Dispensation Order Generation for Pyrosequencing

Mats Carlsson **Nicolas Beldiceanu**

SICS, PO Box 1263, SE-164 29 KISTA, Sweden
Email: matsc@sics.se

Abstract

This article describes a dispensation order generation algorithm for genotyping using the Pyrosequencing method. The input template of the algorithm is a slightly restricted regular expression over the DNA strings that can be expected in a given sample. The algorithm computes a dispensation order that allows for determining, for each polymorphism in the input template, the genotype of any sample.

The algorithm has the structure of a non-deterministic rewrite system, which gives rise to a search tree. We show that within any branch of the search tree, the rewrite system is confluent and terminating. We use nogood generation and limited discrepancy search to prune the search tree and to focus the search for shorter dispensation orders before looking for longer ones.

The algorithm as described herein assumes samples from a diploid genome, but can readily be generalized to general k-ploid genomes.

Keywords: Sequence Analysis, SNP Analysis, Rewrite Systems, Nogood Generation.

1 Introduction

The Pyrosequencing sequencing method was developed (Nyrén, Pettersson & Uhlén 1993, Ronaghi, Karamohamed, Pettersson, Uhlén & Nyrén 1996, Ronaghi, Uhlén & Nyrén 1998) based on the combination of four enzymes: (i) polymerase (bacterial), (ii) sulphurylase (yeast), (iii) luciferase (firefly) and (iv) apyrase (potato). In this way, the incorporation of a specific nucleotide can be quantitatively detected as visible light; see Fig. 1.

Assuming a monoploid input sample, each cycle of the method proceeds as follows[1]: A specific nucleotide is dispensed, i.e. added to the reaction, and is incorporated into a strand of DNA if it matches the current base of the input strand. If it matches a stretch of n bases, it is incorporated n times. Each incorporation event is accompanied by emission of visible light with energy proportional to the amount of incorporated nucleotide, i.e. to n, the number of matched bases. This contributes one peak of height n to a histogram capturing the outcome of the reaction.

Fig. 1 gives the impression that the nucleotides should be dispensed in a cyclic order. While this is appropriate for completely unknown sequences, it is very wasteful in terms of reagents and throughput for single nucleotide polymorphism (SNP) analysis, where all but the polymorphic part(s) of the input sequence is known. Hence, an algorithm that can generate a custom dispensation order for a specific SNP template is of great practical value.

In this article, we will develop such an algorithm for generalized SNP analysis: it handles polymorphisms over sequences of nucleotides rather than single nucleotides, as well as templates containing multiple polymorphisms. In fact, a template can be seen as a somewhat restricted regular expression, which recognizes a language (generates a set of alleles). The general idea of the algorithm is to transform an input template s into a dispensation order o, so that it allows any sample described by s to be unambiguously genotyped[2]. This requirement will be made more precise in Sect. 3.

Example 1
Suppose that we are given the input template[3]

$$\mathtt{C}^1\mathtt{A}^1(\mathtt{A}^1 \mid \mathtt{C}^1)(\epsilon \mid \mathtt{A}^1\mathtt{G}^1\mathtt{A}^1)(\epsilon \mid \mathtt{T}^1\mathtt{G}^1)(\epsilon \mid \mathtt{A}^1 \mid \mathtt{G}^1)\mathtt{T}^1\mathtt{A}^1\mathtt{T}^2\mathtt{C}^1$$

which contains 4 polymorphisms generating 24 alleles. A dispensation order o for this template is CACAGATGATATC. Assuming that we are dealing with a diploid genome, o admits unambiguous genotyping, for:

> It can be shown that for any two pairs of alleles (i, j) and (k, l), either the superimposed histogram for (i, j) is distinct from the superimposed histogram for (k, l), or, for each polymorphism, the multiset of alternatives present in (i, j) equals the multiset of alternatives present in (k, l).

Consider e.g. the allele pairs ($\gamma_1 = \mathtt{CAATATTC}, \gamma_2 = \mathtt{CACTGTATTC}$) and ($\gamma_3 = \mathtt{CACTATTC}, \gamma_4 = \mathtt{CAATGTATTC}$). They are distinct pairs but represent the same genotype considering one polymorphism at a time. So the fact that the pairs yield identical superimposed histograms (see Fig. 2) does not cause any ambiguity. □

How can we arrive at a dispensation order o from a template containing alternatives? We have taken the approach of a rewriting system, the goal of which is to remove all disjunctions while preserving the language being recognized as well as ensuring that the obtained o is separating. The algorithm consists of four components:

- A rewrite system.
- A checker that tests whether o is separating.
- A generator of nogoods to help avoid non-separating dispensation orders.
- A search procedure.

[1] We ignore details of the chemistry and the issue of complementary DNA bases.

[2] In this case, o is called a *separating* dispensation order.

[3] The template notation is a variant of regular expressions and is defined in Sect.2.

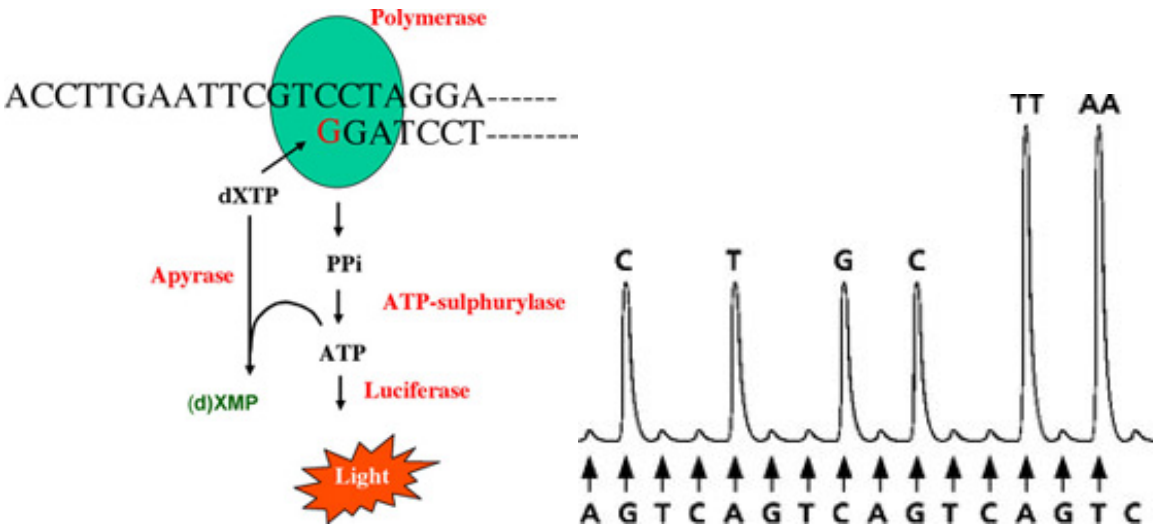

Figure 1: The Pyrosequencing method. A sample is analyzed wrt. a cyclic dispensation order AGTCAGTC... Assuming a monoploid sample, the sequence CTGCTTAA is obtained.

	C	**A**	**C**	**A**	**G**	**A**	**T**	**G**	**A**	**T**	**A**	**T**	**C**
γ_1	1	2	0	0	0	0	1	0	1	2	0	0	1
γ_2	1	1	1	0	0	0	1	1	0	1	1	2	1
γ_3	1	1	1	0	0	0	1	0	1	2	0	0	1
γ_4	1	2	0	0	0	0	1	1	0	1	1	2	1
$\gamma_1 + \gamma_2$	2	3	1	0	0	0	2	1	1	3	1	2	2
$\gamma_3 + \gamma_4$	2	3	1	0	0	0	2	1	1	3	1	2	2

Figure 2: Alleles and histograms

The rest of the article is structured as follows: in Sect. 2, we define some necessary notions and notations; Sect. 3 defines a rewrite system for templates; Sect. 4 describes how to check the separation property; Sect. 5 describes the nogood generator; Sect. 6 gives the search procedure; Sect. 7 discusses related work and future directions. Finally, we conclude.

2 Preliminaries

We will need the following notation: sequence is denoted by juxtaposition; ϵ denotes the empty sequence; alternatives are separated by $|$; $[e_1, e_2, \ldots]$ denotes a list; $\{\{s_1, s_2, \ldots\}\}$ denotes a multiset; parentheses are used for grouping.

For a matrix M, M_i denotes the i^{th} row, and M_j^T denotes the j^{th} column. For two vectors V_1 and V_2, $V_1 \cdot V_2$ denotes their scalar product.

A *DNA string* or simply *string* is a sequence of the four nucleotides $\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}$. For a string γ, $\mathrm{first}(\gamma)$ denotes its first nucleotide.

A *node* n^k denotes a stretch of k nucleotides n, where k is an integer if $k > 0$ is fixed, or $*$ otherwise. A *linear template* is a possibly empty sequence of nodes where no adjacent nodes have the same nucleotide. A *polymorphism* is a number of alternative linear templates. A *constituent* is a node or a polymorphism. A *template* is a sequence of constituents. An *input template* is a template in which all nodes have fixed multiplicity. For a template s, $\mathrm{first}(s)$ ($\mathrm{last}(s)$) denotes its first (last) constituent.

A *dispensation order* is obtained from a linear template by taking the nucleotide sequence, ignoring the multiplicities. A dispensation order o is *separating* if for each sample recognized by the given input template s, the following property holds:

> For each polymorphism p of s, o allows for determining which alternatives of p actually occur in the sample.

Given a dispensation order o and a string γ, $\mathrm{height}(o, i, \gamma)$ denotes the height of the i^{th} peak of the histogram obtained from the Pyrosequencing reaction on γ wrt. o; and $\mathrm{res}(o, i, \gamma)$ denotes the residue of γ, i.e. the possibly empty unincorporated suffix of γ, after completion of the i^{th} cycle of the reaction wrt. o.

3 The Rewrite System

The goal of the rewrite system is to transform the input template s into a linear one. This is done by applying a *rewrite procedure* multiple times, until s is linear. In each iteration, one or more polymorphisms are rewritten away. Each iteration rewrites the template from a form PQR to a form $PQ'R$ where P and R are left unchanged by the procedure, and P and Q' are linear templates. For the dispensation order o obtained from Q', the following properties must hold:

In-phase. For any allele γ generated by Q, the Pyrosequencing reaction wrt. o will fully incorporate γ but will not incorporate any nucleotide of any allele generated by R.

Separation. o is separating.

Optimality. Preferably, o is as short as possible.

3.1 The Rewrite Procedure

The rewrite procedure is concerned with a *variable region* of the template (Q above). A variable region initially consists of a single polymorphism. During the rewrite procedure, the variable region may engulf one node of its left context, and several constituents of its right context.

The rewrite procedure operates on the configuration $\langle P, q, R, \Gamma \rangle$ where q is the polymorphism to be eliminated, P is a linear template, R is a template, and $\Gamma = \{\Gamma_i\}$ is the set of alleles corresponding to the variable region. Each Γ_i is a list of strings and is interpreted as the allele formed by

concatenating its strings. Each Γ_i has the same number k of strings. Each individual string Γ_{ij} arises either from an alternative of some polymorphism, or from a single node. Initially, $k = 1$ and each Γ_{i1} is the string arising from one alternative of q. Several examples are shown in the second column of Fig. 3.

Γ is used by the separation checker. For a fixed j, $\{\{\Gamma_{ij}\}\}$ is the multiset of strings arising from the j^{th} constituent of the variable region.

There are four deterministic rules and one non-deterministic one. The rules are applied until the polymorphism has a single alternative, which is the termination criterion for the procedure. The in-phase requirement can be handled incrementally, whereas the separation requirement can only be checked at the end of the rewrite. We will now describe each rule. The rules are summarized in Fig. 3.

3.2 Absorption Rule

Applies if we are in one of the following cases, where x is a node with the same nucleotide as $\mathrm{last}(P)$:

- x is the first node of some alternative of q.
- x is $\mathrm{first}(R)$ and q contains the empty sequence.

Then $\mathrm{last}(P)$ is distributed into each alternative. Also, the string arising from $\mathrm{last}(P)$ is added to the front of each element of Γ. This rule is language-preserving.

The motivation of this rule is to ensure the in-phase property, and in particular the basic fact that each cycle in the Pyrosequencing reaction incorporates a maximal stretch of a given nucleotide.

3.3 Distribution Rule

Applies if the Absorption Rule doesn't apply and we are in one of the following cases, where x is a node with the same nucleotide as $\mathrm{first}(R)$:

- x is the last node of some alternative of q.
- One of the alternatives of q contains x, and another one is the empty sequence.

Then $\mathrm{first}(R)$ is distributed into each alternative. Also, the string arising from $\mathrm{first}(R)$ is added to the back of each element of Γ. This rule is language-preserving.

The motivation of this rule is to ensure the in-phase property.

3.4 Product Rule

Applies if none of the above rules applies and $\mathrm{first}(R)$ is a polymorphism. Then q and $\mathrm{first}(R)$ are replaced by a single polymorphism over all combinations of their alternatives. A new Γ is obtained by adding the string arising from every alternative of the second polymorphism to every element of Γ. This rule is language-preserving.

The motivation of this rule is to ensure the in-phase property. A careful analysis could perhaps reveal cases where in-phase could be preserved without applying this rule.

3.5 Factoring Rule

Applies if none of the above rules applies and all alternatives of the polymorphism q begin with nodes with a common nucleotide, which is factored out into a node placed in front of q. The factored node may have variable multiplicity, i.e. the rewritten template may recognize a superset of the language recognized by the original template. Γ is unchanged. Also, duplicate alternatives are removed from the polymorphism.

This rule contributes to the transformation into a linear template.

3.6 Ordering Rule

Applies if none of the other rules applies. One of the following rewrites is done:

1. If some alternatives begin with nodes with a common nucleotide, then it is factored out into a node placed in front of q. This rewrite is tried for each nucleotide that occurs in at least one leading node. The nucleotide that starts the longest (in terms of nodes) alternative is tried first. Γ is unchanged.
2. If the j^{th} alternative is the empty sequence, then:
   (a) $\mathrm{first}(R)$ is distributed into each alternative.
   (b) The single node of the j^{th} alternative is factored out and placed in front of q.
   (c) The string arising from $\mathrm{first}(R)$ is added to the back of each element of Γ.

   This rewrite is tried last.

The factored node will have variable multiplicity, i.e. the rewritten template will recognize a superset of the language recognized by the original template. Also, duplicate alternatives are removed from the polymorphism.

This rule contributes to the transformation into a linear template. It also represents a choice-point in the rewrite procedure, thus generating a search for a valid dispensation order. The purpose of the heuristic is to favor shorter dispensation orders over longer ones.

Fig. 3 shows an example where this rule produces three different rewrites from one template.

3.7 Confluence and Termination

The fact that the Ordering Rule is non-deterministic means that the rewrite procedure generates a search tree with choice-points corresponding to alternative rewrites by the Ordering Rule. Each leaf of the search tree corresponds to a linear template, which can be seen as a regular expression recognizing any string that is recognized by the input template s. Nevertheless, it is meaningful to consider confluence and termination within a given branch of the search tree, from the root to a leaf. Within a given branch, each application of the Ordering Rule is committed to one particular choice. We summarize these properties in Prop. 1.

Proposition 1 *For a given branch of the search tree, the rewrite system is confluent and terminating.*

Proof.

Confluence. Trivial, since all rules are mutually exclusive, and q is a specific polymorphism.

Termination. The goal of the rewrite system is to rewrite the configuration until only one alternative of q remains. We will show that this is always done in a finite number of steps. Consider first the effect of the rules on the template:

- Absorption can be applied as the very first step in a branch, and may also be applicable after a Product of two polymorphisms that both contain the empty sequence as an alternative. So the number of applications of Absorption is bounded above by the number of polymorphisms.
- Distribution, Product and Ordering (case 2) incorporate $\mathrm{first}(R)$ into q.
- Factoring and Ordering (case 1) decrease the total size of q and emit one node into P.

Rule	Template Γ	Rewritten Template Rewritten Γ
Absorption	$\mathtt{C}^1(\mathtt{C}^1 \mid \mathtt{G}^1)\mathtt{T}^1$ $\{[\mathtt{C}], [\mathtt{G}]\}$	$(\mathtt{C}^2 \mid \mathtt{C}^1\mathtt{G}^1)\mathtt{T}^1$ $\{[\mathtt{C},\mathtt{C}], [\mathtt{C},\mathtt{G}]\}$
Absorption	$\mathtt{C}^1(\epsilon \mid \mathtt{G}^1)\mathtt{C}^1$ $\{[\epsilon], [\mathtt{G}]\}$	$(\mathtt{C}^1 \mid \mathtt{C}^1\mathtt{G}^1)\mathtt{C}^1$ $\{[\mathtt{C},\epsilon], [\mathtt{C},\mathtt{G}]\}$
Distribution	$\mathtt{T}^1(\mathtt{C}^1 \mid \mathtt{G}^1)\mathtt{G}^1\mathtt{A}^1$ $\{[\mathtt{C}], [\mathtt{G}]\}$	$\mathtt{T}^1(\mathtt{C}^1\mathtt{G}^1 \mid \mathtt{G}^2)\mathtt{A}^1$ $\{[\mathtt{C},\mathtt{G}], [\mathtt{G},\mathtt{G}]\}$
Distribution	$\mathtt{T}^1(\epsilon \mid \mathtt{C}^1\mathtt{G}^1\mathtt{C}^1)\mathtt{G}^1\mathtt{A}^1$ $\{[\epsilon], [\mathtt{CGC}]\}$	$\mathtt{T}^1(\mathtt{G}^1 \mid \mathtt{C}^1\mathtt{G}^1\mathtt{C}^1\mathtt{G}^1)\mathtt{A}^1$ $\{[\epsilon,\mathtt{G}], [\mathtt{CGC},\mathtt{G}]\}$
Product	$\mathtt{T}^1(\mathtt{C}^1 \mid \mathtt{G}^1)(\epsilon \mid \mathtt{G}^1\mathtt{A}^1)\mathtt{T}^1$ $\{[\mathtt{C}], [\mathtt{G}]\}$	$\mathtt{T}^1(\mathtt{C}^1 \mid \mathtt{C}^1\mathtt{G}^1\mathtt{A}^1 \mid \mathtt{G}^1 \mid \mathtt{G}^2\mathtt{A}^1)\mathtt{T}^1$ $\{[\mathtt{C},\epsilon], [\mathtt{C},\mathtt{GA}], [\mathtt{G},\epsilon], [\mathtt{G},\mathtt{GA}]\}$
Factoring	$\mathtt{A}^1(\mathtt{C}^1\mathtt{A}^1 \mid \mathtt{C}^2\mathtt{G}^1 \mid \mathtt{C}^1\mathtt{G}^1)\mathtt{C}^1$ $\{[\mathtt{CA}], [\mathtt{CCG}], [\mathtt{CG}]\}$	$\mathtt{A}^1\mathtt{C}^*(\mathtt{A}^1 \mid \mathtt{G}^1)\mathtt{C}^1$ $\{[\mathtt{CA}], [\mathtt{CCG}], [\mathtt{CG}]\}$
Ordering	$\mathtt{A}^1(\epsilon \mid \mathtt{C}^1 \mid \mathtt{C}^2\mathtt{A}^1\mathtt{C}^1 \mid \mathtt{G}^1\mathtt{C}^1)\mathtt{T}^1$ $\{[\epsilon], [\mathtt{C}], [\mathtt{CCAC}], [\mathtt{GC}]\}$	$\mathtt{A}^1\mathtt{C}^*(\epsilon \mid \mathtt{A}^1\mathtt{C}^1 \mid \mathtt{G}^1\mathtt{C}^1)\mathtt{T}^1$ $\{[\epsilon], [\mathtt{C}], [\mathtt{CCAC}], [\mathtt{GC}]\}$
Ordering	$\mathtt{A}^1(\epsilon \mid \mathtt{C}^1 \mid \mathtt{C}^2\mathtt{A}^1\mathtt{C}^1 \mid \mathtt{G}^1\mathtt{C}^1)\mathtt{T}^1$ $\{[\epsilon], [\mathtt{C}], [\mathtt{CCAC}], [\mathtt{GC}]\}$	$\mathtt{A}^1\mathtt{G}^*(\epsilon \mid \mathtt{C}^1 \mid \mathtt{C}^2\mathtt{A}^1\mathtt{C}^1)\mathtt{T}^1$ $\{[\epsilon], [\mathtt{C}], [\mathtt{CCAC}], [\mathtt{GC}]\}$
Ordering	$\mathtt{A}^1(\epsilon \mid \mathtt{C}^1 \mid \mathtt{C}^2\mathtt{A}^1\mathtt{C}^1 \mid \mathtt{G}^1\mathtt{C}^1)\mathtt{T}^1$ $\{[\epsilon], [\mathtt{C}], [\mathtt{CCAC}], [\mathtt{GC}]\}$	$\mathtt{A}^1\mathtt{T}^*(\epsilon \mid \mathtt{C}^1\mathtt{T}^1 \mid \mathtt{C}^2\mathtt{A}^1\mathtt{C}^1\mathtt{T}^1 \mid \mathtt{G}^1\mathtt{C}^1\mathtt{T}^1)$ $\{[\epsilon,\mathtt{T}], [\mathtt{C},\mathtt{T}], [\mathtt{CCAC},\mathtt{T}], [\mathtt{GC},\mathtt{T}]\}$

Figure 3: Rewrite Rules

Since R is finite and never grows, and the number of applications of Absorption is bounded, q can only be expanded a finite number of times. After its last expansion, it takes at most $|q|$ applications of Factoring and Ordering (case 1) to reach termination, where $|q|$ is the total number of nodes of q.

□

4 Separation Checker

The input of the checker is a candidate dispensation order o and a set Γ of alleles, for one rewritten variable region. In this section, we will describe how to test whether o is separating. The description will assume samples from a diploid genome, but the method can be readily generalized to general k-ploid genomes; see Sect. 7.

First, assume that $|\Gamma| = n$ and $|o| = w$. We simulate the Pyrosequencing reaction on each element of Γ wrt. o, obtaining an $n \times w$ matrix M, where $M_{ij} = \text{height}(o, j, \Gamma_i)$.

Then, we check if the equation $M_{l_1} + M_{l_2} = M_{r_1} + M_{r_2}$ has any non-trivial solution for $l_1, l_2, r_1, r_2 \in [1, w]$. This is implemented by introducing a $4 \times n$ matrix B of 0/1-variables, posing the constraint system[4]:

$$
\begin{array}{rcl}
l_1 & \leq & l_2 \\
l_1 & < & r_1 \\
r_1 & \leq & r_2 \\
l_2 & \neq & r_1 \\
l_2 & \neq & r_2 \\
B_{1h} & = & \begin{cases} 1, \text{if } h = l_1 \\ 0, \text{otherwise} \end{cases} \\
B_{2h} & = & \begin{cases} 1, \text{if } h = l_2 \\ 0, \text{otherwise} \end{cases} \\
B_{3h} & = & \begin{cases} 1, \text{if } h = r_1 \\ 0, \text{otherwise} \end{cases} \\
B_{4h} & = & \begin{cases} 1, \text{if } h = r_2 \\ 0, \text{otherwise} \end{cases} \\
B_1 \cdot M_1^T + B_2 \cdot M_1^T & = & B_3 \cdot M_1^T + B_4 \cdot M_1^T \\
 & \vdots & \\
B_1 \cdot M_w^T + B_2 \cdot M_w^T & = & B_3 \cdot M_w^T + B_4 \cdot M_w^T
\end{array}
\qquad (1)
$$

[4]The first five constraints are for breaking symmetries.

If a solution is found, it means that the allele combinations (l_1, l_2) and (r_1, r_2) will yield identical superimposed histograms for o. This violates the separation property, if these combinations represent distinct genotypes. That is the case iff there exists a p such that:

$$\{\{\Gamma_{l_1 p}, \Gamma_{l_2 p}\}\} \neq \{\{\Gamma_{r_1 p}, \Gamma_{r_2 p}\}\} \qquad (2)$$

If such a p exists, we say that (l_1, l_2, r_1, r_2, p) is a *witness* to the fact that o is non-separating, and the rewrite procedure backtracks to the most recent application of Ordering Rule, trying an alternative rewrite.

5 Nogood Generation

Having encountered a dead end, i.e. a dispensation order o that is non-separating wrt. a given witness, we would like to avoid finding in the future other dispensation orders that are also non-separating wrt. the same witness. To this end, we capture the dead end just encountered in a *nogood*, and in the Factoring and Ordering rules, we check that the partial dispensation order produced so far does not match any nogood. Nogood recording (Schiex & Verfaillie 1994) is a known technique for pruning the search tree in backtrack search. It is well known that nogoods should be as general as possible in order to be maximally effective.

Our first nogood generation scheme is based on the fact that certain prefixes o' of o may also be non-separating. We capture this idea in the following proposition:

Proposition 2 *Let o be a dispensation order satisfying (1) and (2) for alleles l_1, l_2, r_1, r_2 and polymorphism p. Then for any prefix o' of o satisfying the following conditions, any dispensation order beginning with o' also satisfies (1) and (2) for l_1, l_2, r_1, r_2 and p.*

- $R_{l_1} = \text{res}(o', |o'|, \Gamma_{l_1})$ *begins after polymorphism p*
- $R_{l_2} = \text{res}(o', |o'|, \Gamma_{l_2})$ *begins after polymorphism p*
- $R_{r_1} = \text{res}(o', |o'|, \Gamma_{r_1})$ *begins after polymorphism p*
- $R_{r_2} = \text{res}(o', |o'|, \Gamma_{r_2})$ *begins after polymorphism p*
- $\{\{R_{l_1}, R_{l_2}\}\} = \{\{R_{r_1}, R_{r_2}\}\}$

Proof.

1. From (1), we have that the incorporated parts of the alleles yield identical superimposed histograms.

2. The multiset formed by the unincorporated parts of l_1 and l_2 equals the multiset formed by the unincorporated parts of r_1 and r_2, and so any continuation of o' will yield identical superimposed histograms for the given witness.

□

Thus, as a first nogood generation rule, we could search for the shortest prefix o' for which the conditions of Prop. 2 hold. However, we can do better than that. Given such an o', we observe that a dispensation order o'' will also be non-separating provided that o'' yields the same sequence of peaks for the witness alleles, possibly interspersed with zero-height peaks. Namely, if $o'' = \omega_1 o'_1 \cdots \omega_k o'_k$ where ω_i is a sequence of nucleotides such that each ω_i will generate zero-height peaks for the witness alleles. How can we compute ω_i? Let N_i denote consider the set of exposed nucleotides after the i^{th} dispensation of o:

$$\begin{aligned} N_i = \quad & \{\text{first}(\text{res}(o, i, \Gamma_{l_1}))\} \cup \\ & \{\text{first}(\text{res}(o, i, \Gamma_{l_2}))\} \cup \\ & \{\text{first}(\text{res}(o, i, \Gamma_{r_1}))\} \cup \\ & \{\text{first}(\text{res}(o, i, \Gamma_{r_2}))\} \end{aligned}$$

We have that the nucleotides that may occur in ω_i are exactly those that don't occur in N_{i-1}.

We are now ready to present an improved nogood generation rule in the following proposition:

Proposition 3 *Let o be a dispensation order satisfying (1) and (2) for alleles l_1, l_2, r_1, r_2 and polymorphism p. Let o' be a prefix of o satisfying the conditions of Prop. 2. Then any dispensation order o'' generated by the regular expression $(\omega_1{}^* o'_1 \cdots \omega_k{}^* o'_k \ldots)$ where $\omega_i = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\} \setminus N_{i-1}$ also satisfies (1) and (2) for l_1, l_2, r_1, r_2 and p.*

Proof. By construction, the ω_i will generate zero-height peaks for the witness alleles. □

Thus, each nogood is computed as a regular expression, or for efficiency, as the corresponding finite automaton. In each application of Ordering Rule, the rule checks that the dispensation order o obtained from the rewritten template is not recognized by any nogood.

Example 2
Consider:

- Original template: $\mathtt{C}^2(\mathtt{A}^1 \mid \mathtt{C}^1)(\mathtt{C}^1 \mid \mathtt{T}^1 \mid \mathtt{A}^1)\mathtt{T}^1$.
- Tentative rewritten linear template: $\mathtt{C}^*\mathtt{A}^*\mathtt{C}^*\mathtt{T}^*$.
- Tentative dispensation order o: CACT.
- Alleles and corresponding histograms:

i	Γ_i	**C**	**A**	**C**	**T**
1	[CC, A, C, T]	2	1	1	1
2	[CC, C, C, T]	4	0	0	1
3	[CC, A, T, T]	2	1	0	2
4	[CC, C, T, T]	3	0	0	2
5	[CC, A, A, T]	2	2	0	1
6	[CC, C, A, T]	3	1	0	1

- o is not separating, for allele combinations $(2, 3)$ and $(4, 6)$ yield identical superimposed histograms, and for the polymorphism $(\mathtt{A}^1 \mid \mathtt{C}^1)$ we have that:

$$\{\{\mathtt{A}, \mathtt{C}\}\} = \{\{\Gamma_{22}, \Gamma_{32}\}\} \neq \{\{\Gamma_{42}, \Gamma_{62}\}\} = \{\{\mathtt{C}, \mathtt{C}\}\}$$

- CA is a prefix of o satisfying the conditions of Prop. 2, witness:

i	**C**	**A**	$\text{res}(o, 0, \Gamma_i)$	$\text{res}(o, 1, \Gamma_i)$	$\text{res}(o, 2, \Gamma_i)$
2	4	0	CCCCT	T	T
3	2	1	CCATT	ATT	TT
4	3	0	CCCTT	TT	TT
6	3	1	CCCAT	AT	T

- $N_0 = \{\mathtt{C}\}, N_1 = \{\mathtt{A}, \mathtt{T}\}$.
- Regular expression constructed according to Prop. 3:

$$(\mathtt{A} \mid \mathtt{G} \mid \mathtt{T})^*\mathtt{C}(\mathtt{C} \mid \mathtt{G})^*\mathtt{A}\ldots$$

□

The following example compares the two nogood generation schemes on one example taken from a real-life test suite.

Example 3
Given template:

$$\begin{aligned} & (\mathtt{A}^1 \mid \mathtt{C}^1 \mid \mathtt{T}^1)(\mathtt{A}^1 \mid \mathtt{C}^1 \mid \mathtt{T}^1 \mid \mathtt{G}^1)(\mathtt{A}^1 \mid \mathtt{C}^1 \mid \mathtt{T}^1) \\ & \mathtt{C}^1\mathtt{T}^1\mathtt{G}^1\mathtt{C}^1\mathtt{A}^1\mathtt{T}^1(\epsilon \mid \mathtt{C}^1)\mathtt{G}^1\mathtt{A}^1\mathtt{C}^1\mathtt{T}^1 \end{aligned}$$

No separating dispensation order exists for this template. The following table shows for the two schemes: the number of nogoods generated, the number of times that a partial dispensation order was found to match some nogood resp. not match any nogood, and the total execution time:

	By Prop. 2	**By Prop. 3**
# nogoods generated	1095	2
# some nogood matched	10189	12430
# no nogood matched	26565	10644
execution time (msec)	242000	4500

□

6 Search Procedure

We could let the search procedure simply perform a backtrack search over all possible rewrites, stopping as soon as a separating dispensation order is found. However, we would like to find a dispensation order that is as short as possible. Recall that the Ordering Rule tries the rewrite that tends to give the shortest dispensation order first. So we would like to minimize the number of times that the Ordering Rule is allowed to try a non-first rewrite on any branch. This idea is tantamount to Limited Discrepancy Search (Harvey & Ginsberg 1995). In Alg. 1, we show how this can be done.

The rewritechk procedure performs a search with a given discrepancy limit d, bounding the number of times that the Ordering Rule is allowed to try a non-first rewrite. The application of that rule on line 8 will create a choicepoint. The **backtrack** statement on line 17 will make the procedure backtrack to the most recently created choicepoint, causing the rule to try another rewrite.

The rewrite procedure calls rewritechk with stepwise increased d, until either a solution is found, or the discrepancy limit is not reached. Throughout the search, nogoods are accumulated and help prune the search tree.

7 Future and Related Work

In order to generalize the handling of the separation property from diploid to k-ploid genomes, the following straightforward changes need to be made:

1. Witnesses must consist of $2k$ alleles and a polymorphism.
2. In the constraint system (1), we need variables $l_1, \ldots, l_k$ and $r_1, \ldots, r_k$ and a $2k \times n$ matrix B.
3. The constraints (1) and (2), the definition of N_i, and Propositions 2 and 3 must be generalized to taking $l_1, \ldots, l_k$ and $r_1, \ldots, r_k$ into account.

```
PROCEDURE rewrite(s, s')
 1: erase all nogoods and candidate orders
 2: d ← 0
 3: while ¬rewritechk(s, s', d) do
 4:    if the discrepancy limit d was reached then
 5:       d ← d + 1
 6:    else
 7:       warning: no separating dispensation order
 8:       s' ← the first saved candidate order
 9:       return
PROCEDURE rewritechk(s, s', d) : (false, true)
 1: loop
 2:    rewrite s into s'' using all rules but Ordering Rule
 3:    if s'' contains a polymorphism then
 4:       if d ≤ 0 then
 5:          apply Ordering Rule to s'' giving s,
 6:          trying the first rewrite only
 7:       else
 8:          apply Ordering Rule to s'' giving s,
 9:          trying any rewrite
10:       d ← d − 1
11:    else if s'' is separating then
12:       s' ← s''
13:       return true
14:    else
15:       save s'' as a candidate order
16:       compute and assert a nogood
17:       backtrack
```

Algorithm 1: Rewriting a variable region

As an extension of the work reported herein, we have used a constraint programming approach (Hentenryck 1989) to solve the problem of multiplexed analysis, i.e. generating a single dispensation order for multiple input templates describing a mixture of samples. This is done by merging dispensation orders computed by our algorithm in such a way that the ability to genotype each sample is retained.

Our algorithm is the successor of another algorithm, which handled SNPs and indels, but not general input templates. The previous algorithm[5] can roughly be described as a greedy algorithm. It did not decompose the problem into (i) computing individual dispensation orders and (ii) multiplexing them. Our algorithm has been encapsulated into a Pyrosequencing software module.

8 Conclusion

We have described a dispensation order generation algorithm for genotyping using the Pyrosequencing method. The input template of the algorithm is a slightly restricted regular expressions over the DNA strings that can occur in a given sample. The algorithm computes a dispensation order that allows for determining, for each polymorphism in the input template, the genotype of any sample.

We conjecture without proof that if a valid dispensation order exists, the algorithm will find it.

The algorithm has the structure of a non-deterministic rewrite system, which gives rise to a search tree. We have shown that within any branch of the search tree, the rewrite system is confluent and terminating. We prune the search tree by using a strong nogood generator, which generalizes non-separating dispensation orders into regular expressions. We focus the search for short dispensation orders by using limited discrepancy search. The algorithm was implemented in SICStus Prolog (Carlsson et al. 1995), making heavy use of Prolog's unique capabilities for backtrack search, dynamic predicates, and symbolic computation.

We described the algorithm assuming samples from a diploid genome, and then showed how to generalize it to general k-ploid genomes. We outlined how to extend it to multiplexed analysis and briefly compared it with an existing algorithm.

[5] A proprietary, unpublished algorithm.

Acknowledgements

The research reported herein was funded by Pyrosequencing AB[6]. The graphics used in Fig. 1 is courtesy of the Division of Molecular Biotechnology, Royal Institute of Technology, Stockholm[7].

References

Carlsson, M. et al. (1995), *SICStus Prolog User's Manual*, release 3 edn, Swedish Institute of Computer Science. ISBN 91-630-3648-7.

Harvey, W. D. & Ginsberg, M. L. (1995), Limited discrepancy search, *in* C. S. Mellish, ed., 'Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI-95)', Morgan Kaufmann, pp. 607–615.

Hentenryck, P. V. (1989), *Constraint Satisfaction in Logic Programming*, The MIT Press.

Nyrén, Pettersson & Uhlén (1993), 'Solid phase DNA minisequencing by an enzymatic luminometric inorganic pyrophosphate detection assay', *Anal. Biochem.* **208**, 171–175.

Ronaghi, Karamohamed, Pettersson, Uhlén & Nyrén (1996), 'Real-time DNA sequencing using detection of pyrophosphate release', *Anal. Biochem.* **242**, 84–89.

Ronaghi, Uhlén & Nyrén (1998), 'A sequencing method based on real-time pyrophosphate', *Science* **281**, 363–365.

Schiex, T. & Verfaillie, G. (1994), 'Nogood Recording for Static and Dynamic Constraint Satisfaction Problems', *International Journal of Artificial Intelligence Tools* **3**(2), 187–207.

[6] www.pyrosequencing.com

[7] www.biotech.kth.se/molbio