data can be viewed in two different ways: the *DIRECT* representation and in temporally grouped data model; both can be represented in XML and efficiently queried by XQuery. Furthermore in this section, we discuss available methods to represent current time and highlight their limitations and disadvantages. Also, in Section 3, we illustrate the experiment taken to evaluate the identified approaches to represent current time and present the experimental results and analysis. Finally, in Section 4, we present our conclusions and suggest future work.

2 Temporal data and XML

A significant percentage of data for web pages is dynamic and generated as a result of queries most often in form of XML, so it is more natural and more convenient to store data directly in XML format. There are two basic methods to store XML documents in a database. The first is to map the document's schema to a database schema and transfer data according to that mapping. The second is to use a fixed set of structures that can store any XML document. Databases that support the first method are called XML-enabled databases while databases that support the second method are called native XML databases (xml.com 2005).

XML-enabled databases are useful when publishing existing data as XML or importing data from an XML document into an existing database. For instance, DB2s XML Extender (www3.ibm.com 2005) takes advantage of user-defined functions and stored procedures to map between XML data and relational data. Another approach of XML-enabled databases is a middleware based approach such as used in SilkRoute (Fernandez, Tan & Suciu 2000) and XPERANTO (Carey et al. 2000). However, XML-enabled databases are not a good way to store complete XML documents. The reason is because they store data and hierarchy but discard everything else: document identity, sibling order, comments, processing instructions, and so on. This approach obviously has some limitations as XML itself is more powerful and conversion from relational model to XML is considered as straightforward, but the opposite conversion is not always possible without some ingenuity.

Native XML databases, on the other hand, store complete documents and can store any document, regardless of schema. Native XML databases are used in a wide number of fields, such as health care, genetics, insurance, data integration, messaging, Web sites, etc. The most popular of these are storing and querying document-centric XML, integrating data, and storing and querying semi-structured data. Native XML databases are used in these cases because the data involved does not easily fit the relational data model, while it does fit the XML data model.

In order to access data in XML that are valid at particular (and most often present) time, it is necessary to represent time dimensions in XML. This issue opens a big question not only for databases but also in computer science in general, i.e. how to handle and store current time or *now*. The assumption mostly taken in the literature is that only closed intervals of validity exist (Wang & Zaniolo 2003). This is obviously unrealistic; often ending points will be unknown and will follow the advancing current time indicating that, for example, fact is valid *now* and that its ending time of validity is unknown.

2.1 Time dimensionality of interest

Research on adding temporal features to XML has identified different time dimensions of interest. The focus of some approaches was on the representation and management of changes, where different versions of data are produced by updates. In this approach, temporal attributes are often used to timestamp stored versions (Amagasa, Yoshikawa, & Uemura 2000), (Chawathe et al. 1999) and they represent the time the updates were applied, which basically has the same semantics as transaction time. Transaction time is the time that shows the status of the data in a database, from when it is inserted in the database to when it is logically deleted, if ever. With respect to the Web, it represents the on-line availability and versioning of resources in a Web site, even if they are basically not created by *transactions.* This notion of time requires storing the current time or *now* to represent that the element is current and belong to the current database state that is not logically deleted.

Another approach, basically represents the classical notion of valid time. It is a XML/XSL infrastructure, named 'The Valid Web', designed to represent and manage temporal Web documents containing historical information (Grandi & Mandreoli 2000). In this approach timestamps are explicitly encoded by the document authors to assign validity to information content. Temporal documents can then be selectively browsed in accordance with a user-supplied temporal period of interest. This approach is further extended in (Wang & Zaniolo 2002). The valid time is the time when some fact is true in the real world. In Web applications, it concerns the temporal validity of the information carried by the contents of a Web resource. It is obvious that this notion of time also requires storing the current time or *now.* This is because it is expected to have facts that started to be valid at certain past time, they are valid *now*, and the end of their validity is unknown. The end time of validity of facts follows the current time.

There are several other temporal dimensions that have been also mentioned in the literature in relation to XML: *navigation time*, which concerns the interaction of users during their browsing of Web sites. Furthermore, a *publication time*, in the context of legal documents, and *efficiency time* (Grandi, Mandreoli, Tiberio & Bergonzini 2003). Navigation time and publication time do not require to store 'now' and are relevant for this study.

2.2 Representation of temporal dimensions

There are basically two different approaches to represent temporal dimensions in XML.

- to represent timestamps by XML elements,
- to represent timestamps by the temporal attributes of the XML elements.

For simplicity, in our running samples we will use data from temporal relational databases but our discussion applies to any more complex nested XML structure. We will consider valid time data but it is applicable to any other temporal dimension that requires to handle 'now'. The running sample used in this paper, which captures the history of Scott's positions, is shown in Table 1.

Name	Position	Vstart	Vend
Scott	A	2000-05-19	2001-03-12
Scott	B	2001-03-12	2004-03-10
Scott	C	2004-03-10	now

Table 1: Employment history

The first method represents a simple flat translation of relational attributes to XML elements, as introduced in (Fernandez et al. 2000). In this model, which we dubbed as the *DIRECT* model, each attribute is represented by an element in XML. According to this approach, the XML structure representing the data shown in Table 1 would look like:

```
<Employment>
<row>
    <Name>Scott </Name>
    <Position>A</Position>
    <Vstart>2000-05-19</Vstart>
    <Vend>2001-03-12</Vend>
</row>
<row>
    <Name>Scott </Name>
    <Position>B</Position>
    <Vstart>2001-03-12</Vstart>
    <Vend>2004-03-10</Vend>
</row>
    <row>
    <Name>Scott </Name>
    <Position>B</Position>
    <Vstart>2004-03-10</Vstart>
    <Vend>now</Vend>
</row>
</Employment>
```

The second approach relies on XML ability to have attributes within the elements. By adding attributes to the element it becomes a complex type. But the attributes themselves are always declared as a simple type. This means that an element with attributes always has a complex type definition. For example:

```
<Position Vstart="2000-05-19"
       Vend="2001-03-12">A</Position>
```

This approach is suitable for storing XML temporal data related to the representation and management of changes. Also, it is suitable to manage temporal Web documents containing historical information. Furthermore, this approach enables usage of temporally grouped data model. Clifford et al. have shown that the temporally grouped data model has more expressive power and is more natural since it is history oriented (Clifford et al. 1994).

It is possible to restrict data supplied for attributes and also to ensure that they are supplied. When an XML element or attribute has a type defined, it puts a restriction on the element's or attribute's content. For example, if an XML element is of type "xs:date" and contains a string the element is not valid.

```
<xs:attribute name="Vstart" type="xs:date"
                          use="required"/>
```

Considering the running sample from Table 1 temporally grouping data by Employee *Name*, will result in data as represented in Table 2.

Temporally grouped data, presented in Table 2, can be easily represented in XML using the attribute approach:

Name	Position
2000-05-19	*2000-05-19*
	A
	2001-03-12
	2001-03-12
Scott	**B**
	2003-02-15
	2003-02-15
	C
now	*now*

Table 2: Temporally Grouped Valid Time History of Employees

```
<Employment Vstart="2000-05-19" Vend="now">
    <Name Vstart="2000-05-19"
          Vend="now">Scott </Name>
    <Position Vstart="2000-05-19"
              Vend="2001-03-12">A</Position>
    <Position Vstart="2001-03-12"
              Vend="2003-02-15">B</Position>
    <Position Vstart="2003-02-15"
              Vend="now">C</Position>
</Employment>
```

It is considered easy to perform different queries using XQuery, for example to retrieve the snapshot at the certain date on closed intervals in either of approaches to add temporal dimensions to XML. But it is an open question how to represent current time or *now* in XML-temporal in order to efficiently and accurately access open ended intervals, *now-relative* XML-temporal data.

2.3 Modelling *now-relative* temporal data in XML

In line with research in temporal databases applied to relational technology, with respect to different representations for *now*, we decided to evaluate same approaches in XML.

MAX

The often mentioned approach to represent current time is to represent the current time as unrealistic large date most often used '31-DEC-9999', which in XML to ensure the order has to be in format 'YYYY-MM-DD' or '9999-12-31'. In the reminder of this paper we will refer to this approach as *MAX* approach.

Considering the nature of XML and Native XML databases, and the fact that data are stored as text, there are several other different representations for current time that could be of interest.

Null Data

Despite being ruled out as not suitable in relational databases, as columns with *NULL* can not be efficiently indexed, we decided to evaluate the *NULL* approach due to the specific nature of *NULL* in XML. In the database world, null data means that data simply is not there. Considering the XML, *NULL* can mean that value is inapplicable or value is missing. XML supports the concept of *NULL* data through optional element types and attributes. If the value of an optional element type or attribute is null, it simply is not included in the document. As with databases, attributes containing zero length strings and empty elements are not null: their value is a zero-length string.

Some Native XML databases offer the choice of defining what constitutes *NULL* in an XML document, including support for `xsi:null` attribute from XML Schema.

Variables

Due to the nature of XML and Native XML databases to store all data as text, it is possible to consider variables such as 'now' or 'UC' (until changed) to be stored as words 'now' or 'UC' to represent current time. These variables can be simple stored as text in native XML databases. Variables to represent current time are widely recommended in the literature but have not been appropriate for storing it in relational temporal database environment, and this approach has been ruled out. This is because *date type* cannot accommodate such variables. In contrast XML offers the possibility to create new complex data-type that inherit properties of the data-types used to create them. Thus it is possible to create a new temporal data-time as the union of normal time data-type and the string 'now'. In any case, even if XML has data-types these are just defined as combinations of Unicode characters, thus they are simply strings of text.

A further advantage of the 'now' approach is that opens the possibility to separate the time value representation from the representation of *now*, for example by introducing an empty and optional sub-element `<now/>` of `<vend>`. Thus we can have:

```
<vend><now/></vend>
```

for the ending time of *now*. Accordingly queries that require 'now' can traverse a path expression that descents to *now*, e.g.,

```
//vend/now
```

while, for queries where this is not needed, the expression

```
//vend < current-date()
```

is well-defined.

In addition, this strategy can take advantage of DBMS offering element indexes beside word indexes. In this case all `<now/>` will be included in an element index and the index can be used to faster search. This is in contrast with word indexes where 'now' can occur inside a sentence instead of as a special timestamp; consequently we have to check that the elements where 'now' occurs are of the right type.

3 Empirical study

In order to find the best choice for representing *now* in XML we decided to test performance and accuracy of all identified approaches. Testing was performed both on XML structure where timestamps are represented as XML elements (Direct model) and where timestamps are represented as attributes of XML elements (on temporally grouped data). Also, we intended to compare query response time and space usage for *Direct* and grouped model.

During the experiment we identified that some of the approaches to represent *now* do not yield the correct answer. This is the case if data contains closed intervals where the ending point is bigger than the current time. For that reason, we decided to include in our tests checking whether the query yields the correct number of elements that satisfy the given condition.

3.1 Environment

The experimental results presented in this section are computed on four 450MHZ CPU - SUN UltraSparc II processor machine running the open source Native XML eXist database (*eXist:http://exist.sourceforge.net/* 2004). During the testing server did not have any other significant load.

We decided to test the performance of a point query that timeslices time line at the current time. Point queries, as a special type of range queries, are considered to be the most important query type for temporal data. This is because it is expected that the current state of reality will be queried most often.

Searching a native XML database is handled in different ways, depending on the vendor of the database. Some native XML databases require the user to select the elements or attributes to be indexed. This information is then used to build an index that the searching mechanism can use to faster locate matching documents. Other native XML databases simply index all elements in a document, which obviously causes the need for more storage space. Indexing all elements in native XML databases has more sense compared with indexing of all columns in a relational database. While most of the XML native databases use well proven B^+-tree structures for indexing, specific demand of XML databases has forced introduction of different approaches such as: Reverse-Lookup indexing and Forward Dictionary Segment Build-Up indexing invented by QuiLogic, as well as traditional indexing technologies like hashing. The native XML eXist database, used for this experiment, uses B^+-tree structures for indexing. Users have option to define the elements and depth that should be indexed.

In our experiments index depth was set to three and all elements were included in the index.

3.2 Data sets

In order to investigate the effect of different percentage of *now-relative* data we used different data distributions. The start position of the intervals was always uniformly distributed on the interval domain, while the duration and percentage of *now-relative* data was varied. The following data distributions have been considered:

- Uniformly distributed interval start and uniform distributed duration within the range [1,10000] with 10% of uniformly distributed *now-relative* data.
- Uniformly distributed interval start and uniform distributed length within the range [1, 10000] with 20% of uniformly distributed *now-relative* data.

In relational environment 100.000 rows of sample data was generated and then converted to XML format. Same sample data set represented in *Direct* model required 400.000 elements while temporally grouped model for same data, due to the grouping, required only 154.256 XML elements. Part of data for temporary grouped model is shown in Table 2. For each approach to represent *now* we created two XML files, one for temporally grouped model and one for *Direct* model. XML files differ only on the sematic to represent *now*. The resulting files were imported into *eXist* XML Native database.

3.3 Query Sets

We focus on intersection queries and particularly on Point queries as specific cases. The results for intersection queries also hold for the containment and enclosure queries, as those are a subset of the intersection query. We use two different query sets:

- Window: a set of queries sorted according to the start point and with a fixed length. This query set is covering the whole data space.
- Random: A set of random query intervals with different answer size.

3.4 Results

All identified approaches to represent *now* have been tested. For *MAX* approach, where current time is stored as some unrealistically big date, we used 9999-12-31. XML does not support data types in any meaningful sense of the word, all data in an XML document are stored as text. This is even if data represents another data type, such as date or integer. For that reason, it was necessary to represent date in format YYYY-MM-DD to ensure that dates can be ordered into ordered list.

The XQuery code used to test the performance of the *MAX* approach without referencing the current time approach is:

```
for $b in doc("/db/now/dirmax.xml")/table/row
    where $b/vend='9999-12-31' and
        $b/vstart<xs:string(current-date())
    return
        <result> { $b } </result>
```

We found that without referencing to the current time the query yields the wrong answer if data contains closed intervals with ending point bigger than the current date. These intervals are meaningful for all temporal data. In order to get the correct answer, considering the number of elements returned, there is a need to compare ending point of the interval validity with the current time. This can be achieved using the XQuery function `current-date()` that references to the current time:

```
for $b in doc("/db/now/dirmax.xml")/
                table/row
    where $b/vend>=xs:string(current-date())
     and $b/vstart<xs:string(current-date())
    return
        <result> { $b } </result>
```

Usage of user-defined functions ensures that the query yields the correct answer when current time is represented with *MAX* approach. A sample user-defined function `check_now` (expressed in XQuery) returns `vend`, if the value is different from '9999-12-31' and `current-date` otherwise.

```
declare function local:check_now
                ($n as xs:string) as
     xs:string{ if ($n="9999-12-31") then
       xs:string(current-date()) else $n };
 for $b in doc("/db/now/dirmax.xml")/
                table/row
    where local:check_now($b/vend)>=
                xs:string(current-date())
        and  $b/vstart<
            xs:string(current-date())
    return
        <result> { $b } </result>
```

Usage of this user-defined function guarantees correct answers but suffers from extremely poor performance as it cannot use indexes, so a sequential search is required.

The nature of the XML native databases, all data are stored as text, opens the possibility to store *now* as the word "now". This approach could not be considered for relational databases, since it is not possible to store characters into date data type. Introduction of variables in temporal relational databases leads to the under researched area of *Variable databases.* In XML it is possible to create a new complex data-type as the union of normal time data-type and the string 'now'. Because storing variables as text in XML is straightforward we decided to test performance and accuracy of representing current time with the variable such as 'now'. To indicate that the fact is currently valid the variable 'now' is assigned for the ending point of their validity. The following XQuery was used for performance and accuracy testing of variable approach for representing current time.

```
for $b in doc("/db/now/dirnow.xml")/
                table/row
       where $b/vend>='now' and
        $b/vstart<xs:string( current-date())
       return
            <result> { $b } </result>
```

Without referencing to the current time the variable approach also does not yield correct answers if data contains closed intervals with ending point bigger than the current date. Referencing to the current time yields correct answer.

Because XML native databases have a different view to *NULL* data than the relational databases we decided to reconsider and test the performance and accuracy of the *NULL* approach to represent current time. Advantages of *NULL* are obviously from used space point of view. Response time for *NULL* approach is very good in case of not referencing to the current time but due to the wrong answer problem it can not be considered. To get the correct answer there is a need to reference to the current time with user defined functions. Despite being previously suggested as favorite in the literature, usage of user defined functions performed very poorly.

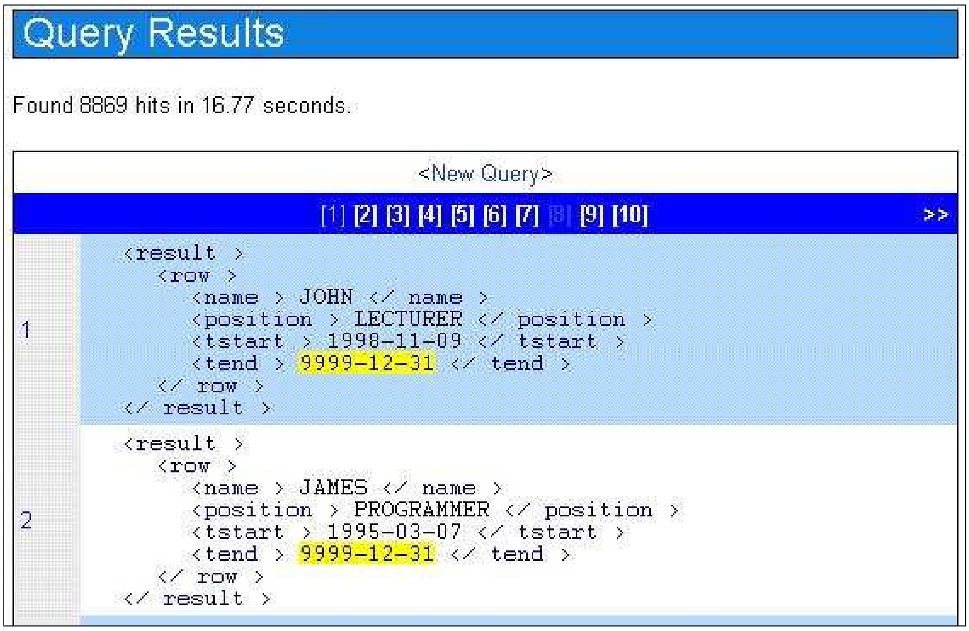
For temporally grouped model we performed queries on identified approaches to representing *now* with and without referencing to the current time. Same as for *DIRECT* model, any approach that does not reference to the current time yields wrong answer if data contains closed intervals of validity where the ending point of validity is beyond the current time. All such intervals are omitted and not included in the answer. The sample query for *MAX* approach that references to the current date function is as follows:

```
for $s in doc("/db/now/grmax.xml")/
                table/row/
position[@vstart<xs:string(current-date())
 and
@vend>=xs:string(current-date())]
return
<result> { $s/../name }
         {$s }
</result>
```

Also, we performed a query that finds all employees that have current position and have started their employment before the certain date. We performed this experiment to investigate how answer size effects the response time.

For *NULL* approach to represent current time it is not possible to compare with the current time directly because *NULL* is represented by the empty string and it is not clear whether the empty string is bigger or smaller than any string. Only possibility is to use user defined functions.

Testing *NULL* with temporally grouped model yielded totally wrong number of elements and had a very long query response time so we did not include the results in the above table.



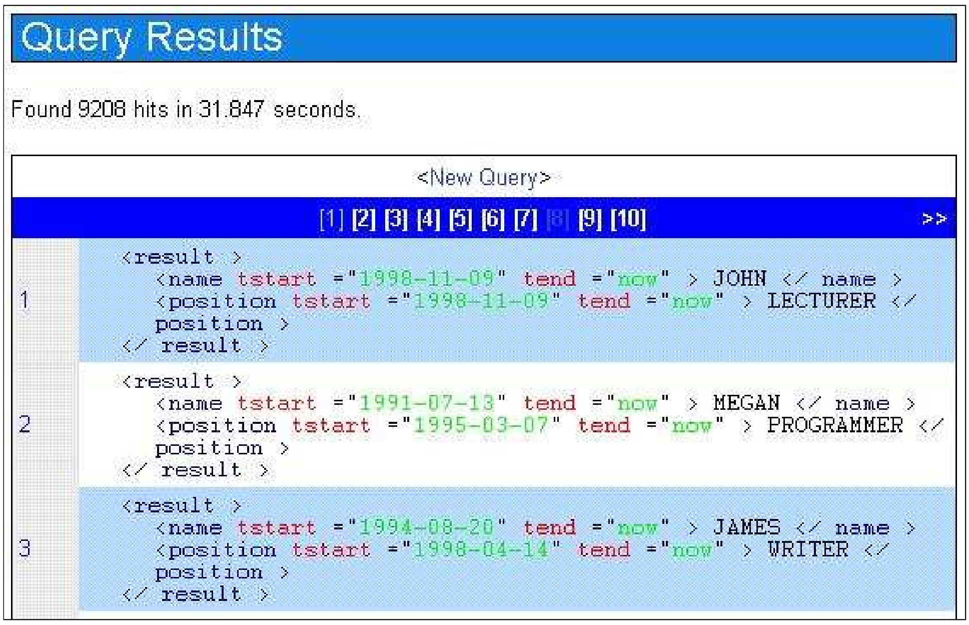
Query Results

Found 8869 hits in 16.77 seconds.

```
1   <result >
      <row >
        <name > JOHN </ name >
        <position > LECTURER </ position >
        <tstart > 1998-11-09 </ tstart >
        <tend > 9999-12-31 </ tend >
      </ row >
    </ result >

2   <result >
      <row >
        <name > JAMES </ name >
        <position > PROGRAMMER </ position >
        <tstart > 1995-03-07 </ tstart >
        <tend > 9999-12-31 </ tend >
      </ row >
    </ result >
```

Figure 1: MAX approach without reference to the current date



Query Results

Found 9208 hits in 31.847 seconds.

```
1   <result >
      <name tstart ="1998-11-09" tend ="now" > JOHN </ name >
      <position tstart ="1998-11-09" tend ="now" > LECTURER </
      position >
    </ result >

2   <result >
      <name tstart ="1991-07-13" tend ="now" > MEGAN </ name >
      <position tstart ="1995-03-07" tend ="now" > PROGRAMMER </
      position >
    </ result >

3   <result >
      <name tstart ="1994-08-20" tend ="now" > JAMES </ name >
      <position tstart ="1998-04-14" tend ="now" > WRITER </
      position >
    </ result >
```

Figure 2: Temporally Grouped model - variable approach with reference to the current time

Approach	Reference current time	Query yields correct answer	Timeslice query (sec)	Contained query (sec)
MAX	No	No	16.77	7.78
MAX	Yes	Yes	27.47	14.08
MAX with user defined function	Yes	Yes	100.48	98.98
’now’	No	No	14.96	7.14
’now’	Yes	Yes	23.88	12.25
NULL	No	No	22.08	9.13
NULL with user defined function	Yes	Yes	97.64	94.26

Table 3: Direct model - response time for different representations of *now*, 20 % of now-relative data

3.5 Analysis

It is significant to note that, if there is no reference and comparison to the current time the query yields the wrong answer, considering the number of elements returned that satisfy the query criteria. This is because all elements with closed interval whose ending time is bigger than the current time are not included in the answer. Usage of user defined functions, which was previously suggested as favorite in the literature, has poor response time as index can not be used and sequential search is required. Due to the nature of Native XML databases, where all data is stored as text, the current time can be stored as variable such as ‘now’ or ‘UC’. Because the ASCI code of the mentioned variables is bigger than `xs:string(current-date())`, the usage of variables yields correct answer and at the same time uses less space. Surprisingly,

Approach	Reference current time	Query yields correct answer	Timeslice query (sec)	Contained query (sec)
MAX	No	No	32.36	20.78
MAX	Yes	Yes	36.10	20.23
MAX with user defined function	Yes	Yes	104.43	94.18
’now’	No	No	27.03	16.95
’now’	Yes	Yes	31.84	18.52

Table 4: Temporally Grouped Model - response time for different representations of *now*, 20 % of now-relative data

temporally grouped model has worse response time than the *Direct* model, despite having more expressive power and is more natural since it is history oriented. This is due to more complex structure of elements. Grouped model is slower due to the complexity of elements that consist of attributes `vstart` and `vend` and index can not be used efficiently.

Faster response time when there is no reference to the current time is partly due to the wrong and smaller answer size and because in case of referencing to the current time there is additional processing cost for build-in function `current-date()` and need for conversion of the date to the string.

Slightly better performance of the variable approach to represent current time in comparison to the *MAX* approach is due to the smaller length of string ‘now’ comparing to the ‘9999-12-31’, which causes smaller XML file size and also higher fanout of elements in index nodes. Also, computational cost for comparison is smaller due to the shorter length of the string ‘now’.

4 Conclusion and future work

This study makes the following contributions to the field:

- By investigating different representations of *now* in XML, we presented a better understanding of the significance of modelling current time, particularly how it influences the efficiency and accuracy;
- We identified available approaches for adding time dimension to XML;
- We empirically demonstrated that usage of user defined functions to handle current time in XML, which is previously recommended as favorite in the literature, is basically inefficient and is obviously not appropriate;
- We showed that the flat translation of temporal attributes to XML elements (*DIRECT* model), is more efficient than the usage of attributes within the XML elements.
- We identified available options to represent and store current time in native XML databases and empirically evaluated their suitability;
- We concluded that any approach to represent ‘now’ if not referencing to the current time yields wrong answer.
- We introduced the notion of storing variables such as ‘now’ or ‘UC’ as strings in XML native databases. Usage of variables at the same time yields the correct answer and query response time is good. Another advantage of using variables is clear semantics, the meaning of the word ‘now’ suggests of current time in contrast to ‘9999-12-31’ where the meaning is introduced by convention. For those reasons storing variables to represent current time can be considered as the most appropriate approach in XML temporal.

The present paper shows that native XML databases offer better support for temporal reasoning than relational databases and at the same time they support richer data models. As we have argued, temporal data is very frequent in real life application, thus we believe that native XML database will present a viable alternative to relational temporal database when complex time dependent data has to be manipulated and recorded. On the contrary due to the nature of XML data and the verbosity of XML, the response time of the Native XML temporal databases does no compare with the response time of relational databases. This also indicates the need for further research on efficient storage architecture and access methods for Native XML temporal databases.

We intend to work on more efficient access method for temporal XML data, based on the intrinsic nature and format of temporal data types in XML databases. We believe that the resulting access method will prove useful not only in dealing with XML temporal data, but also can be employed on XML data of different nature.

References

Amagasa, T., Yoshikawa, M., & Uemura, S. (2000), ‘A Data Model for Temporal XML Documents’, *In Proc. of 11th Intl’ Conf. on Database and Expert Systems Applications (DEXA 2000), London, England* .

Carey, M., Kiernan, J., Shanmugasundaram, J. & et al (2000), ‘XPERANTO: A middleware for publishing objectrelational data as XML documents’, *VLDB* .

Chawathe, S. S., Abiteboul, S. & Widom, J. (1999), ‘Managing Historical Semistructured Data’, *Theory and Practice of Object Systems* **5**(3), 143–162.

Clifford, J., Croker, A., Grandi, F. & Tuzhilin, A. (1995), ‘On temporal grouping’, *In Recent Advances in Temporal Databases, Springer Verlag* p. 194213.

Clifford, J., Croker, A. & Tuzhilin, A. (1994), ‘On Completeness of Historical Relational Query Languages’, **19**(1), 64–116.

Date, C., Darwen, H. & Lorentzos, N. (2002), *Temporal Data and the Relational Model*, Morgan Kaufmann.

eXist:http://exist.sourceforge.net/ (2004). *http://exist.sourceforge.net/

Fernandez, M., Tan, W. & Suciu, D. (2000), ‘Silkroute: trading between relations and XML’, **33**(16), 723–745.

Funderburk, J., Kiernan, G., Shanmugasundaram, J., Shekita, E. & Wei, C. (2002), 'XTABLES: Bridging Relational Technology and XML', *IBM Systems Journal* **41**(4).

Gao, D. & Snodgrass, R. T. (2003), 'Syntax, Semantics, and Query Evaluation in the τXQuery, Temporal XML Query Language', *TR-72 A TIMECENTER Technical Report* .

Grandi, F. & Mandreoli, F. (2000), 'The Valid Web: an XML/XSL Infrastructure for Temporal Management of Web Document', *In Proc. of the Intl' Conf. on Advances in Information Systems (ADVIS'2000), Izmir, Turkey* .

Grandi, F., Mandreoli, F., Tiberio, P. & Bergonzini, M. (2003), 'A Temporal Data Model and System Architecture for the Management of Normative Texts (Extended Abstract)', *Proc. SEBD 2003 - Natl' Conf. on Advanced Database Systems, Cetraro, Italy* pp. 169–178.

http://www.sqlx.org (2004).
*http://www.sqlx.org

Jensen, C. S. (2000), 'http://www.cs.auc.dk/ csj/Thesis/pdf/'.
*http://www.cs.auc.dk/ csj/Thesis/pdf/

Ozsoyoglu, G. & Snodgrass, R. (1995), 'Temporal and real-time databases: A survey', *IEEE Trans. On Knowledge and Data Engineering* **7**(4), 513–532.

Stantic, B., Khanna, S. & Thornton, J. (2004), 'An Efficient Method for Indexing Now-relative Bitemporal data', *In Proceeding of the 15th Australasian Database conference (ADC2004), Denidin, New Zealand* **26**(2), 113–122.

Torp, K., Jensen, C. S. & Bohlen, M. (1999), 'Layered implementation of temporal DBMS concepts and techniques', *A TimeCenter Technical Report TR-2* .

Wang, F. & Zaniolo, C. (2002), 'Preserving and Querying Histories of XML-published Relational Databases', *In Proc. of 2nd Intl' Workshop on Evolution and Change in Data Management (ECDM 2002), Tampere, Finland* pp. 26–38.

Wang, F. & Zaniolo, C. (2003), Temporal Queries in XML Document Archives and Web Warehouses, *in* 'In Proceeding of the 10th International Symposium on Temporal Representation and Reasoning (TIME-ICTL 2003), Cairns,Australia', pp. 47–55.

www3.ibm.com (2005), 'In DB2 XML Extender'.
*http://www3.ibm.com/software/data
/db2/extenders/xmlext/

xml.com (2005), 'Making the case for XML Native databases'.
*http://www.xml.com/pub/a/2005/03/30/
native.html

XQuery 1.0: An XML query language (2004).
*http://www.w3.org/TR/xquery/

A Heuristic Approach to Cost-Efficient Fragmentation and Allocation of Complex Value Databases

Hui Ma, Klaus-Dieter Schewe, Qing Wang

Massey University, Department of Information Systems
& Information Science Research Centre
Private Bag 11 222, Palmerston North, New Zealand,
Email: h.ma|k.d.schewe|q.q.wang@massey.ac.nz

Abstract

The quality of database distribution design, which involves fragmentation and allocation, should be assessed by the performance of a system. In particular, this applies to non-relational database systems. This paper addresses fragmentation and allocation in the context of complex value databases. Fragmentation and fragment allocation are performed simultaneously. For this we present a query processing cost model to evaluate the performance of the system. The core of the paper is a heuristic approach for fragmentation and fragment allocation, which uses the cost model and is targeted at globally minimising these costs. The validity of the approach is supported by experimental results.

Keywords: fragmentation, complex value database, allocation, query cost model, heuristics

1 Introduction

With the increasing demand of database applications that are accessed by users from different geographical locations, database distribution design becomes an essential part of the database design, which targets at increasing the overall system performance. From early 1980s the problem of database distribution design has attracted researchers interests. It has first been discussed in the context of the relational datamodel, then in the object oriented datamodel. With the current popularity of web information systems, there are increasing needs for distributed database systems (DDBMS) to provide back-end support for Web-based database applications. In particular this applies to non-relational database systems such as object oriented databases (Schewe & Thalheim 1993), object-relational databases (Thalheim 2000) or databases that are based on the eXtensible Markup Language (XML) (Abiteboul, Buneman & Suciu 2000), which are becoming more and more the standard models for advanced database applications. In this article we concentrate on one common aspect of these models, the presence of complex values.

In the context of the relational datamodel (RDM) distribution design mainly addressed the problems of schema fragmentation and allocation of the fragments to the machines in a computer network (Özsu & Valduriez 1999). Fragmentation is commonly divided into horizontal fragmentation, which splits a relation into disjoint unions, and vertical fragmentation, which projects a relation onto a subset of its attributes. Horizontal and vertical fragmentation have been discussed intensively in (Chu 1992, Chu & Ieong 1993, Khalil, Eid & Khair 1999, Lin, Orlowska & Zhang 1993), allocation in (Apers 1988) and also integrated approaches have been tried (Tamhankar & Ram 1998).

Several authors have approached the generalisation of fragmentation techniques to complex value and object oriented datamodels. For instance, horizontal fragmentation is discussed in (Bellatreche, Karlapalem & Simonet 2000, Ezeife & Barker 1995, Ma 2003, Schewe 2002), and vertical fragmentation in (Chinchwadkar & Goh 1999, Ezeife & Barker 1998, Malinowski & Chakravarthy 1997, Schewe 2002). Due to the similarity between object oriented and semi-structured data, some of these techniques have also been adapted to semi-structured data (Schewe 2002) and XML (Ma & Schewe 2003).

The aim of database distribution design is to make applications that access the database more efficient and effectively. Therefore, the global queries have to be analysed in order to design an adequate distribution of the data. In this paper we address the problem to design fragments and to allocate them in such a way that the overall performance of the distributed database system is better than the one of an equivalent centralised one. That is, we first develop a query cost model for complex value databases. Then we present a heuristic approach to minimise query costs for the case of horizontal fragmentation. We show that the minimisation of transportation costs is decisive, and that this can be achieved locally by either accepting or rejecting a horizontal fragmentation with a simple predicate that arises from one of the most frequent queries.

In Section 2 we present the basic definitions of a complex value datamodel that is adapted from the higher-order Entity Relationship model (HERM) from (Thalheim 2000). For this model we also describe a general query algebra following the general approach in (Schewe 2001). This leads to query trees which can be subject to algebraic query optimisation, for which we adapt the techniques from the RDM.

In Section 3 we discuss fragmentation operations, which we slightly generalise by taking an additional splitting operation into account (Schewe 2002). In Section 4 we then discuss a cost model and approach the optimisation of costs. In Section 5 we present some heuristics needed in the process of horizontal fragmentation and allocation. Section 6 shows some experimental results that validate our proposed heuristics. We conclude with a short summary in Section 7.

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

2 Complex Value Databases

In this section we present the basic definitions of a complex value datamodel following basically the work in (Thalheim 2000). Furthermore, we introduce a generic query algebra for this model, and discuss heuristic algebraic query optimisation.

2.1 The Datamodel

In order to define complex values we use a type system. Using abstract syntax, types can be defined by

$$t = b \mid (a_1 : t_1, \dots, a_n : t_n) \mid \{t\}.$$

Here b represents an arbitrary collection of *base types*, e.g. *BOOL* for boolean values **T** and **F**, *OK* for a single value *ok*, *PIC* for images, *MPIC* for video data, *CARD* and *INT* for numbers, *DATE* for dates, etc. Finally, $(\cdot)$ and $\{\cdot\}$ are constructors for records and finite sets, respectively.

On the basis of this type system we can define database schemata, which are sets of database types.

A *database type of level k* has a name E and consists of a set $comp(E) = \{r_1 : E_1, \dots, r_n : E_n\}$ of components with pairwise different role names r_i and database types E_i on levels lower than k with at least one database type of level exactly $k-1$, a set $attr(E) = \{a_1, \dots, a_m\}$ of attributes, each associated with a data type $dom(a_i)$ as its domain, and a key $id(E) \subseteq comp(E) \cup attr(E)$. We shall write $E = (comp(E), attr(E), id(E))$.

A *database schema* is a finite set $\mathcal{S}$ of database types such that for all $E \in \mathcal{S}$ and all $r_i : E_i \in comp(E)$ we also have $E_i \in \mathcal{S}$. That is schemata are closed under component references.

Example 2.1. The following database schema is adapted from the ODIN system (Feyer, Kao, Schewe & Thalheim 2000):

Department = (∅, {name, homepage, contact}, {name})
Course = (∅, {name, degree}, {name})
Lecturer = ({in:Department}, {name, position, homepage, email}, {name, in:Department})
Paper = (∅, {no, kind, name, level, description, regularity, points}, {no})
Prerequisite = ({of:Paper, for:Paper}, ∅, {of:Paper, for:Paper})
Lecture = ({goal:Paper}, {semester, schedule, literature, comment}, {goal:Paper, semester})
Teacher = ({who:Lecturer, for:Lecture}, ∅, {who:Lecturer, for:Lecture})
Contribution = ({for:Course, of:Paper}, {optional}, {for:Course, of:Paper})
Involved = ({who:Department, in:Course}, ∅, {who:Department, in:Course})

As a domain for an attribute we have $dom(\text{schedule}) = \{(\text{kind}{:}\mathit{STRING}, \text{time}{:}\mathit{TIME}, \text{day}{:}\mathit{STRING}, \text{room}{:}\mathit{STRING})\}$ in the database type Lecture. All other domains have been omitted.

Given a database schema $\mathcal{S}$ we associate two types $t(E)$ and $k(E)$ – called *representation type* and *key type*, respectively – with each $E = (\{r_1 : E_1, \dots, r_n : E_n\}, \{A_1, \dots, A_k\}, \{r_{i_1} : E_{i_1}, \dots, r_{i_m} : E_{i_m}, A_{j_1}, \dots, A_{j_\ell}\}) \in \mathcal{S}$:

- The representation type of E is the tuple type
  $t(E) = (r_1 : t(E_1), \dots, r_n : t(E_n), A_1 : dom(A_1), \dots, A_k : dom(A_k))$.
- The key type of E is the tuple type
  $k(E) = (r_{i_1} : k(E_{i_1}), \dots, r_{i_m} : k(E_{i_m}), A_{j_1} : dom(A_{j_1}), \dots, A_{j_\ell} : dom(A_{j_\ell}))$.

Finally, a *database db* over a schema $\mathcal{S}$ is an $\mathcal{S}$-indexed family $\{db(E)\}_{E \in \mathcal{S}}$ such that each $db(E)$ is a finite set of values of type $t(E)$ satisfying the following two conditions:

- whenever $t_1, t_2 \in db(E)$ coincide on their projection to $id(E)$, they are already equal;
- for each $t \in db(E)$ and each $r_i : E_i \in comp(E)$ there is some $t_i \in db(E_i)$ such that the projection of t onto r_i is t_i.

2.2 Query Algebra

According to our definition of complex value databases above the definition of a query algebra is mainly determined by operations defined on the type system. There is no operation on OK, but for $BOOL$ we may consider the operations $\wedge : BOOL \times BOOL \rightarrow BOOL$ (conjunction), $\neg : BOOL \rightarrow BOOL$ (negation) and $\Rightarrow : BOOL \times BOOL \rightarrow BOOL$ (implication). Furthermore, we consider two constants $\texttt{true} : 1\!\!1 \rightarrow BOOL$ and $\texttt{false} : 1\!\!1 \rightarrow BOOL$.

For tuple types we consider *projection* $\pi_i : t_1 \times \cdots \times t_n \rightarrow t_i$ and *product* $o_1 \times \cdots \times o_n : t \rightarrow t_1 \times \cdots \times t_n$ for given operations $o_i : t \rightarrow t_i$.

For set types we may consider $\cup$ (union), $-$ (difference), the constant $\texttt{empty} : 1\!\!1 \rightarrow \{t\}$ and the *singleton* operation $\texttt{single} : t \rightarrow \{t\}$ with well known semantics. In addition, we consider structural recursion, which exploits the constructors for finite sets, i.e. the constant $\texttt{empty}$, the singleton operation and the union operation. In order to define an operation on a set type, say $\texttt{op} : \{t\} \rightarrow t'$ it is therefore sufficient to define it on the empty set, on singleton sets and on unions.

Formally, we define $\texttt{op} = \texttt{src}[e, g, \sqcup]$ with a value e of type t', a function $g : t \rightarrow t'$ and a function $\sqcup : t' \times t' \rightarrow t'$. Then $\texttt{src}[e, g, \sqcup]$ is defined as follows:

$$\begin{aligned} \texttt{src}[e,g,\sqcup](\emptyset) &= e \quad , \\ \texttt{src}[e,g,\sqcup](\{x\}) &= g(x) \text{ for each } x \text{ of type } t \text{, and} \\ \texttt{src}[e,g,\sqcup](X \cup Y) &= \texttt{srcd}[e,g,\sqcup](X) \sqcup \texttt{src}[e,g,\sqcup](Y) \\ &\qquad \text{for disjoint } X, Y \text{ of type } \{t\}. \end{aligned}$$

It is easy to see that structural recursion is able to express projections, selections and aggregate functions as they appear in standard query languages such as SQL.

For completeness we also use operations on functions. In particular, we consider *composition* $\circ : (t_2 \rightarrow t_3) \times (t_1 \rightarrow t_2) \rightarrow (t_1 \rightarrow t_3)$, *evaluation* $\texttt{ev} : (t_1 \rightarrow t_2) \times t_1 \rightarrow t_2$, and *abstraction* $\texttt{abstr} : (t_1 \times t_2 \rightarrow t_3) \rightarrow (t_1 \rightarrow (t_2 \rightarrow t_3))$. All these operations are standard.

Furthermore, assume an equality predicate $=_t : t \times t \rightarrow BOOL$ for all types t except function types and a membership predicate $\in : t \times \{t\} \rightarrow BOOL$. We shall also use a unique "forget"-operation $\texttt{triv} : t \rightarrow 1\!\!1$ for each type t. Combining all the operations for all types of the type system gives all operations induced from the type system.

In (Schewe 2001) it has been shown that these operations suffice to express more complex operations for nesting and unnesting.

Finally, we will need a generalized join-operation. For this, we first define subtyping in the standard way as the smallest preorder such that the following holds:

- For any type t we have $t \leq OK$.
- For set types (or list types, multiset types, respectively) we have $\{t\} \leq \{t'\}$ (or $[t] \leq [t']$, $\langle t \rangle \leq \langle t' \rangle$, respectively) iff $t \leq t'$ holds.

- For tuple types we have $t_1 \times \cdots \times t_m \le t'_1 \times \cdots \times t'_n$ iff $t_{\sigma(i)} \le t'_i$ holds for some injective $\sigma : \{1,\dots,n\} \to \{1,\dots,m\}$.

Then each subtype relation $t \le t'$ defines an associated subtype function $\pi_{t'} : t \to t'$. In (Schewe 2001) the following result has been proven:

If t is a common supertype of t_1 and t_2 with associated subtype functions $\pi^i_t : t_i \to t$, then there exists a common subtype $t_1 \bowtie_t t_2$ together with subtype functions $\pi_{t_i} : t_1 \bowtie_t t_2 \to t_i$ such that $\pi^1_t \circ \pi_{t_1} = \pi^2_t \circ \pi_{t_2}$ holds. Furthermore, for any other common subtype t' with subtype functions $\pi'_{t_i} : t' \to t_i$ with $\pi^1_t \circ \pi'_{t_1} = \pi^2_t \circ \pi'_{t_2}$ there is a unique subtype function $\pi : t' \to t_1 \bowtie_t t_2$ with $\pi_{t_i} \circ \pi = \pi'_{t_i}$.

With the existence of the *join types* $t_1 \bowtie_t t_2$ the join over t can be defined as

$$C_1 \bowtie_t C_2 = \{z : T_{C_1} \bowtie_t T_{C_2} \mid \exists z_1 \in C_1 . \exists z_2 \in C_2 . \\ \pi_{t_1}(z) = z_1 \wedge \pi_{t_2}(z) = z_2\}.$$

2.3 Heuristic Query Optimisation

Using the query algebra, each query gives rise to a query tree in the same way as for the relational data model. Furthermore, there are equalities among the operators, which will allow us to rearrange the execution order of the operations. Without going into detail – the heuristics are the same as for the RDM – we can rearrange a query tree in a way that (if possible) we first apply structural recursion operations $src[e, g, \sqcup]$ on the sets of input database, i.e. on some $db(E)$.

In particular, selections and projections, i.e. those operations that will reduce the size of a set of complex values, can be expressed by structural recursion (Schewe 2001).

First consider a function $f : t \to t'$ for arbitrary types t and t'. We want to "raise" f to a function $\mathtt{map}(f) : \{t\} \to \{t'\}$ by applying f to each element of a set. Obviously, we have $\mathtt{map}(f) = \mathtt{src}[\emptyset, \mathtt{single} \circ f, \cup]$. Projection is just a special case of map.

Next consider a function $\varphi : t \to \mathit{BOOL}$. We define selection as an operation $\mathtt{filter}(\varphi) : \{t\} \to \{t\}$, which associates with a given set the subset of all elements "satisfying the predicate" φ, i.e. elements that are mapped to $\mathbf{T}$. Then we may write

$$\mathtt{filter}(\varphi) = \mathtt{src}[\emptyset, \mathtt{if\_then\_else} \circ (\varphi \times \mathtt{single} \times (\mathtt{empty} \circ \mathtt{triv})), \cup]$$

with the function $\mathtt{if\_then\_else} : \mathit{BOOL} \times t \times t \to t$ with $(\mathbf{T}, x, y) \mapsto x$ and $(\mathbf{F}, x, y) \mapsto y$.

3 Schema Fragmentation

Let us now introduce operations for fragmentation. Similar to the RDM horizontal fragmentation exploits the fact that each database type E defines a set $db(E)$ in a database db, thus can be partitioned into disjoint subsets. As our complex value datamodel provides deeply nested structures, we use splitting as another fragmentation operation. Roughly speaking, splitting introduces new components.

3.1 Horizontal Fragmentation

As in any database db the database type E is associated with a finite set $db(E)$, we obtain an easy generalisation of relational horizontal fragmentation. For this let E be some database type. Take boolean valued functions φ_i $(i = 1,\dots,n)$ such that for each database db we obtain

$$db(E_i) = \bigcup_{i=1}^{n} \mathtt{filter}(\varphi_i)(db(E))$$

with disjoint sets $\sigma_{\varphi_i}(db(E))$. We then replace E in the schema by n new database types E_i, all with the same definition as E.

Example 3.1. Take the schema from Example 2.1 and fragment the database type PAPER into two new instances ADVANCED_PAPER and BASIC_PAPER using $\varphi_1 \equiv \text{level} \ge 300$ and $\varphi_2 \equiv \text{level} < 300$.

3.2 Vertical Fragmentation

Let E be a database type with $k(E)$ as *key type*. It takes a form, $E = (\{r_1 : E_1,\dots,r_n : E_n\},\{A_1,\dots,A_k\},\{r_{i_1} : E_{i_1},\dots,r_{i_m} : E_{i_m}, A_{j_1},\dots,A_{j_\ell}\})$. Vertical fragmentation on E replace E with a set of new types $E_1,\dots,E_m$ with $E_j = (\{r^j_1 : E^j_1,\dots,r^j_n : E^j_n\},\{A^j_1,\dots,A^j_k\},\{r^j_{i_1} : E^j_{i_1},\dots,r^j_{i_m} : E^j_{i_m}, A^j_{j_1},\dots,A^j_{j_\ell}\})$ such that:

- the components and attributes will be distributed:
$$\{E_1,\dots E_n\} = \bigcup_{j=1}^{m} \{E^j_1,\dots,E^j_{n_i}\},$$
$$\{A_1,\dots A_k\} = \bigcup_{j=1}^{m} \{A^j_1,\dots,A^j_{n_i}\},$$
- $db(E)$ is split into $db(E_1),\dots,db(E_m)$ such that $db(E)$ could be reconstructed by using the join operation on all the instances:
$$db(E) = db(E_1) \bowtie \cdots \bowtie db(E_m),$$
- in the query algebra $E = E_i \bowtie \cdots \bowtie E_m$.

Using the query algebra, vertical fragmentation could be written as $db(E_j) = \mathtt{map}(\pi_{E_j})(db(E))$ for all $j \in \{1,\dots,m\}$. It normally requires that the key type $k(E)$ is part of all E_j.

Example 3.2. Take the schema from Example 2.1 and fragment the database type

LECTURER = ({in:DEPARTMENT}, {name, position, homepage, email}, {name, in:DEPARTMENT}) into two new types LECT_DETAILS and LECT_PAGE with

LECT_DETAILS=
({in:DEPARTMENT}, {name, position, email}),
LECT_PAGE =
({in:DEPARTMENT}, {name, homepage}).

Correspondingly the instance of LECTURER is fragmented using the map operation:

$$db(\text{LECT\_DETAILS}) = \mathtt{map}(\pi_{\text{LECTURER\_DETAILS}})(db(E))$$

$$db(\text{LECT\_PAGE}) = \mathtt{map}(\pi_{\text{LECTURER\_HOMEPAGE}})(db(E))$$

3.3 Fragmentation by Splitting

The splitting operation is quite simple. Suppose the schema contains a database type E and take subsets $\mathit{comp}'(E) \subseteq \mathit{comp}(E)$ and $\mathit{attr}'(E) \subseteq \mathit{attr}(E)$. Then simply add a new database type $E' = (\mathit{comp}'(E), \mathit{attr}'(E), \mathit{id}(E'))$ to the schema $\mathcal{S}$ and change the definition of E to

$$E_{new} = (\mathit{comp}(E) - \mathit{comp}'(E) \cup \{r' : E'\}, \mathit{attr}(E) - \mathit{attr}'(E), \mathit{id}'(E))$$

with $\mathit{id}'(E) = \mathit{id}(E) - \mathit{comp}'(E) - \mathit{attr}'(E) \cup \{r' : E'\}$ if $\mathit{id}(E) \cap (\mathit{comp}'(E) \cup \mathit{attr}'(E)) \ne \emptyset$, $\mathit{id}'(E) = \mathit{id}(E)$ otherwise, and $\mathit{id}(E') = \mathit{comp}'(E) \cup \mathit{attr}'(E)$.

Then $db(E)$ is reconstructed from $db(E_{new})$ and $db(E')$ by a dereferencing operation δ.

Example 3.3. Example 2.1 suffices to illustrate the splitting operation. We could split the database type Contribution into the types

$$\text{Contribution} = (\{\text{of:Paper}, r'\text{:Contribution}'\}, \\ \emptyset, \{\text{of:Paper}, r'\text{:Contribution}'\})$$

and

$$\text{Contribution}' = (\{\text{for:Course}\}, \{\text{optional}\}, \\ \{\text{for:Course}, \text{optional}\})$$

If we first apply splitting, we increase the number of database types in the schema. Thus, splitting widens the possibilities for applying horizontal fragmentations.

3.4 The Impact of Fragmentation on Optimised Query Trees

Using the query algebra presented in the last section we can write queries in the form of query trees. The "optimisation assumption" then allows us to assume that the leaves of such trees are the input document(s), and predecessors of leaves are structural recursion nodes, which represent first a selection expressed via `filter`, then a projection expressed via `map`. Equivalently, we may assume that we have subqueries of the form

$$\mathtt{map}(\pi_X)(\mathtt{filter}(\varphi)(db(E))) \qquad (*)$$

Horizontal fragmentation corresponds to replacing $db(E)$ by some union $db(E_1) \cup \cdots \cup db(E_n)$. We may assume that splitting has been executed first to obtain enough such horizontal fragments.

As the new sets $db(E_i)$ $(i = 1, \dots, n)$ are all expressed by selections, we may assume again $n = 2$ by switching to horizontal fragmentation with *normal predicates*, i.e. satisfiable conjunctions of simple selection formulae. That is, we basically replace $db(E)$ in the subquery (*) above by $db(E_1) \cup db(E_2)$.

Another round of query optimisation might shift the selection $\mathtt{filter}(\varphi)$ and the projection $\mathtt{map}(\pi_X)$ inside the newly introduced union, but the "upper part" of the query tree would not be affected. Therefore, in order to optimise horizontal fragmentation, it is decisive and sufficient to consider subqueries of the form (*) above.

Vertical fragmentation corresponds to replacing $db(E)$ by some join $db(E_1) \bowtie \cdots \bowtie db(E_n)$. Another round of query optimisation might shift the selection $\mathtt{filter}(\varphi)$ and the projection $\mathtt{map}(\pi_X)$ inside newly introduced join, but again the "upper part" of the quey tree would not be affected. Hence, it is also decisive and sufficient to consider subqueries in the form (*) above for the purpose of optimising vertical fragmentation.

Similarly, splitting fragmentation corresponds to replacing $db(E)$ by $\delta(db(E_{new}), db(E'))$. Query optimization might shift the selection $\mathtt{filter}(\varphi)$ and the projection $\mathtt{map}(\pi_X)$ inside the newly introduced dereference. At the same time the upper part of the query tree would also be affected. Again, we can just consider subqueryies in the form for optimising splitting operation.

4 A Cost Model

We now analyse the query costs in the case of horizontal fragmentation. The major objective is to base the fragmentation decision on the efficiency of the most frequent queries. As a general pragmatic guideline we follow the recommended rule of thumb to consider only the 20% most frequent queries, as these usually account for most of the data access (Özsu & Valduriez 1999).

4.1 Size Estimation

Crucial to the query costs are the sizes of sets of complex values that have to be built during query execution, as these sets have to be stored at secondary storage, retrieved from there again, and sent between the locations of a network. Therefore, we first approach an estimation of these sizes.

In order to do so, we look at the components $r : E' \in comp(E)$ and the attributes $A \in attr(E)$, and estimate the size $s(t(E))$, i.e. the average size of an element in $db(E)$. The size of $db(E)$ itself is then $n_E \cdot s(t(E))$, where n_E is the average number of elements in the set $db(E)$.

Let s_i be the average size of elements $dom(b_i)$ for a base type b_i. This can be used to determine first the size $s(A)$ of an attribute A, i.e. the average space needed for it in storage. We obtain:

$$s(A) = \begin{cases} s_i & \text{if } dom(A) = b_i \\ \sum_{i=1}^{n} s(t_i) & \text{if } dom(A) = (a : t_1, \dots, a_n : t_n) \\ r \cdot s(t) & \text{if } dom(A) = \{t\} \end{cases}$$

In the last of these cases r is the average number of elements in sets of type $\{t\}$ within an attribute A.

Then for $E = (\{r_1 : E_1, \dots, r_n : E_n\}, \{A_1, \dots, A_k\}, id(E))$ we obtain

$$s(t(E)) = \sum_{i=1}^{n} s(t(E_i)) + \sum_{j=1}^{k} s(A_i).$$

The calculation of sizes of class instances applies also to the intermediate results of all queries. However, we can restrict our attention to the nodes in the subqueries of the form (*), as the other nodes in the query tree will not be affected by horizontal fragmentation and subsequent heuristic query optimisation. Thus, we only have to look at selection and projection nodes and ignore all other nodes in query trees.

- The size of a selection node $\mathtt{filter}(\varphi)$ is $p \cdot s$, where s is the size of the successor node and p is the probability that an element in the successor will satisfy φ.
- The size of a projection node $\mathtt{map}(\pi_X)$ is $(1 - c) \cdot s \cdot \dfrac{s(t_X)}{s(t)}$ where t is the type of elements in the set associated with the successor node, and t_X is the type of the elements in the set associated with the projection node.

The work in (Ma 2003) contains a discussion of sizes of results for other algebra operations as well, but we will not need this here.

4.2 Query Processing Costs

Fragmentation of type E results in a set of fragments $\{E_1, \dots, E_n\}$ of average sizes $s_1, \dots, s_n$. If the network has a set of nodes $N = N_1, \dots, N_k$ we have to allocate these fragments to the nodes, which gives rise to a mapping $\lambda : \{1, \dots, n\} \to \{1, \dots, k\}$, which we call a *location assignment*.

However, the fragments only appear on the leaves of query trees. More generally, we must associate a node $\lambda(v)$ with each node v in each relevant query tree. $\lambda(v)$ indicates the node in the network, at which the intermediate query result corresponding to v will be stored.

Given a location assignment λ we can compute the total costs of query processing. Let the set of queries be $Q^m = \{Q_1, \dots, Q_m\}$. Query costs are composed of two parts: *storage costs* and *transportation costs*: $costs_\lambda(Q_j) = stor_\lambda(Q_j) + trans_\lambda(Q_j)$.

The storage costs give a measure for retrieving the data back from secondary storage, which is mainly determined by the size of the data. The transportation costs provide a measure for transporting between two nodes of the network.

The storage costs of a query Q_j depend on the size of the intermediate results and on the assigned locations, which decide the storage cost factors. It can be expressed as

$$stor_\lambda(Q_j) = \sum_h s(h) \cdot d_{\lambda(h)},$$

where h ranges over the nodes of the query tree for Q_j, $s(h)$ are the sizes of the involved sets, and d_i indicates the storage cost factor for node N_i $(i = 1, \dots, k)$.

The transportation costs of query Q_j depend on the sizes of the involved sets and on the assigned locations, which decide the transport cost factor between every pair of sites. It can be expressed by

$$trans_\lambda(Q_j) = \sum_h \sum_{h'} c_{\lambda(h')\lambda(h)} \cdot s(h').$$

Again the sum ranges over the nodes h of the query tree for Q_j, h' runs over the predecessors of h in the query tree, and c_{ij} is a transportation cost factor for data transport from node N_i to node N_j $(i, j \in \{1, \dots, k\})$.

Furthermore, for each query Q_j we assume a value for its frequency $freq_j$. The total costs of all the queries in Q^m are the sum of the costs of each query multiplied by its frequency:

$$\sum_{j=1}^{m} cost_\lambda(Q_j) \cdot freq_j.$$

In general, the distribution could be called optimal if we find a fragmentation and allocation schema such that the resulting total query costs are minimal. As this problem is practically incomputable, we suggest to use a heuristic instead.

4.3 The Impact of Horizontal Fragmentation on Query Costs

Let us now discuss the impact of fragmentation on query costs (Ma & Schewe 2005). We concentrated on horizontal fragmentation, but vertical fragmentation and splitting can be handled similarly. As distribution is intended to increase query performance, let us now ask which fragmentations are reasonable. As already stated we may concentrate on the most frequent queries only and ignore all others. Each such query defines a set of simple selection formulae. Obviously, we should restricted ourself in horizontal fragmentation based on normal predicates due to the limitation of the available space.

Therefore, let Φ^w denote the set of all simple selection formulae that are used within the selected set $Q^m = \{Q_1, \dots, Q_m\}$ containing the most frequent queries. Let $\mathcal{N}^w$ be the set of normal predicates defined by Φ^w. We address the optimisation of horizontal fragmentation under the constraint that we want to select a reasonable subset of $\mathcal{N}^w$, say $\mathcal{N}^y \subseteq \mathcal{N}^w$, such that query performance with fragments defined by $\mathcal{N}^y$ is (almost) optimal. We further assume that the storage cost factors d_i are the same for all nodes N_i of the network.

Let us first look at the effects to query costs of a fragmentation with a single simple formula. Basically we have to adapt query trees, take into account algebraic optimisation, the effect of which will be on the subqueries of the form (*), and find a modified location assignment λ that would reduce the query processing costs, provided such a λ exists. We have to distinguish three cases for this scenario.

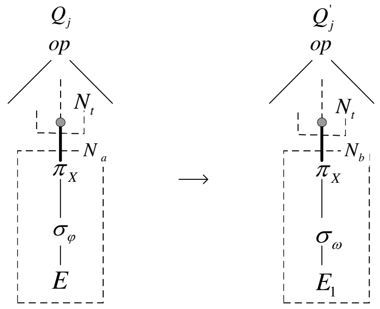


Figure 1: Scenario I for Query Tree Rewriting in Case of Horizontal Fragmentation

Scenario I

Assume that the selection formula φ in (*) has the form $\varphi = \psi \wedge \omega$. Using simple predicates ψ to perform fragmentation on $db(E)$ in (*) we get two fragments: $db(E_1) = \texttt{filter}(\psi)(db(E))$ and $db(E_2) = \texttt{filter}(\neg\psi)(db(E))$ with $db(E) = db(E_1) \cup db(E_2)$. Then query Q_j only needs to access fragment $db(E_1)$ and should be rewritten as

$$Q_j = op(\dots(\texttt{map}(\pi_X)(\texttt{filter}(\omega))(db(E_1))))$$

Correspondingly, on the query tree the leave $db(E)$ would be replaced by the fragment $db(E_1)$; and its predecessor in the query tree would become $\texttt{filter}(\omega)$. Here we neglect the special case that ω is *true*, in which case the selection would completely disappear. Consequently, allocation of the subquery tree would be changed from N_a to some N_b as shown Figure 1. There is no further change to the query tree.

Let p be the possibility that the objects in $db(E)$ satisfy the condition ψ then we have $s_{E_1} = p \cdot s_E$. If we know the size of selection node $\texttt{filter}(\omega)(E_1)$ or size of selection node $\texttt{filter}(\omega \wedge \psi)(db(E))$ then we get

1. the leaf node reduced in size by factor p, and
2. one internal node may be reduced to 0, i.e. not existing anymore.

Assume the storage cost factors d_i for all sites among the network are equal, the effect on storage costs is:

$$stor(Q_j') = stor(Q_j) - (1-p) \cdot s_E$$

Lets use $s_\mathcal{G}$ to represent the size of the node $\texttt{map}(\pi_X)(\texttt{filter}(\varphi)(db(E)))$ of the query tree Q_j. Assume the processor of π_X is allocated to node N_t, then only the transport cost is changed from $s_\mathcal{G} \cdot c_{at}$ to $s_\mathcal{G} \cdot c_{bt}$, if E_1 is allocated to a site b which is different from site a to which E is allocated. Hence we get:

$$trans(Q_j') = trans(Q_j) - c_{at} \cdot s_\mathcal{G} + c_{bt} \cdot s_\mathcal{G}$$

From the cost analysis above we make the following observations:

- The size associated with the projection node does not change. Therefore, if the location assignment λ was optimal before the fragmentation, it does not make sense to change it for any node other than the three nodes corresponding to the subquery (*).
- Unless the selection node is deleted from the query tree, its size remains unchanged. This implies that there is at most a small change to the storage costs. In particular, as transportation

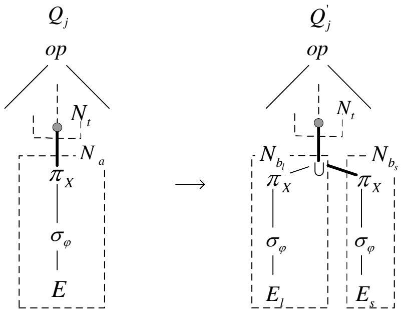


Figure 2: Scenario II for Query Tree Rewriting in Case of Horizontal Fragmentation

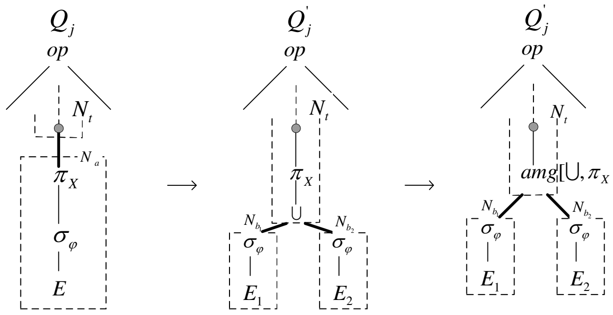


Figure 3: Scenario III for Query Tree Rewriting in Case of Horizontal Fragmentation

costs are assumed to be larger anyway, we may neglect this effect on the storage costs.

- The processing of the subquery itself does not require any transport of data. Therefore, we can assume that λ assigns the same network node to these nodes in the subquery tree.

Scenario II

Assume now that fragmentation with the selection formula ψ' leads to two fragments E_1 and E_2, i.e. the query Q_j needs data from both fragments. Then (*) would be replaced by

$$\texttt{map}(\pi_X)(\texttt{filter}(\varphi)(db(E_1) \cup db(E_2))). \qquad (\dagger)$$

Algebraic query optimisation will move the selection and projection inside the union, i.e. we obtain

$$\begin{aligned}&\texttt{map}(\pi_X)(\texttt{filter}(\varphi)(db(E_1)))\\ &\qquad \cup\texttt{map}(\pi_X)(\texttt{filter}(\varphi)(db(E_2))). \qquad (\ddagger)\end{aligned}$$

As a consequence, the fragment allocation can only require that the network nodes assigned to the two new fragments are different and maybe also be different from the target network node N_t. Then the fragment with the larger size after the selection will determine the location assignment for the union, unless the fragment with the smaller size after the selection is at the target site N_t, in which case the larger fragment should be moved to site N_t. Figure 2 shows the changes of the query tree with site allocation.

The effect of the horizontal fragmentation on the storage costs is only extra costs for storing a union node:

$$stor(Q'_j) = stor(Q_j) + s_E.$$

Using the same argument as for Scenario I we may again neglect this change to the storage costs.

Let b_1 be the site allocated to E_1, b_2 the one for E_2. Let l be 1 or 2 such that size s_{jl} of node $\texttt{map}(\pi_X)(\texttt{filter}(\omega)(db(E_l)))$ is maximal, s_{js} be the size of the smaller fragments. The effect of the fragmentation on the transportation costs is different in the following two situations:

- If $N_s \neq N_t$, then $trans(Q'_j) = trans(Q_j) - c_{at} \cdot s_{\mathcal{G}} + c_{b_l t} \cdot s_{\mathcal{G}} + c_{b_s b_l} s_{js}$.
- If $N_s = N_t$, then $trans(Q'_j) = trans(Q_j) - c_{at} \cdot s_{\mathcal{G}} + c_{b_l b_s} s_{jl}$.

We observe the following:

- As in case I the size of data transported to the target note N_t does not changed because the size of the union node in Q'_j is the same as the projection node in Q_j. Also, we obtain again marginal changes to the storage costs. Thus, we can concentrate on the transportation costs.
- The processing of the subquery would require sending the smaller selection result to the location of the larger one if the fragment of smaller size is not allocated to the target site N_t.

Scenario III

Taking the same assumption as for Scenario II, we now consider a special case when the sizes of $\texttt{filter}(\varphi)(db(E_i))$ are almost equal for $i = 1, 2$, and that the projection has only a small impact on the size of the result of (†). Then we get $s_{j1} \approx s_{j2} \approx \frac{s_E}{2}$.

In this case it is advantageous to read both fragments $\texttt{filter}(\varphi)(db(E_1))$ and $\texttt{filter}(\varphi)(db(E_2))$ and to transfer both to the same node, at which the projection operation will be performed after the union of the two fragments. We may even assume that the union and the follow-on projection can be combined in a single computation, $amg(\cup, \pi)$, which can reduce the costs of sorting. The result on the query tree is illustrated in Figure 3.

As in the two other cases, we may neglect the changes to the storage costs. Then the total query costs are:

$$\begin{aligned}trans(Q'_j) &= trans(Q_j) - c_{at} \cdot s_{\mathcal{G}} + c_{b_1 t} \cdot s_{j1} + c_{b_2 t} \cdot s_{j2}\\ &\approx trans(Q_j) + (\tfrac{c_{b_1 t} + c_{b_2 t}}{2} - c_{at}) \cdot s_{\mathcal{G}}.\end{aligned}$$

We observe that when both fragments are needed by Q_j and allocated the same site as E, there is no change for the query costs.

5 A Heuristic Approach for Horizontal Fragmentation and Allocation

The first step in the design of fragmentation is acquiring application information to determine a set of *simple predicates*, which takes the form *path* θ *v* with a path expression *path* on type E, a value of the corresponding type, and a comparison operator θ, which can be one of $=, \neq, \leq, <, \geq, >, \supseteq, \subset, \supseteq, \supset, \in, \ni, \notin$, and $\not\ni$.

Let $\Phi^m = \{\varphi_1, \ldots, \varphi_m\}$ denote a set of simple predicates defined on type E. Then the set of *normal predicates* $\mathcal{N}^m = \{\mathcal{N}_1, \ldots, \mathcal{N}_n\}$ on type E is the set of all satisfiable predicates of the form

$$\mathcal{N}_j \equiv \varphi_1^* \wedge \cdots \wedge \varphi_m^*$$

where φ_i^* is either φ_i or $\neg\varphi_i$.

Each of the normal predicates defines a *atomic horizontal fragment*,

$$F_j = \texttt{filter}(\mathcal{N}_j)(db(E))$$

Let $J \subseteq \{1, \ldots, m\}$ be a subset of all simple predicates. Normal predicates can therefore be written as:

$$\bigwedge_{j \in J} P_j \wedge \bigwedge_{j \notin J} \neg P_j$$

Let h be a network node, $J_h = \{j | j \in J \wedge P_j \text{ executed at site } h\}$ be all simple predicates executed at site N_h. The *request* of a atomic fragment at site h are:

$$request_h(F_j) = \sum_{j=1, j \in J_h}^{N} freq_j$$

with $freq_j$ as the frequency of query Q_j.

Note that the biggest number of atomic fragments can be up to 2^x with x as the number of simple predicates using for fragmentation. As we know the biggest number of fragments of a type E is the number of network node k. Therefore, atomic fragment recombination is a necessary step for getting a final horizontal fragmentation schema. In the following, we introduce a heuristic that performs the task of horizontal fragmentation, atomic fragments recombination as well as fragment allocation simultaneously.

5.1 A Heuristic for Horizontal Fragmentation and Fragment Allocation

According to our discussion above about the three basic cases of how fragmentation affects the query costs, the allocation of fragments to network nodes following the cost minimisation heuristics, already determines the location assignment, provided that an optimal location assignment for the queries was given prior to the fragmentation. This suggests the following horizontal fragmentation and allocation Heuristic.

Algorithm: *HF_Frag_Alloc*
Input: $\Phi^x = \{\varphi_1, \dots, \varphi_x\}$ /* a set of simple predicates
Output: Horizontal fragmentation schema and fragment allocation schema

```
Begin
    for each $h \in \{1, \dots, k\}$
        $E_h = \emptyset$
    end-for
    define a set of normal predicates $\mathcal{N}^x$ using $\Phi^x$
    define a set of atomic fragments $\mathcal{F}^x$ using $\mathcal{N}^x$
    for each atomic fragment $F_j \in \mathcal{F}^x, 1 \le i \le 2^x$ do
        for each node $h \in \{1, \dots, k\}$ do
            calculate $request_h(F_j)$
        end_for
        choose $w$ such that $request_w(F_i) =$
            $Max(request_1(F_i), \dots, request_k(F_j))$
                        /* find the maximum value
        $\lambda(F_j) = \mathcal{N}_w$    /* allocate the atomic fragment
                        to the site of biggest request
        define $E_h$ with $E_h = \bigcup\{F_j | \lambda(F_j) = N_h\}$
        /* put the atomic fragment into the
                        corresponding fragment
    end_for
End {HF_Frag_Alloc}
```

The above algorithm first finds the site that has the biggest value of *request* then allocates the atomic fragment to the site. A fragmentation schema and fragment allocation schema can be obtained simultaneously. Note that heuristics have to be employed, because the complexity of finding an optimal allocation schema is NP-hard due to the fact that there are as many as $k^{(2^x)}$ atomic fragments where x is the number of simple predicates and k is the number of network nodes.

5.2 An Allocation Heuristic for Distributed Query Trees

Site allocation of a query tree is performed from bottom to top. Horizontal fragmentation results leaf E on all query trees replaced with $E_1 \cup \dots \cup E_n$, if fragments E_i is relevant to the query. Another round of query optimization is performed to further optimise all the queries. In this section, we introduce an *allocation heuristic* for the allocation of distributed query trees.

Firstly, the root of a query tree is allocated to the site that issued it. Secondly, we do allocation for each leave of the query tree. A leaf of query trees is either a horizontal fragment or type, which is not fragmented. If leaves are horizontal fragments, allocation is decided by *HF_Frag_Alloc* introduced above. If a leaves is type, its allocation is decided by the needs of queries from it, i.e. to allocate the instance of a type to the site that have the highest value of needs of all queries calculated by

$$needs_h(E) = \sum_{j=1, N_j = h}^{N} s_j \cdot freq_j$$

with s_j as the size of data volume required by query Q_j and $freq_j$ as the frequency of Q_j, $j = 1, \dots, N$.

To allocate sites to intermediate nodes, following situations may occur:

- if there is a rooted path having two ends at the same site, allocate all nodes on the path at that site.
- if a node has one successor or two successors at the same site, allocate it to the site of its successor or successors.
- if a node has two successors at different sites, allocate the node the same site of the successor of bigger size.

5.3 A Heuristic Procedure of Primary Horizontal Fragmentation

The discussion about how horizontal fragmentation affect query costs in (Ma & Schewe 2005) shows that fragmentation of a type using a predicate φ in Φ will lead the follow two cases:

- One of the two fragments resulting from horizontal fragmentation with a formula in Φ will reside at the same location as the document before fragmentation, whereas the other fragment will be moved to a new location.
- Both fragments will reside at the same node.

In the second case the transportation costs remain the same. In the first case the transportation costs will be reduced. This suggests to take a total order on the elements of Φ according to their frequency, i.e. let $\Phi^w = [\varphi_1, \dots, \varphi_w]$ with $freq(\varphi_i) \ge freq(\varphi_j)$ iff $i \le j$. Then determine $\Phi^y = [\varphi_1, \dots, \varphi_y]$ such that fragmentation with elements in Φ^y leads to a reallocation of a fragment, whereas the fragmentation with elements in $\Phi^w - \Phi^y$ does not add changes. Then y can be determined by binary search.

For a given database schema $\mathcal{S} = \{E_1, \dots, E_i, \dots, E_n\}$, there is a set of queries, $Q^m = \{Q_1, \dots, Q_m\}$, that access the database 20% most frequently or are used by the most critical transactions. A heuristic procedure of horizontal fragmentation is proposed in (Ma & Schewe 2004*a*) to get a reasonable fragmentation schema by looking at most frequently used simple predicates. We now adopt the heuristic to the complex value databases. Based on the cost model introduced above a heuristic procedure of horizontal fragmentation for complex value databases includes the following steps:

1. sort queries by decreasing frequency to get a list of queries $\mathcal{Q} = [Q_1, \dots, Q_j, \dots, Q_m]$
2. optimise all the queries and extract simple predicates from the queries to get a list of Φ of simple predicates,
3. construct a usage matrix based on Φ to obtain a list of simple predicates Φ^w for each type E,
4. take the list of selection predicates $\Phi^w = [\varphi_1, \dots, \varphi_w]$ to get a list of indices $X = [0, 1, \dots, x_1, \dots, x_2, \dots, w]$,
5. determine y, $1 \leq y \leq n$, such that fragmentation with the first y predicates in Φ^w leads to a re-allocation of a fragment, whereas the fragmentation with elements in φ_{y+1} does not add changes. y can be determined by binary search using *Num_Simple_Predicates*,
6. take the first y simple predicates in Φ^w to get a subset of simple predicates Φ^y,
7. perform horizontal fragmentation with Φ^y using *PF_Frag_Alloc*. This results a fragmentation schema of type E and allocation schema of its fragments.

5.4 Simple Predicates for Horizontal Fragmentation

We first introduce an algorithm, *Cost_PH_Fragmentation*, for calculating total query cost using the cost model, with following parameters used in the algorithm: $Q^m = \{Q_1, \dots, Q_m\}$ as a set of 20% most frequent queries, $\Phi^w = [\varphi_1, \dots, \varphi_w]$ as a list of simple predicate abstracted from Q^m, $X = [0, 1, \dots, x_1, \dots, x_2, \dots, w]$ as the list of indices corresponding to Φ^w, x as a index of a simple predicate with $x \in X$, $cost_x$ as the total query costs when E is fragmented using the first x simple predicates in Φ^w, s_{ji} as the size of data volume needed from fragment E_i of E by Q_j.

```
Algorithm: Cost_PH_Fragmentation
Input: $Q^m = \{Q_1, \dots, Q_m\}$
       $\Phi^w = [\varphi_1, \dots, \varphi_w]$
       $x$
Output: $cost_x$
Begin
  take the first $x$ simple predicates in $\Phi^w$ to get $\Phi^x$
  apply $HF\_Frag\_Alloc$ to get a horizontal
        fragmentation schema of $E$ and allocation
        schema of its fragments $E_1, \dots, E_n$
  for each query tree $Q_j \in Q^m$ do
     replace $E$ with $E_1 \cup \dots \cup E_n$ in the query
           trees if $s_{ji} > 0$
     optimise all query trees using heuristic query
           optimisation
     apply the allocation heuristic to allocate
           intermediate nodes of query trees
     calculate query cost $cost_\lambda(Q_j)$ for each
           query using the cost model
  end-for
  calculate total query cost using the cost model:
        $cost_x = \sum_{j=1}^{m} cost_\lambda(Q_j) \cdot freq_j$
End {Cost_PH_Fragmentation}
```

Using the *Cost_PH_Fragmentation* algorithm above, an algorithm for finding the number y of simple predicates that should be used for horizontal fragmentation is presented as below.

```
Algorithm: Num_Simple_Predicates
Input: $Q^m = \{Q_1, \dots, Q_m\}$
       $\Phi^w = [\varphi_1, \dots, \varphi_w]$
       $X = [0, 1, \dots, x_1, \dots, x_2, \dots, w]$
Output: $y$
Begin:
  set $a = 0$, $b = w$
  do while $b - a \geq 3$
   randomly choose $x_1, x_2$ from $X$ such that
               $0 < x_1 < x_2 < b$
     for each $x \in \{a, x_1, x_2, b\}$ do
           $Cost\_PH\_Fragmentation$
     end-for
     $min := Min\{cost_a, cost_{x_1}, cost_{x_2}, cost_b\}$
       /* find the minimal among the four values
       if $cost_{y_1} = cost_{y_2}$ and $y_1 < y_2$ with
            $y_1, y_2 \in \{a, b, x_1, x_2\}$ then
          $min := cost_{y_1}$
       end-if
     if $cost_a = min$ then
        $b := x_1$
     else-if $cost_{x_1} = min$ then
        $b := x_2$
     else-if $cost_{x_2} = min$ then
        $a := x_1$
     else $cost_b = min$ then
        $a := x_2$
   end-do
   choose $y \in \{a, \dots b\}$ such that
          $cost_y = Min\{cost_x : x \in \{a, \dots, b\}\}$
End{Num_Simple_Predicates}
```

Basically, the above algorithm takes as input a list of indices of simple predicates, iteratively chooses four numbers a, b, x_1, x_2 with $a < x_1 < x_2 < b$, starting with $a = 0$ and $b = w$, calculates the corresponding total query costs, compares the costs to decide a reasonable number y of simple predicates for horizontal fragmentation.

6 Experimental Evaluation of the Heuristics

We present here some experiments that are conducted to verify the algorithms, *HF_Frag_Alloc* and *Num_Simple_Predicates* proposed above. We first build up a test bed. For this we design a database schema $\mathcal{S}$ and populate the database with records to get $db(\mathcal{S})$. Then we assume from four sites over a network there are 30 queries, which are the 20% most frequently queries or used by most critical transactions. We design these 30 queries by applying the similar pattern of queries as in OO7 project (Carey, DeWitt & Naughton 1993). According to the well-known 20/80 rule, the system performance is assessed by the total query costs of these 30 queries. There is one type $E \in \mathcal{S}$, on which there are a set Φ of 14 simple predicates abstracted from the 30 queries.

Before testing *Num_Simple_Predicates* we need first to validate the algorithm *HF_Frag_Alloc* because *HF_Frag_Alloc* is used by *Num_Simple_Predicates*. For this purpose, we fragment the type E by using all 14 simple predicates, as we did traditionally. The fragmentation results 42 atomic fragments. It is impossible to get an optimal allocation schema for the atomic fragments because the complexity of it is $4^{42} \approx 10^{26}$, which is practically incomputable. Therefore, to allocate the resulting set of atomic fragments we try the following two different allocation strategies and compare the total query costs from them:

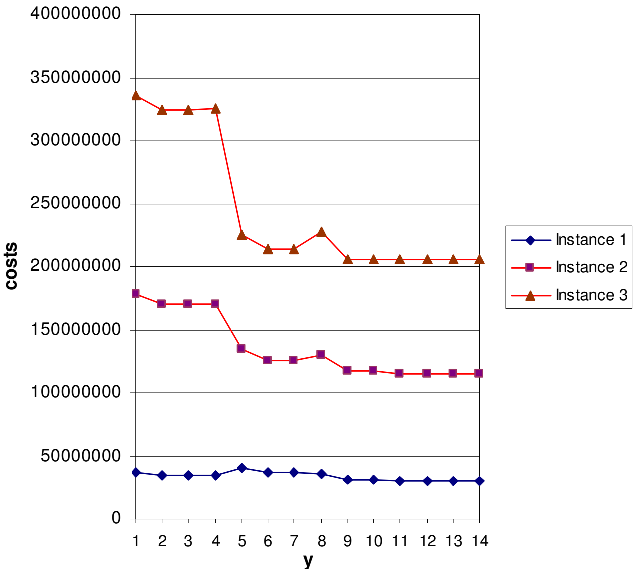


Figure 4: A Chart of Simple Predicate Query Costs

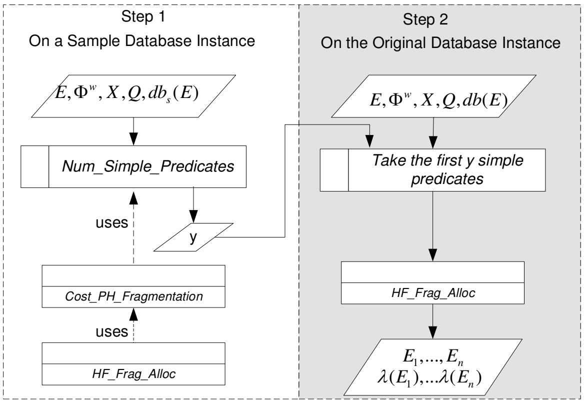

Step 1
On a Sample Database Instance
$E, \Phi^w, X, Q, db_s(E)$
Num_Simple_Predicates
uses
y
Cost_PH_Fragmentation
uses
HF_Frag_Alloc
Step 2
On the Original Database Instance
$E, \Phi^w, X, Q, db(E)$
Take the first y simple predicates
HF_Frag_Alloc
$E_1, \ldots, E_n$
$\lambda(E_1), \ldots \lambda(E_n)$

Figure 5: A Fragmentation Design Model

- get an allocation schema of each atomic fragment by using *HF_Frag_Alloc* and get the corresponding total query costs.
- get a set of allocation schemata by randomly allocate each atomic fragments to a site over the network and calculate total query costs under each of the allocation schemata. Choose the least query costs among them.

The experimental results show that the total query costs from both strategies are same. This means that the heuristic *HF_Frag_Alloc* can leads to semi_optimal allocation schema. We conclude that given a set of simple predicates to do horizontal fragmentation, allocating each atomic fragment to a site that requests it most frequently leads to a fragment allocation schema with nearly least total query costs.

To valid the algorithm *Num_Simple_Predicates*, we first calculate total query costs with respect to every value $x \in \{1, 2, \ldots, 14\}$, the number of simple predicates from Φ^w that is used for fragmenting a type E, on three different database instances that have the same selectively with respect to each simple predicate in Φ^w but with different sizes. The results are shown in Figure 6. We note that the trends of all the three graphs are similar, i.e. the lines become flat from $y = 11$.

We then test the *Num_Simple_Predicates* algorithm on the 3 different instances and get the following results:

test	1	2	3
instance size	1,000	5,000	10,000
y	11	11	11

From the above results we conclude that:

- given a set of simple predicates Φ^w, we only need a subset Φ^y of simple predicates with $\Phi^y \in \Phi^w$ to perform horizontal fragmentation such that the system performance can be improved in the same way as using all the simple predicates in Φ^w.
- the heuristic is efficient in the sense of rapidly getting the value y.
- with the increase of sizes of the instances, the query costs vary obviously with the change of the number of simple predicates for fragmentation, i.e., the more simple predicates the less total query costs. But total query costs stop changing from a certain point.
- with the decrease of the sizes of sample database instances the speed of running the algorithms increases tremendously.

This suggests that to fragment a instance $db(E)$ we can first get a value of y by applying the *Num_Simple_Predicates* algorithm on a small sample instance, which have the same selectivity as the original instance regarding the set of simple predicates. Then we use the first y simple predicates to do fragmentation and fragment allocation on the original database instance. Note that with the reduce of number of simple predicates for fragmentation, the complexity of the following up allocation problem can be reduced, i.e, from k^{2^w} to k^{2^y} with w as the number of simple predicates abstracted from the most frequent 20% queries. This procedure is depicted in Figure 6.

It takes as input a given type E, a set of simple predicates Φ^w on it, the corresponding simple predicate indices X of simple predicates, the set of queries Q^m considered for system performance assessment, uses *Num_Simple_Predicates* on a sample database instance to find out the value y. Once get the value y, we use *HF_Frag_Alloc* on the original database instance to perform fragmentation. The outputs of the procedure are a set of fragments $E_1, \ldots, E_n$ of type E, with $E = E_1 \cup \cdots \cup E_n$, and a allocation schema of all the fragments. Note that the algorithm *Num_Simple_Predicates* uses algorithm *Cost_PH_Fragmentation* which in turn uses algorithm*HF_Frag_Alloc*.

7 Conclusion

In this paper we continued our work on distribution design for complex value databases from (Ma 2003, Schewe 2002, Ma & Schewe 2004*b*, Ma & Schewe 2005). We presented slightly generalised horizontal fragmentation operations. The core of the work in this paper, however, was a detailed query performance cost model and a set of heuristics for fragmentation and allocation. We addressed the problem to design fragments and to allocate them in such a way that the overall performance of the distributed database system can be improved in the same way as using all the simple predicates abstracted from the most 20% queries.

The query cost model is directly based on a query algebra derived from the work in (Schewe 2001) which does not assume unnesting or any other relational representation. In addition, to the presentation of a heuristic approach to minimise query costs for the case of horizontal fragmentation, we presented some experiments that valid the heuristic approach. We showed that the minimisation of transportation costs is decisive, and that this can be achieved locally by either accepting or rejecting a horizontal fragmentation with a simple predicate that arises from one of the most frequent queries.

The next natural step in our work is a generalisation from horizontal fragmentation to fragmentation in general, for this we will discuss the effects of vertical fragmentation, splitting and replication on the query costs using the same query cost model. Of course, considering the requirement of global optimisation, we need to integrate the handling of horizontal and vertical fragmentation. Further, we will have more rigorous experiments to validate the allocation heuristic by studying the performance against the optimal case for a small set of queries over some types in the database schema. Another planned extension is to provide more rigorous mathematical support for the allocation and fragmentation heuristics, though it is well known that this is a very hard task.

Acknowledgement

Authors are grateful for valuable comments from Sven Hartmann, at Massey university, on this paper and during the research.

References

Abiteboul, S., Buneman, P. & Suciu, D. (2000), *Data on the Web: From Relations to Semistructured Data and XML*, Morgan Kaufmann Publishers.

Apers, P. M. G. (1988), ‘Data allocation in distributed database systems’, *ACM Trans. Database Syst.* **13**, 263–304.

Bellatreche, L., Karlapalem, K. & Simonet, A. (2000), ‘Algorithms and support for horizontal class partitioning in object-oriented databases’, *Distributed and Parallel Databases* **8**(2), 155–179.

Carey, M. J., DeWitt, D. J. & Naughton, J. F. (1993), ‘The OO7 benchmark’, *SIGMOD Record (ACM Special Interest Group on Management of Data)* **22**(2), 12–21.

Chinchwadkar, G. S. & Goh, A. (1999), ‘An overview of vertical partitioning in object oriented databases’, *The Computer Journal* **42**(1).

Chu, P.-C. (1992), ‘A transaction oriented approach to attribute partitioning’, *Information Systems* **17**(4), 329–342.

Chu, P.-C. & Ieong, I. T. (1993), ‘A transaction-based approach to vertical partitioning for relational databases’, *IEEE Transactions on Software Engineering* **19**(8), 804–812.

Ezeife, C. I. & Barker, K. (1995), ‘A comprehensive approach to horizontal class fragmentation in a distributed object based system’, *Distributed and Parallel Databases* **3**(3), 247–272.

Ezeife, C. I. & Barker, K. (1998), ‘Distributed object based design: Vertical fragmentation of classes’, *Distributed and Parallel Databases* **6**(4), 317–350.

Feyer, T., Kao, O., Schewe, K.-D. & Thalheim, B. (2000), Design of data-intensive web-based information services, *in* Q. Li, Z. M. Ozsuyoglu, R. Wagner, Y. Kambayashi & Y. Zhang, eds, ‘Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE 2000)’, IEEE Computer Society, pp. 462–467.

Khalil, N., Eid, D. & Khair, M. (1999), Availability and reliability issues in distributed databases using optimal horizontal fragmentation, *in* T. J. M. Bench-Capon, G. Soda & A. M. Tjoa, eds, ‘Database and Expert Systems Applications’, Vol. 1677 of *Lecture Notes in Computer Science*, Springer, pp. 771–780.

Lin, X., Orlowska, M. & Zhang, Y. (1993), ‘A graph-based cluster approach for vertical partitioning in databases systems’, *Data & Knowledge Enginееring* **11**(2), 151–170.

Ma, H. (2003), Distribution design in object oriented databases, Master’s thesis, Massey University.

Ma, H. & Schewe, K.-D. (2003), Fragmentation of XML documents, *in* ‘Proceedings XVIII Simpósio Brasileiro de Bancos de Dados (SBBD 2003)’, Manaus, Brazil, pp. 200–214.

Ma, H. & Schewe, K.-D. (2004*a*), A heuristic approach to horizontal fragmentation in object oriented databases, *in* J. Barzdins, ed., ‘Proceedings of the 2004 Baltic Conference on Databases and Information Systems’, Riga, Latvia, pp. 31–46.

Ma, H. & Schewe, K.-D. (2004*b*), Query cost analysis for horizontally fragmented complex value databases, *in* ‘Proc. 3rd Chilenean Database Workshop’.

Ma, H. & Schewe, K.-D. (2005), Query optimisation as part of distribution design for complex value databases, *in* Y. Kiyoki, H. Kangassalo, H. Jaakkola & J. Henno, eds, ‘Proc. 15th European-Japanese Conference on Information Modelling and Knowledge Bases’, Tallinn University of Technology, Estonia, pp. 269–276.

Malinowski, E. & Chakravarthy, S. (1997), Fragmentation techniques for distributing object-oriented databases, *in* D. W. Embley & R. C. Goldstein, eds, ‘Conceptual Modeling - ER ’97’, Vol. 1331 of *Lecture Notes in Computer Science*, Springer, pp. 347–360.

Özsu, M. T. & Valduriez, P. (1999), *Principles of Distributed Database Systems*, Alan Apt, New Jersey.

Schewe, K.-D. (2001), On the unification of query algebras and their extension to rational tree structures, *in* M. Orlowska & J. Roddick, eds, ‘Proc. Australasian Database Conference’.

Schewe, K.-D. (2002), Fragmentation of object oriented and semi-structured data, *in* H.-M. Haav & A. Kalja, eds, ‘Databases and Information Systems II’, Kluwer Academic Publishers, pp. 1–14.

Schewe, K.-D. & Thalheim, B. (1993), ‘Fundamental concepts of object oriented databases’, *Acta Cybernetica* **11**(4), 49–84.

Tamhankar, A. M. & Ram, S. (1998), ‘Database fragmentation and allocation: An integrated methodology and case study’, *IEEE Transactions on Systems Management* **28**(3), 194–207.

Thalheim, B. (2000), *Entity-Relationship Modeling: Foundations of Database Technology*, Springer-Verlag.

OCP- A Distributed Real Time Commit Protocol

Udai Shanker* Manoj Misra Anil K. Sarje

Department of Electronics & Computer Engineering
Indian Institute of Technology Roorkee
Roorkee-247 667 India

udaigkp@gmail.com,{manojfec, sarjefec}@iitr.ernet.in

Abstract

Most of the existing commit protocols try to improve the system performance by allowing a committing cohort to lend its data to an executing cohort, thus reducing data inaccessibility. However, these protocols block the borrower when it tries to send WORKDONE/PREPARED message (Qin & Liu 2003, Haritsa, Ramamritham & Gupta 2000, Gupta, Haritsa, Ramamritham & Seshadri 1996, Gupta, Haritsa, & Ramamritham 1997), thus increasing the transactions commit time. This paper first analyzes all kind of dependencies that may arise due to data access conflicts in executing-committing transaction when a committing cohort is allowed to lend its data to an executing cohort, and then proposes a static two phase locking based optimistic commit protocol i.e. OCP. In OCP, the execution phase of a cohort is divided into two parts locking phase and processing phase and then, in place of WORKDONE message, WORKSTARTED message is sent just before the start of processing phase of the cohort. Again, in case of dependency, borrower with only commit dependency is allowed to send WORKSTARTED message instead of being blocked. This reduces the time needed for commit processing and is free from cascaded aborts. To ensure non-violation of ACID properties, checking of completion of processing and removal of dependency of cohort are required before sending the Yes-Vote message. The performance of the OCP is also analyzed for partial read-only optimization.

.

Keywords: cohort, commit, workstarted message, workdone message, distributed real time database systems.

1 Introduction

1.1 General

A real time database system operating on distributed data should maintain data consistency as well as it must satisfy timing constraints associated with transaction, typically expressed in term of deadlines. One of the important factors that contribute to the difficulty in meeting transaction deadlines is the data access conflicts among them. The lifetime of a transaction is divided into two stages, viz., execution stage and commitment stage (Lam, Pang, Son & Cao 1999). In the execution stage, the operations of a transaction are processed at different sites of the system, while, in the commit stage, a commit protocol is used to ensure transaction atomicity. Data conflicts can be between two transactions in execution stage (execute-execute conflicts) or between one transaction in execution stage and other in commit stage (execute-commit conflicts). Though, the issue of handling data conflicts between two executing transactions has been addressed up to some extent, very little work has been done on the issue of handling data conflicts between executing-committing transactions. The commit processing in a DRTDBS can significantly increase the execution time of a transaction. This may adversely affect the system's ability to meet transaction deadlines. Therefore, designing a good commit protocol is important for the DRTDBS not only for resilience to failure and speed of recovery but also for normal processing.

1.2 Background and Related Work

The Two Phase Commit (2PC) is still the one most commonly used protocol in the study of DRTDBS (Gray 1991). Most of the existing protocols proposed in the literature are based on it, such as, presumed commit (PC) and presumed abort (PA) (Mohan, Lindsay & Obermarck 1986). PA is optimized for read-only transactions and a class of multisite update transactions, whereas PC is optimized for other classes of multisite update transactions. Soparkar et al. (1992) have proposed a protocol that allows individual sites to unilaterally commit. If it is later found that the decision is not consistent globally then compensation transactions are executed to rectify errors. The problem with this approach is that many actions are irreversible in nature. In (Yoon, Cho & Han 1996), same compensation approach is used.

Gupta et al. proposed optimistic commit protocols in (Gupta, Haritsa, Ramamritham & Seshadri 1996, Gupta, Haritsa, & Ramamritham 1997). These protocols try to improve system concurrency by allowing executing transactions to borrow data from transactions in their commit stage. But these protocols create dependencies among transactions. If a transaction depends on other transactions, it is not allowed to start commit processing and has to be blocked until the transactions, on which it depends, have committed. The blocked committing transaction may include a chain of dependencies as other executing transactions may have data conflicts with it. Enhancement has been made in PROMPT commit

*Lecturer, M. M. M. Engg. College, Gorakhpur - 273010, Uttar Pradesh, India under QIP Scheme.

Copyright © 2006, Australian Computer Society, Inc. This paper appeared at the 17th Australasian Database Conference (ADC 2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 49. Gillian Dobbie and James Bailey, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

protocol proposed in Haritsa et al. (2000), which allows executing transactions to borrow data in a controlled manner only from the healthy transactions in their commit phase. However, it does not consider the type of dependencies between two transactions. The abort of a lending transaction aborts all transaction that has borrowed the data from it. The performance of the system is dependent on chosen threshold value of health factor (HF), which is defined as ratio of TimeLeft to MinTime, where TimeLeft is the time left until the transaction's deadline and MinTime is the minimum time required for commit processing. The impact of buffer space and admission control is also not studied since it is assumed that buffer space is sufficiently large to allow the retention of data updates until commit time. The technique proposed by Lam et al. (1999) maintains three copies of each modified data item (before, after and further) for resolving execute-commit conflicts. This not only creates additional workload on the systems but also has priority inversion problems. Based on the concepts of papers (Lam, Pang, Son, & Cao 1999) and (Haritsa, Ramamritham & Gupta 2000), Biao Qin et al. (2003) proposed a protocol (2SC) which classifies the dependencies between lender and borrower into two types; commit and abort. The abort of lending transaction only forces transactions in its abort dependency set to abort. The transactions in the commit dependency set of aborted lending transaction continue as normal.

There are two locking approaches used by the transactions to obtain a lock on data item i.e. static two phase locking (S2PL) and dynamic two phase locking (D2PL) (Lam, Hung & Son 1997). In conventional database system, D2PL is more favourable than S2PL because the prior locking information of the transactions required in S2PL is not available in most cases. As a result of the better defined nature of real time transactions, it is not uncommon to assume the locking information of transactions to be known before processing. Actually, many real-time concurrency control protocols have made these assumptions or even stronger assumptions. For example, priority ceiling protocols (Rajkumar 1989, Sha, Rajkumar & Lehoczky 1988) explicitly assume the availability of such information. It is acceptable for real time database system because the designer of the real time database systems know in what environment the system operate in order to design system satisfying timing requirement. Actually, real-time S2PL (RT-S2PL) protocols do possess desirable features making them suitable for RTDBS, especially for distributed real-time database systems (DRTDBS), in which remote locking is required and distributed deadlock is possible. At the same time, the problem of prior knowledge on the required data objects of a transaction is easy to address in DRTDBS as it is generally agreed that the behaviour and the data items to be accessed by real-time transactions, especially hard real-time transactions, are much more well-defined and predictable. In addition, the number of messages and the time delay for remote locking can be significantly reduced as they can be packed into a single message for each site. There is no local deadlock. A distributed deadlock is much easier to resolve with S2PL than with D2PL. Deadlock freedom and lower communication overhead of locking by using S2PL makes it attractive for DRTDBS. The use of S2PL for concurrency control in real-time database systems (RTDBS) has received little attention in the past. Actually, there is a complete lack of work on the use of real-time S2PL for RTDBS and DRTDBS (Lam 1994).

The protocols, that allow an executing cohort to borrow data from a committing cohort, do not allow the borrower to send WORKDONE/ PREPARED message and block it until the lender commits. This avoids the problem of cascading aborts but the blocked borrower may be aborted by a higher transaction. They also deal with either blind write model or update model. Our work first analyzes all kinds of dependencies that may arise due to data access conflicts in executing-committing cohort considering an update model, and then proposes a new static locking based real time commit protocol OCP. Here, the execution phase of a cohort is divided into two parts locking phase and processing phase, and in place of WORKDONE message, WORKSTARTED message is sent just before the start of processing phase of the cohort. A cohort may wait due to data contention only during its locking phase. Once, it acquires all the locks, the transaction either completes or is aborted either by a higher priority transaction or due to the expiry of its deadline In case of dependency, borrower is allowed to send WORKSTARTED message, if it is only commit dependent on other cohorts instead of being blocked as opposite to Qin et al. (2003), Haritsa et al. (2000), Gupta, et al. (1996), Gupta et al. (1997). This reduces the time needed for commit processing and is free from cascaded aborts. To ensure non-violation of ACID properties, checking of completion of cohort's processing and the removal of its dependency are required before sending the Yes-Vote. The performance of the OCP is also analyzed for partial read-only optimization which minimizes intersite message traffic, execute-commit conflicts and log writes consequently resulting with a better response time. Simulation of the proposed work is also done which shows that the proposed protocol improves the system performance.

The remainder of the paper is organized as follows. Section 2 introduces a distributed real time database model. Section 3 describes data access conflicts resolving strategies. Section 4 presents OCP with pseudo code of algorithms. Section 5 discusses simulation results, performance measure and evaluation. Section 6 shows the impact of partial read optimization. Section 7 concludes the paper.

2 Distributed Real-Time Database Model

The performance of the OCP is evaluated by developing two simulation models for DRTDBS. The first one is for main memory resident DRTDBS which eliminates the impact of different disk scheduling algorithms on the performances. Since main memory resident databases are not so common in commercial database system, we have also developed another model for disk resident DRTDBS. The structure of our simulation model followed by description of various components such as system model, network model, cohort execution model, database model and model assumptions has been given below.

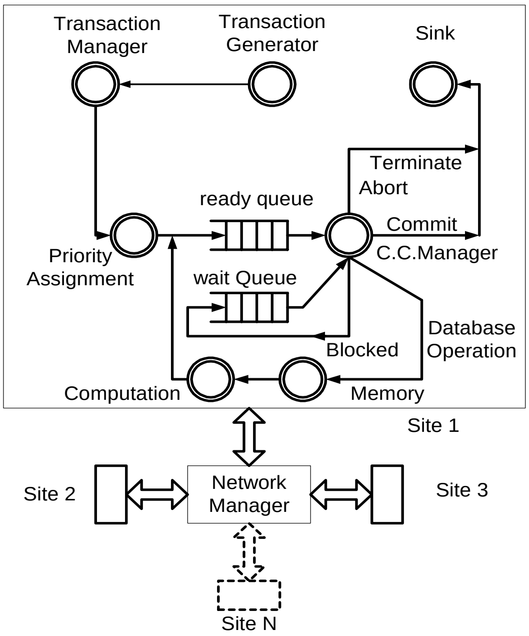


Fig. 1 Distributed Real-time Database System Model

2.1 System Model

In a distributed database system model, the global database is partitioned into a collection of local databases stored at different sites. Each site consists of a transaction generator, a transaction manager, a concurrency controller, a CPU, a ready queue, a local database, a communication interface, a sink and a wait queue. The transaction generator is responsible for the creation of transactions with inter-arrival time and is independent of the generation at other sites. At each site, two types of transactions are generated: global transactions and local transactions. The transaction manager generates cohorts on remote site on behalf of the coordinator. Before a cohort performs any operation on a data object, it has to go through the concurrency control component to obtain a lock on that object. If the request is denied, the cohort is placed into the wait queue. The waiting cohort will be awakened when the requested lock is released and all the locks are available. If the request of all the locks is granted, the cohort will access the memory and perform computation on data items. Finally, cohort may commits/aborts and releases all the locks it has been holding. The sink component of the model is responsible for gathering the statistics for the committed or terminated transactions.

2.2 Database Model

The database is modelled as collection of data items that are uniformly distributed across all the sites. Transactions make requests for the data items and concurrency control is implemented at the data item level. No replication of data items at various sites is considered here.

2.3 Network Model

A communication network interconnects the sites. There is no global shared memory in the system, and all sites communicate via messages exchange over the communication network. Thus, a network manager models the behaviour of the communications network.

2.3 Locking Mechanism

Some of the main techniques used to control concurrent execution of transactions are based on the concept of locking data items. A transaction is said to follow the two phase locking protocol if all locking operations precede first and then unlock operation in the transaction. A variation known as conservative 2PL or static 2PL (S2PL) used here requires a transaction to lock all the data items it accesses before the transaction begins execution, by predeclaring it’s read-set and write-set. If any of the predeclared items can not be locked, the transaction does not lock any items; instead, it wait until all the items are available for locking.

2.4 Cohort Execution Model

There are two types of distributed transaction execution model i.e. sequential and parallel. In sequential execution model, there can be at most one cohort of a transaction at each execution site, and only one cohort can be active at a time. While, in case of parallel execution model, the coordinator of the transaction spawns cohorts all together by sending a message to remote site with a request to activate the cohort, lists all operations to be executed at that site and then cohorts may start execution at the same time in parallel. We have considered cohorts executing in parallel way.

2.5 Model Assumptions

We assume that the transactions are firm real time transactions. Each transaction in this model exists in the form of a coordinator process that executes at the originating site of the transaction and a collection of cohorts executing at remote sites, where the required data items reside. If there is any local data in the access list of the transaction, one cohort is executed locally. Before accessing a data item, the cohort needs to obtain locks on the data item. We also assume that:

- The processing of a transaction requires the use of CPU and data items located at local site or remote site.
- Arrivals of transactions at a site are independent of the arrivals at other sites and use Poisson distribution.
- Each transaction pre-declares its read-set (set of data items that the transaction will only read) and write-set (set of data items that the transaction will write).
- A lending transaction cannot lend the same data item in read/update mode to more than one cohort.
- The cohort already in the dependency set of another cohort cannot permit another incoming cohort to read or update.
- A distributed real time transaction is said to commit, if the coordinator has reached to the commit decision before the expiry of the deadline at its site. This definition applies irrespective of whether cohorts have also received and recorded the commit decision by the deadlines.

- Studies have been made for both main memory resident and disk resident database Communication delay is considered either 0ms or 100ms.
- In case of disk resident database, buffer space is sufficiently large to allow the retention of data updates until commit time.

3 Data Access Conflicts Resolving Strategies

Sharing of data items in conflicting modes creates dependencies among conflicting transactions and constraints their commit order.

Real time transactions used in DRTDBS can be classified on basis of types of operation performed (Ramamritham 1993). It is given below.

- Blind Write Model (Write Only): These types of transactions obtain state of the environment and write into the database.
- Update Model (Read before Write): These types of transactions derive new data and store in the database.
- Read only: These types of transactions read data from database.

3.1 Update Model

Updating data items is often more complex task than querying data items. Updating means editing (or changing) database data, satisfying some condition. When a cohort updates a data item, all constraints must be enforced and controlled, so that action of one cohort does not interfere with those of other and database must be free from inconsistency problem of the concurrent execution of a number of transactions.

3.1.1 Types of Dependencies & their Definitions

We assume that a cohort requests update lock if it wants to update a data item x. The modified definitions of dependency in this paper are given below:

Commit dependency (CDS)
If a transaction T2 updates a data item read by another transaction T1, a commit dependency is created from T2 to T1. Here, T2 is not allowed to commit until T1 is committed.

Abort dependency (ADS)
If T2 reads / updates an uncommitted data item written by T1, an abort dependency is created from T2 to T1. T2 aborts, if T1 aborts and T2 is not allowed to commit before T1.

These dependencies are required to maintain the ACID properties of the transaction. Each transaction Ti, that lends its data while in prepared state to an executing transaction, maintains following set.

CDS (Ti): the set of transactions Tj, that are commit dependent on transaction Ti.

ADS (Ti): the set of transactions Tj, that are abort dependent on transaction Ti

3.1.2 Type of Dependency in Different Cases of Data Conflicts

When data conflicts occur, there are three possible cases of conflict.

Case 1: Read-Update Conflict.

If T2 requests update-lock while T1 is holding a read-lock, a commit dependency is defined from T2 to T1. First, the transaction id of T2 is added to the CDS (T1). Then T2 acquires the update-lock.

Case 2: Update – Update Conflict.

If both locks are update –locks and HF(T1) $\geq$ MinHF, an abort dependency is defined from T2 to T1. The transaction id of T2 is added to ADS (T1), and T2 acquires the update-lock; otherwise, T2 is blocked. {discuss later in section 4)

Case 3: Update -Read Conflict

If T2 requests a read-lock while T1 is holding a update-lock and HF(T1) $\geq$ MinHF, an abort dependency is defined from T2 to T1. The transaction id of T2 is added to ADS (T1), and T2 acquires the read-lock; otherwise, T2 is blocked.

On the basis of the data conflicts discussed in section 3.2.1, the accesses of data item in conflicting mode are processed by lock manager as follows.

```
If (T2 CD T1)
{
    CDS (T1) =CDS (T1) ⋃ {T2};
    T2 is granted Update lock;
}
else  if ((T2 AD T1) AND (HF(T1) ≥ MinHF))
    {
        ADS (T1) =ADS (T1) ⋃ {T2};
        T2 is granted the requested lock (read or
        Update);
    }
    else T2 will be blocked;
```

3.2 Mechanics of interaction between lender and borrower cohorts

When T2 had accessed the already locked data by T1, one among the three possible scenarios described below may arise.

Scenario 1: T1 receives decision before, T2 is going to start processing phase after getting all locks.

If the global decision is to commit, T1 commits.

(1) All cohorts in ADS (T1) and CDS (T1) will execute as usual and the set of ADS (T1) and CDS (T1) deleted.

(2) If the global decision is to abort, T1 aborts. The cohorts in the dependency set of T1 will execute as follows:

- all cohorts in ADS (T1) will be aborted;
- all cohorts in CDS (T1) will execute as usual;
- the set of ADS (T1) and CDS (T1) deleted.

Scenario 2: T2 is going to start processing phase after getting all locks Before, T1 Receives Global Decision.

T2 is allowed to send a WORKSTARTED (discussed in later section) message to its coordinator, if it is commit dependent only; else, it is blocked for sending the WORKSTARTED message (So, the coordinator cannot initiate the commit processing operation). It has to wait, until

1. either T1 receives its global decisions, or
2. its own deadline expires, whichever occurs earlier.

In case 1, the system will execute as the first scenario, whereas in the case 2, T2 will be killed and will be removed from the dependency set of T1.

Scenario 3: T2 aborts before, T1 receives decision

In this situation, T2's updates are undone and T2 will be removed from the dependency set of T1.

4 The OCP Commit Protocol

A critical task in the execution of a transaction in a DRTDBS is to ensure its consistent termination. This is the atomic commitment problem. To address this issue, we have designed a new real time protocol based on the concepts describe below.

Basic Idea

A commit protocol can improved transaction success percentage by

1. Reducing the commit duration for each transaction,
2. Causing locks to be released sooner reducing the wait time of other transactions, and
3. Allowing ordered sharing of locks.

The ideas discussed below are based on the factors listed above.

(A) The execution of a transaction may be delayed due to resource (CPU and disk) or data contentions. The optimization proposed in this paper is based on the following observations:

1. Data contention is the main cause of delay in transaction's execution.
2. A cohort may wait due to data contention only during its locking activity.

We therefore propose to divide the execution phase of the transaction into two parts:

1. Locking Phase, and
2. Processing Phase.

The execution of cohort is carried out according to the following steps:

1. During locking phase, the transaction locks the data items.
2. Just before the start of processing phase, the cohort sends the WORKSTARTED message to its coordinator. Then, it is executed.
3. After the receipt of WORKSTARTED messages from all its cohorts, the coordinator sends PREPARE message at time t, calculated as follows:

$$t = \max \{t_i + \text{processing\_time}_i\} - T_{com}$$

where,

t_i = arrival time of WORKSTARTED message from cohort$_i$

processing_time$_i$ = processing time needed by cohorti

$T_{com=}$ Communication Delay

The important point to note here is that the required optimization is local to each site and do not requires inter-site communications. Moreover the proposed optimization can be integrated with any other protocol based on 2SC.

(B) If a cohort T2 utilizes the dirty data items already locked by other cohorts, one of the following three types of dependencies may arise.

1. It may be commit dependent on other cohorts.
2. It may be abort dependent on other cohorts.
3. It may be commit dependent as well as abort dependent both on other cohorts.

Let us consider the case; T2 is going to start processing phase after getting all locks, before lenders receive global decision. In case of completion of data processing of T2 before receipt of global decision by T1, existing commit protocols including 2SC block borrower from sending the WORKDONE message until lender commits/aborts. We propose to allow a commit dependent borrower to send the WORKSTARTED message as the abort of lender never aborts it. Hence, one of the following two decisions is taken based on types of dependencies, cohort locks all the required lock and going to start processing phase.

1. T2 is allowed to send WORKSTARTED message to its coordinator if it is only commit dependent on other cohorts. This is free from cascaded abort because abort of T1 (lender) causes T2 (borrower) not to abort.
2. T2 is not allowed to send a WORKSTARTED message to its coordinator if it is only abort dependent or combination of commit and abort dependent both on other cohorts. So the coordinator cannot initiate the commit processing. Instead, it has to wait until either T1 receives its global decisions or its own deadline expires, whichever occurs earlier.

(C) Cohort sends the Yes-Vote in response to its coordinator's PREPARE message only when its dependencies are removed and it has finished its processing. If it is still dependent on any cohort or processing is not finished, the Yes-Vote message is deferred. The deferred Yes-Vote message is being sent by the borrower after completion of processing and removals of dependency either due to abort or commit of lender. The important point to note here is that the required modification is local to each site and do not requires inter-site communications.

(D) The CPU scheduling algorithm for the cohorts is described below.

1. When two cohorts are ready to run on the same processor, the higher priority cohort is scheduled first.
2. While in locking period, if a higher priority cohort (T_h) arrives, the lower priority cohort (T_L) aborts. T_L releases all its locked data items.
3. Cohort processing is done on the basis of availability of the CPU and priority. If a higher priority cohort (T_h) arrives during the processing phase of T_L it aborts the lower priority transaction (T_L). The aborted cohort/transaction releases all its locked data items.

Algorithm

On the basis of above discussion, the complete pseudo code of the protocol is given below.

```
if (T1 receives global decision before, T2 is going to start
processing phase after getting all locks)
{ONE: if (T1's global decision is to commit)
      {
          T1 enters in the decision phase;
          All cohorts in ADS (T1) and CDS (T1) will
          execute as usual;
          Delete set of ADS (T1) and CDS (T1);
      }
     else //T1's global decision is to abort
      {   T1 aborts;
          The cohorts in CDS (T1) will execute as usual;
          Transaction in ADS (T1) will be aborted;
           Delete sets of ADS (T1) and CDS (T1);
      }
 }
else if (T2 is going to start processing phase after getting
     all locks before, T1 receives global decision)
     {
         check type of dependencies;
        if (T2's dependency is commit only)
            T2 sends WORKSTARTED message;
        else
        {
             T2 is blocked for sending WORKSTARTED
             message;
            while (! (T1 receive global decision OR T2
             misses deadline))
            {
              if (T2 misses deadline)
               {
                  Undo computation of T2;
                  Abort T2;
                  Delete T2 from CDS (T1) & ADS (T1);
               }
              else  GoTo ONE;
            }
         }
     }
    else //T2 is aborted by higher transaction before,
      // T1 receives decision
    {
        Undo computation of T2;
        Abort T2;
       Delete T2 from CDS (T1) & ADS (T1);
     }
```

5 Simulation Results, Performance Measure and Evaluation

5.1 Performance Parameters and Measures

Since there is no practical benchmark program for DRTDBS in the market or in the research community to evaluate the performances of proposed policy (Lee, Lam & Kao 1999), a distributed real time database system consisting of N sites is simulated. The default values of different parameters used in the simulation experiments are given in Table 1. These settings were chosen to ensure significant levels of both resource contention (RC) and data contention (DC). They were chosen to be in accordance with those used in earlier studies.

The concurrency control scheme used is S2PL with high priority. When a cohort requests a lock on a data item that is held by one or more higher priority cohorts in a conflicting lock mode, the requesting cohort waits for the data item to be released. On the other hand, if the data item is held by only lower priority cohorts in a conflicting way, the lower priority cohorts are aborted and requesting cohort is granted the desired locks. The new reader can join a group of lock-holding readers only if its priority is higher than that of all the writers waiting for the locks.

For simplicity, the cohort's priority assignment policy used is Earliest Deadline First (EDF) in all the experiments. Here, the cohort with closest deadline is assigned highest priority. If any two of the cohorts have same deadline, the one with earliest arrival (FCFS) is assigned a higher priority.

For DRTDBS, one of the most important performance metrics is how well the system can meet the deadlines. Thus, our primary performance measure is the proportion of missed deadlines (or miss ratio, MR) which is defined as the percentage of input transactions that system is unable to complete on or before their deadlines.

MR= number of transactions aborted / number of transactions submitted to system for processing

Miss percentage values in the range of 0 to 20 percent are taken to represent system performance under normal loads, while miss percentage in the range of 20 to 100 percent represents system performance under heavy load (Haritsa, Carey & Livny 1992).

In our simulation model, a small database (200 data items) is used to create a high data contention environment. This helps us in understanding the interaction among the different policies. A small database means that degree of data contention in the system can easily be controlled by the sizes of the transactions. The small database also allows us to study the effect of hot-spots, in which a small part of the database is accessed very frequently by most of the transactions.

The deadline of a transaction is controlled by the runtime estimate of a transaction and the parameter slack factor, which is the mean of an exponential distribution of slack time. The deadlines of transactions (both global and local) are calculated based on their expected execution times. The deadline (D_i) of transaction (T_i) is defined as:

$$D_i=A_i+ SF *R_i$$

where,

A_i is the arrival time of transaction (T_i) at a site.

SF is the slack factor.

R_i is the minimum transaction response time.

As cohorts are executing in parallel, the R_i can be calculated as:

$R_i =R_p+R_c$

where, R_p, the time for execution phase and R_c, the time for commitment phase are given as below.\

For global transaction

$R_p= \max.((2*T_{lock} + T_{process}) \times N_{oper\_local,}\ (2*T_{lock} + T_{process})*N_{oper\_remote})$

$R_{c=}\ N_{comm}* T_{com}$

For local transaction

$R_{p=}\ (2*T_{lock} + T_{process}) \times N_{oper\_local}$

$R_{c=}\ 0$

Where,

T_{lock} is the time required to lock/unlock a data item;

$T_{process}$ is the time to process a data item (assuming read operation takes same amount of time as write operation);

N_{comm} is no. of messages;

T_{com} is communication delay i.e. the constant time estimated for a message going from one site to another;

N_{oper_local} is the number of local operations;

N_{oper_remote} is maximum number of remote operations taken over by all cohorts.

Parameters	Meaning	Default setting
N_{site}	Number of Site	4
AR	Arrival Rate	4 Transactions/ Second
T_{com}	Communication Delay	100 ms (constant)
SF	Slack Factor	1-4 (uniform distribution)
N_{oper}	No. of Operations in a Transaction	3-20 (uniform distribution)
PageCPU	CPU page Processing Time	5 ms
PageDisk	Disk page Processing Time	20 ms
DBsize	Database Size	200 Data Objects/Site
P_{write}	Write Operation Probability	.60

Table I. Default values for the model parameters

5.2 Simulation results

We compare the OCP with PROMPT and 2SC

5.2.1 Main Memory Database

Fig.2 and Fig. 3 show the comparison of OCP with PROMPT and 2SC respectively at communication delay 100ms & 0ms as a function of the average transaction inter-arrival rate/site under normal and heavy load conditions in main memory based database. It can be seen that the proposed protocol works best under all load conditions. The performance improvements are primarily due to not blocking the cohorts to send their WORKSTARTED messages and by not allowing higher priority transactions to abort a borrower. The communication delay is minimized by overlapping the processing time and message transmission time and early sending of WORKSTARTED messages by commit dependent only cohorts, thus reducing the overall time needed for commit processing.

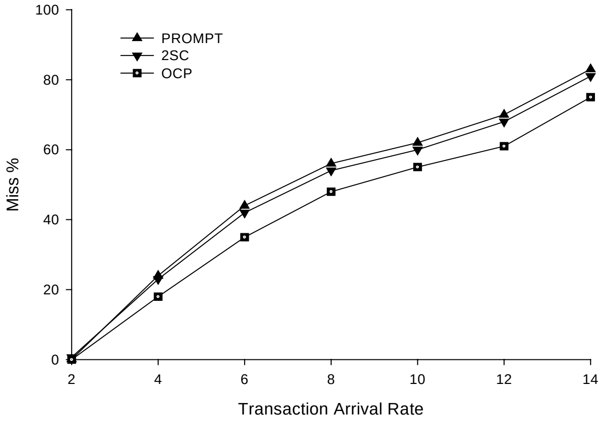


Fig.2 Miss % with (RC+DC) at Communication Delay=100
Normal & heavy Load

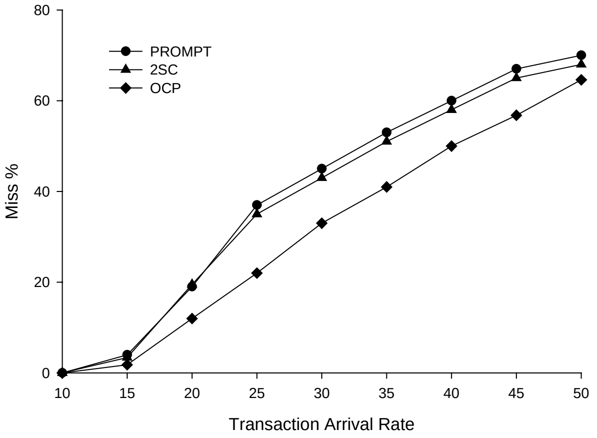


Fig. 3 Miss % with (RC+DC) at Communication Delay=0ms Normal & Heavy Load

5.2.3 Disk Resident Database

Fig.4, Fig. 5 and Fig 6 show the miss ratio at communication delay at 100ms as well as 0ms at different transaction arrival rate in disk resident based database. It can be seen that the proposed protocol works again better than 2SC and PROMPT at communication delay 100ms under all load conditions. The performance improvements are primarily due to not blocking the cohorts to send their WORKSTARTED messages and minimizing queuing delay. The communication delay is also minimized by early sending of WORKSTARTED messages by cohorts only commit dependent.

However, it is not better at communication delay 0ms at higher transaction arrival rate. Rather it is almost at par with 2SC and PROMPT due to higher number of aborts, increased number of dependent cohorts and longer queuing delay for use of resources in the system.

.

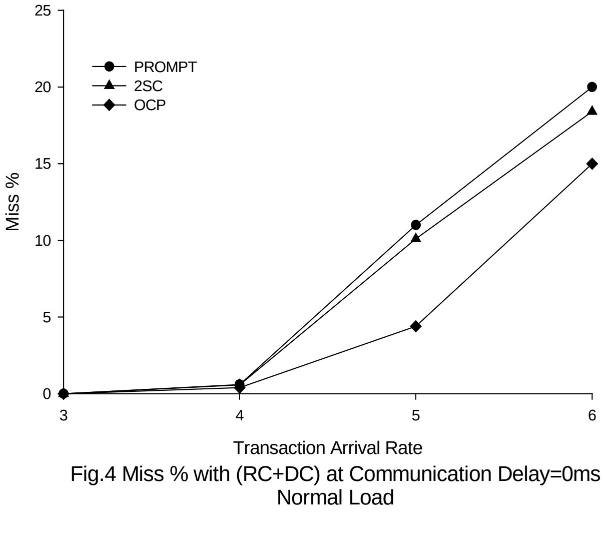


Fig.4 Miss % with (RC+DC) at Communication Delay=0ms Normal Load

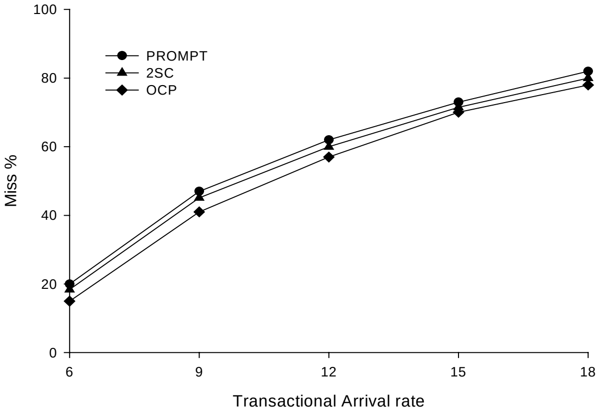


Fig. 5 Miss % with (RC+DC) at Communication Delay=0ms Heavy Load

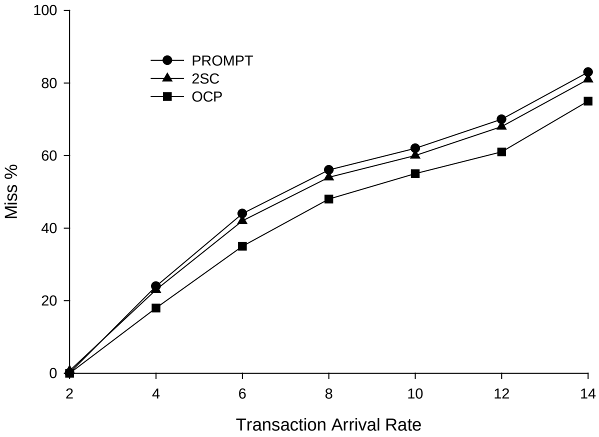


Fig. 6 Miss % with (RC+DC) at Communication Delay=100 Normal & Heavy load

6 Performance of OCP with Partial Read Optimization

This most common optimization is called partial read optimization. It means that a cohort has no work to do at commit and so does not need a PREPARE, commit or abort message from its coordinator. Therefore, a cohort having read only locks will have no locks after sending WORKSTARTED message. This cohort may send a read-only WORKSTARTED message to its coordinator indicating that it is no longer needed by cohort to participate in 2PC. Hence, the write-write and write-read are the only possible conflicts in this case. So, the dependency required in this case is given below:

Abort dependency (ADS)

If T2 reads/writes an uncommitted data item written by T1, an abort dependency is created from T2 to T1. T2 aborts, if T1 aborts and T2 is not allowed to commit before T1.

There are two possible cases of data conflict [1]. Let T1 be the transaction in commit phase and T2 be the transaction in execution phase.

Case 1: Write –Write Conflict

If both locks are write–locks and HF(T1) ≥ MinHF, an abort dependency is defined from T2 to T1. The transaction id of T2 is added to ADS (T1), and T2 acquires the write-lock; otherwise, T2 is blocked.

Case 2: Write-Read Conflict

If T2 requests a read-lock while T1 is holding a write-lock and HF(T1) ≥ MinHF, an abort dependency is defined from T2 to T1. The transaction id of T2 is added to ADS (T1), and T2 acquires the read-lock; otherwise, T2 is blocked.

The effect of partial read only optimization has been also studies in the main memory and disk resident database at communication delay 0 as well as 100 ms. However, the performance gain is better. It is varying in between the range 1%-5% which has been shown in Fig. 7, Fig. 8, Fig. 9, Fig. 10 and Fig. 11 for different types of cases. At low arrival rates it is slightly better but it improve at higher arrival rates.

Main Memory Database

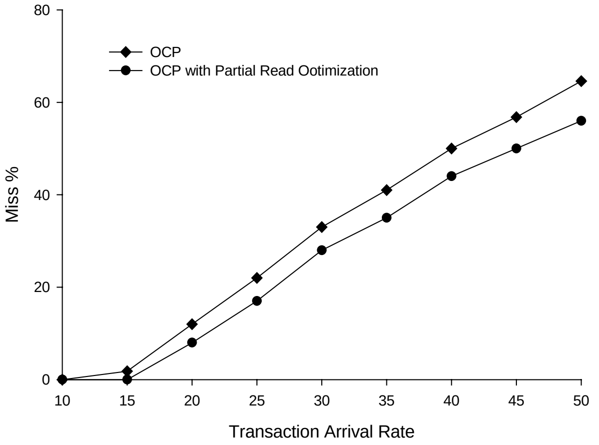


Fig. 7 Miss % with (RC+DC) at Communication Delay=0ms Normal & Heavy Load

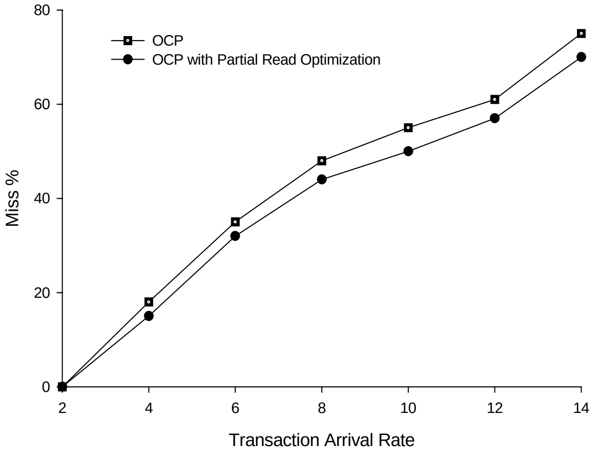


Fig.8 Miss % with (RC+DC) at Communication Delay=100 Normal & heavy Load

Disk Resident Database

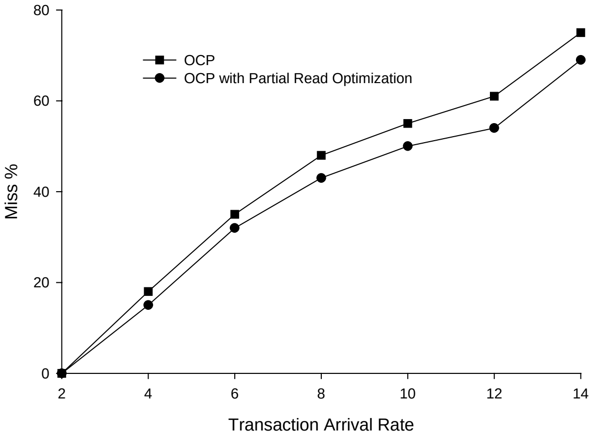


Fig. 9 Miss % with (RC+DC) at Communication Delay=100 Normal & Heavy load

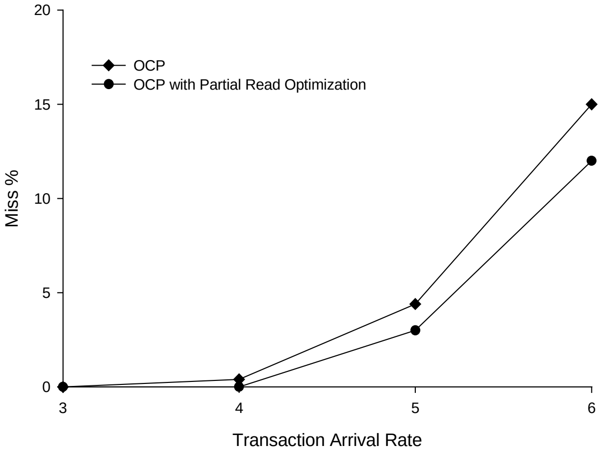


Fig.10 Miss % with (RC+DC) at Communication Delay=0ms Normal Load

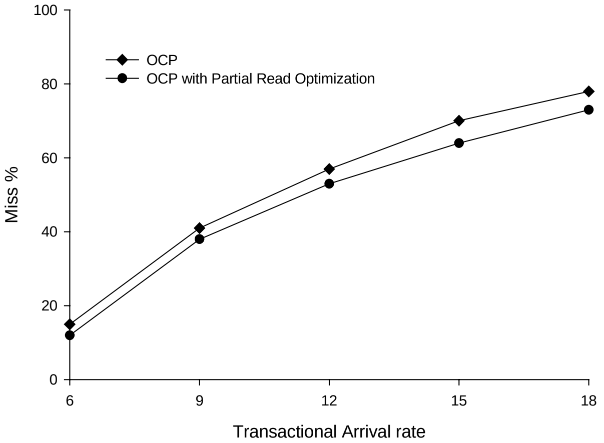


Fig. 11 Miss % with (RC+DC) at Communication Delay=0ms Heavy Load

7 Conclusion

In a large network, communication and queuing delays become a bottleneck. In this paper, we propose to send WORKSTARTED message just before the start of cohort's processing in place of sending WORKDONE message. This overlap the message transmission time with the cohort's processing time and reduces the overall transaction's completion time. It also allows the borrower to send WORKSTARTED message if there is only commit dependencies between borrower and its lenders. It is free from cascaded abort since borrower with only commit dependency is not aborted in case its lenders abort. This reduces the blocking period of the borrower. The simulation results show that the gain in performance can be achieved at low and moderate loads. A suitable modification in distributed real time commit protocol at the time of sending Yes Voting message has been made to ensure atomicity. The important point to note here is that the protocol's feature required is local to each site and do not requires inter-site communications. Moreover, the proposed optimization can be integrated with any other protocol based on 2PC.

8 References

Qin, B. and Liu, Y. (2003): High performance distributed real time commit protocol. *Journal of Systems and Software,* Elsevier Science Inc, 1-8.

Mohan, C., Lindsay, B. and Obermarck, R. (1986): Transaction management in the R* distributed database Management System. *ACM transaction on Database Systems*, Volume **11**(4).

Soparkar, N., Levy, E., Korth, H. F. and Silberschatz, A. (1992): Adaptive Commitment for Real-time Distributed Transaction. *Technical Report TR-92-15*, Department of Computer Science, University of Texax, Austin.

Yoon, Y., Cho, J. and Han, C. (1996): Real-Time Commit Protocol for Distributed Real-Time Database Systems. *Proceedings of Second International Conference on Engineering of Complex Computer Systems*, Canada, IEEE Computer Society Press, Los Alamitos, CA, 221-225.

Lam, K.Y., Pang, C-L., Son, S.H. and Cao, J. (1999): Resolving executing-committing conflicts in distributed real-time database systems. *The computer Journal,* **42** (8): 674-692.

Haritsa, J., Ramamritham, K. and Gupta, R. (2000): The PROMPT real time commit protocol. *IEEE Transaction on parallel and distributed systems,* **11**(2):160-181.

Lam, K. Y. (1994): Concurrency Control in Distributed real time database systems. PhD Thesis, City University of Hong Kong.

Gupta, R., Haritsa, J., Ramamritham K. and Seshadri, S. (1996): Commit processing in distributed real time database systems. *Proceedings of Real-time Systems Symposium,* Washington DC. IEEE Computer Society Press, San Francisco.

Gupta, R., Haritsa, J. and Ramamritham, K. (1997): More optimistic about real time distributed commit processing. *Proceedings of Real-Time Systems Symposium*.

Ramamritham, K. and Chrysanthis, P. (1996): A Taxonomy of correctness criteria in database applications. *VLDB J.*, 5: 85-97.

Ulusoy, O. (1995): A study of two transaction processing architecture for distributed real-time database systems. *J. Syst. Softw*are, 31: 97-108.

Ulusoy, O. and Buchmann, A. (1998): A real time concurrency control protocol for main memory database systems. *Inf. System*, 23:109-125.

Ulusoy, O. (1992): Concurrency control in real time database systems. PhD Thesis, Department of Computer Science, University of Illinois Urbana-Champaign.

Gray, J. and Reuter, A. (1993): *Transaction Processing: Concepts and Technique.* San Mateo, CA, Morgan Kaufman.

Pang, C-L. and Lam, K.Y. (1998): On using similarity for resolving conflicts at commit in mixed distributed real-time databases. *Proceedings of the Fifth International Conference on Real-Time Computing Systems and Applications*.

Gray, J. (1991): Notes on database operating systems. Operating Systems*:* an Advanced Course. 60:397–405.

Lam, K. Y., Hung, S L. and Son, S. H. (1997): On Using Real-Time Static Locking Protocols for Distributed Real-Time Databases. Real-Time Systems, 13: 141–166.

Rajkumar, R. (1989): Task Synchronization in real time systems. Ph.D. Thesis, Carnegie-Mellon University.

Sha, L., Rajkumar, R. and Lehoczky, J. P. (1988): Concurrency Control for distributed real time data bases. ACM SIGMOD Record, **17**(1): 82-98.

Thomasian, A. (1993): Two Phase Locking Performance and It's Thrashing Behavior. ACM Transactions on Database Systems, **18**(4): 579-625.

Silberschatz, A., Korth, H.F. and Sudarshan, S. (2002): *Database Management.* McGraw Hill Higher Education, International Edition.

Lee, Victor C. S., Lam, Kam-yiu and Kao, B. (1999): Priority Scheduling of Transactions in Distributed Real-Time Databases. The International Journal of Time-Critical Computing Systems, 16: 31–62.

Taina J. and Son, S. H. (1999): Towards a General Real-Time Database Simulator Software Library. Proceedings of Active and Real-Time Database Systems.

Agrawal, D., Abbadi, A. El., Jeffers, R. and Lin, L. (1995): Ordered share Locks for real Time Databases. Journal of VLDB, 4: 87-126.

Ramamritham, K (1993): Real-time databases. Distributed and Parallel Databases, Special issue: Research topics in distributed and parallel databases, **1**(2):199-226.

Haritsa, J. R. Carey, M. J. and Livny, M. (1992): Data Access Scheduling in Firm Real-Time database Systems. Journal of Real-Time systems, **4**(3):203-242.

Author Index

Recent Volumes in the CRPIT Series

ISSN 1445-1336

Listed below are some of the latest volumes published in the ACS Series *Conferences in Research and Practice in Information Technology*. The full text of most papers (in either PDF or Postscript format) is available at the series website `http://crpit.com`.

Volume 41 - Theory of Computing 2005
Edited by Mike Atkinson, *University of Otago, New Zealand* and Frank Dehne, *Griffith University, Australia*. January, 2005. 1-920-68223-6.
Contains the papers presented at the Eleventh Computing: The Australasian Theory Symposium (CATS2005), Newcastle, NSW, Australia, January/February 2005.

Volume 42 - Computing Education 2005
Edited by Alison L. Young, *UNITEC, New Zealand* and Denise Tolhurst, *University of New South Wales, Australia*. January, 2005. 1-920-68224-4.
Contains the papers presented at the Seventh Australasian Computing Education Conference (ACE2005), Newcastle, NSW, Australia, January/February 2005.

Volume 43 - Conceptual Modelling 2005
Edited by Sven Hartmann, *Massey University, New Zealand* and Markus Stumptner, *University of South Australia*. January, 2005. 1-920-68225-2.
Contains the papers presented at the Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005), Newcastle, NSW, Australia, January/February 2005.

Volume 44 - ACSW Frontiers 2005
Edited by Rajkumar Buyya, *University of Melbourne*, Paul Coddington, *University of Adelaide*, Paul Montague, *Motorola Australia Software Centre*, Rei Safavi-Naini, *University of Wollongong*, Nicholas Sheppard, *University of Wollongong* and Andrew Wendelborn, *University of Adelaide*. January, 2005. 1-920-68226-0.
Contains the papers presented at the Australasian Workshop on Grid Computing and e-Research (AusGrid 2005) and the Third Australasian Information Security Workshop (AISW 2005), Newcastle, NSW, Australia, January/February 2005.

Volume 45 - Information Visualisation 2005
Edited by Seok-Hee Hong *NICTA, Australia*. January, 2005. 1-920-68227-9.
Contains the papers presented at the Asia-Pacific Symposium on Information Visualisation, APVis.au, Sydney, Australia, January 2005.

Volume 46 - ICT in Education
Edited by Graham Low *University of New South Wales, Australia*. October, 2005. 1-920-68228-7.
Contains selected refereed papers presented at the South East Asia Regional Computer Confederation (SEARCC) 2005: ICT Building Bridges Conference, Sydney, Australia, September 2005.

Volume 47 - Safety Critical Systems and Software 2004
Edited by Tony Cant, *University of Queensland*. March, 2005. 1-920-68229-5.
Contains all papers presented at the Ninth Australian Workshop on Safety-Related Programmable Systems, (SCS2004), Brisbane, Australia, October 2004.

Volume 48 - Computer Science 2006
Edited by Vladimir Estivill-Castro, *Griffith University* and Gillian Dobbie, *University of Auckland, New Zealand*. January, 2006. 1-920-68230-9.
Contains the papers presented at the Twenty-Ninth Australasian Computer Science Conference (ACSC2006), Hobart, Tasmania, Australia, January 2006.

Volume 49 - Database Technologies 2006
Edited by Gillian Dobbie, *University of Auckland, New Zealand* and James Bailey, *University of Melbourne*. January, 2006. 1-920-68231-7.
Contains the papers presented at the Seventeenth Australasian Database Conference (ADC2006), Hobart, Tasmania, Australia, January 2006.

Volume 50 - User Interfaces 2006
Edited by Wayne Piekarski, *University of South Australia*. January, 2006. 1-920-68232-5.
Contains the papers presented at the Seventh Australasian User Interface Conference (AUIC2006), Hobart, Tasmania, Australia, January 2006.

Volume 51 - Theory of Computing 2006
Edited by Barry Jay *UTS, Australia* and Joachim Gudmundsson, *NICTA, Australia*. January, 2006. 1-920-68233-3.
Contains the papers presented at the Twelfth Computing: The Australasian Theory Symposium (CATS2006), Hobart, Tasmania, Australia, January 2006.

Volume 52 - Computing Education 2006
Edited by Denise Tolhurst, *University of New South Wales, Australia* and Samuel Mann, *Otago Polytechnic, Otago, New Zealand*. January, 2006. 1-920-68234-1.
Contains the papers presented at the Eighth Australasian Computing Education Conference (ACE2006), Hobart, Tasmania, Australia, January 2006.

Volume 53 - Conceptual Modelling 2006
Edited by Markus Stumptner, *University of South Australia*, Sven Hartmann, *Massey University, New Zealand* and Yasushi Kiyoki*Keio University, Japan*. January, 2006. 1-920-68235-X.
Contains the papers presented at the Third Asia-Pacific Conference on Conceptual Modelling (APCCM2006), Hobart, Tasmania, Australia, January 2006.

Volume 54 - ACSW Frontiers 2006
Edited by Rajkumar Buyya, *University of Melbourne*, Tianchi Ma, *University of Melbourne*, Rei Safavi-Naini, *University of Wollongong*, Chris Steketee, *University of South Australia* and Willy Susilo, *University of Wollongong*. January, 2006. 1-920-68236-8.
Contains the papers presented at the Fourth Australasian Workshop on Grid Computing and e-Research (AusGrid 2006) and the Fourth Australasian Information Security Workshop (AISW 2006), Hobart, Tasmania, Australia, January 2006.

Volume 55 - Safety Critical Systems and Software 2005
Edited by Tony Cant, *University of Queensland*. Late 2005. 1-920-68237-6.
Contains all papers presented at the 10th Australian Workshop on Safety Related Programmable Systems, August 2005, Sydney, Australia.

Volume 56 - Visual Information Processing 2005
Edited by Hong Yan, *City University of Hong Kong*, Jesse Jin, *University of Newcastle, Australia*, Zhiqiang Liu, *City University of Hong Kong* and Daniel Yeung, *Hong Kong Polytechnic University*. Late 2005. 1-920-68238-4.
Contains papers from the Asia-Pacific Workshop on Visual Information Processing (VIP2005), Hong Kong, December 2005.

Volume 57 - Multimodal User Interaction 2005
Edited by Fang Chen and Julien Epps *National ICT Australia*. December, 2005. 1-920-68239-2.
Contains the proceedings of the Multimodal User Interaction Workshop 2005, NICTA-HCSNet, Sydney, Australia, 13-14 September 2005.

Volume 58 - Advances in Ontologies 2005
Edited by Thomas Meyer, *National ICT Australia, Sydney* and Mehmet Orgun *Macquarie University*. December, 2005. 1-920-68240-6.
Contains the proceedings of the Australasian Ontology Workshop (AOW 2005), Sydney, Australia, 6 December 2005.