Computational Efficiency of Intermolecular Gene Assembly

Tseren-Onolt Ishdorj[1], Remco Loos[2], and Ion Petre[1]

[1]Computational Biomodelling Laboratory
Department of Information Technologies
Åbo Akademi University, Turku 20520 Finland
E-mail:{tishdorj,ipetre}@abo.fi
[2]Research Group on Mathematical Linguistics
Rovira i Virgili University
Plaça Imperial Tàrraco 1, 43005 Tarragona, Spain.
E-mail:remcogerard.loos@urv.cat

Abstract

In this paper, we investigate the computational efficiency of gene rearrangement operations found in ciliates, a type of unicellular organisms. We show how the so-called guided recombination systems, which model this gene rearrangement, can be used as problem solvers. Specifically, we prove that these systems can uniformly solve SAT in time $O(n \cdot m)$ for a boolean formula of m clauses over n variables.

1 Introduction

Ciliates are unicellular organisms [1] which have attracted attention from computer scientists because of the complex nature of the gene rearrangement of some species. Specifically, the DNA in their micronucleus, used for conjugation only, is transformed into shorter molecules used for transcription. This process is called *gene assembly* and is in some sense reminiscent of the use of "linked lists" in software engineering, see [1].

Two main computational models have been proposed to model gene assembly in ciliates. The so-called *intermolecular* model, introduced by Landweber and Kari [10, 8], allows for operations involving two molecules. The *intramolecular* model, proposed by Ehrenfeucht, Prescott and Rozenberg [2, 14], contains only operations acting on a single molecule. The computational power of the intermolecular model has been well studied, specifically in [8] it is shown that in some formulation the model is as powerful as a Turing machine. It was recently proved that also the intramolecular model is computationally universal [7].

Continuing the investigation of gene assembly from the perspective of computability theory, it was recently proved that the intramolecular model is computationally efficient: SAT may be solved in this model in linear time, see [6]. We refer to

[11] for a related result on splicing systems. In this paper we address the same question for the intermolecular model. We show how the guided recombination model of [8] can be regarded as a problem solving device. The model we consider involves the maximal parallel application of contextual recombination rules, as defined in this paper. We present an algorithm to show that in this model, SAT can be solved in time $O(n \cdot m)$ by a guided recombination system, with n denoting the number of variables and m the number of clauses.

2 Preliminaries and Notation

We assume the reader to be familiar with the basic elements of formal languages, Turing computability [15], DNA computing [13], and computational complexity [12]. We present here only some of the necessary notions and notation.

An *alphabet* is a finite set of symbols (letters), and a word (string) over an alphabet Σ is a finite sequence of letters from Σ; the empty word we denote by λ. The set of all words over an alphabet Σ is denoted by Σ^*. The set of all non-empty words over Σ is denoted as Σ^+, i.e., $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. The length $|x|$ of a word x is the number of symbols that x contains. The empty word has length 0.

We also define circular words over Σ by declaring two words u, v to be equivalent if and only if $u = xy$ and $v = yx$, for some words x, y. We also call u, v *conjugates*. Then the *circular word* $\bullet w$ is the equivalence class of w with respect to this relation, for all $w \in \Sigma^*$. The set of all circular words over Σ is denoted by $\Sigma^\bullet$.

A splicing scheme [5] is a pair $R = (\Sigma, \sim)$, where Σ is an alphabet and $\sim$, the pairing relation of the scheme, $\sim \subseteq \Sigma^*\Sigma^+\Sigma^* \times \Sigma^*\Sigma^+\Sigma^*$. Assume we have two strings x, y and a binary relation between two triples of words $(\alpha, p, \beta) \sim (\alpha', p, \beta')$, such that $x = x'\alpha p\beta x''$ and $y = y'\alpha' p\beta' y''$; then, the strings obtained by the recombination in the context from above are $z_1 = x'\alpha p\beta' y''$ and $z_2 = y'\alpha' p\beta x''$.

When having a pair $(\alpha, p, \beta) \sim (\alpha', p, \beta')$ and two strings x and y as above, $x = x'\alpha p\beta x''$ and $y = y'\alpha' p\beta' y''$, we consider just the string $z_1 = x'\alpha p\beta' y''$ as the result of the recombination (we call it one-output-recombination), because the string $z_2 = y'\alpha' p\beta x''$, we consider as the result of the one-output-recombination with the respect to the symmetric pair $(\alpha', p, \beta') \sim (\alpha, p, \beta)$.

A *Boolean expression* is an expression composed of variables, parentheses and the operators $\bar{\cdot}$, $\wedge$ and $\vee$. The variables can take values 0 (false) and 1 (true). An expression is satisfiable if there is some assignment of variables such that the expression is true. The *satisfiability problem*, commonly denoted as SAT, is to determine, given a Boolean expression, whether it is satisfiable. SAT is a well known NP-complete problem. A Boolean expression is said to be in *conjunctive normal form (CNF)* if it is of the form $E_1 \wedge E_2 \wedge \cdots \wedge E_k$, where each E_i, called a *clause*, is of the form $\alpha_{i1} \vee \alpha_{i2} \vee ... \vee \alpha_{ir_i}$, where each α_{ij} is a literal, that is either x or $\bar{x}$, for some variable x.

3 Guided recombination systems

A splicing scheme is a pair $P = (\Sigma, \sim)$, where Σ is an alphabet and $\sim$, the pairing relation of the scheme, $\sim \subseteq \Sigma^*\Sigma^+\Sigma^* \times \Sigma^*\Sigma^+\Sigma^*$. In the splicing scheme $P = (\Sigma, \sim)$ pairs $(\alpha, p, \beta) \sim (\alpha', p, \beta')$ define the contexts necessary for a recombination between the repeats p. Then the *contextual intramolecular recombination* was defined in [8].

$$\begin{array}{ll} (\mathsf{del}_p) & \{upwpv\} \Longrightarrow_{\mathsf{del}_p} \{upv, \bullet wp\}, \\ \text{where} & u = u'\alpha, w = \beta w' = w''\alpha', v = \beta' v', \text{ and } (\alpha, p, \beta) \sim (\alpha', p, \beta'). \end{array}$$

This constrains intramolecular recombination within $upwpv$ to occur only if the restrictions of the splicing scheme concerning p are fulfilled, i.e., the first occurrence of p is preceded by α and followed by β and its second occurrence is preceded by α' and followed by β'.

Also, if $(\alpha, p, \beta) \sim (\alpha', p, \beta')$, then the *contextual intermolecular recombination* was defined in [8] as

$$\begin{array}{ll} (\mathsf{ins}_p) & \{upv, \bullet wp\} \Longrightarrow_{\mathsf{ins}_p} \{upwpv\} \\ \text{where} & u = u'\alpha, v = \beta v', w = w'\alpha' = \beta' w'', \text{ and } (\alpha, p, \beta) \sim (\alpha', p, \beta'). \end{array}$$

Intermolecular recombination between the linear strand upv and the circular strand $\bullet wp$ may take place only if the occurrence of p in the linear strand is flanked by α and β and its occurrence in the circular strand is flanked by α' and β'.

Definition 1 *For a splicing scheme $P = (\Sigma, \sim)$, we define the set of all contextual gene rearrangement operations under guiding of the splicing scheme P as follows:*

$$\widetilde{P} = \{\mathsf{ins}_p, \mathsf{del}_p \mid (\alpha, p, \beta) \sim (\alpha', p, \beta') \text{ for some } \alpha, \alpha', \beta, \beta' \in \Sigma^*\}.$$

We now define a *guided recombination system* that captures the series of dispersed homologous recombination events that take place during scrambled gene rearrangement in ciliates.

Definition 2 *A guided recombination system is a triple $R = (\Sigma, \sim, t)$ where $(\Sigma, \sim)$ is a splicing scheme, and $t \in \Sigma^+$ is a linear string called the axiom.*

A guided recombination system R defines a derivation relation that produces a new multiset from a given multiset of linear and circular strands, as follows. Starting from a "collection" (multiset) of strings with a certain number of available copies of each string, the next multiset is derived from the first one by an intra- or intermolecular recombination between existing strings. The strands participating in the recombination are "consumed" (their multiplicity decreases by 1) whereas the products of the recombination are added to the multiset (their multiplicity increases by 1).

For two multisets S and S' in $\Sigma^* \cup \Sigma^\bullet$, we say that S derives S' and we write $S \Longrightarrow_R S'$, iff one of the following two cases hold:

(1) there exist $x \in S, y, \bullet z \in S'$ such that

- $\{x\} \Longrightarrow_{\mathsf{del}} \{y, \bullet z\}$ according to an intramolecular recombination step in R,
- $S'(x) = S(x) - 1, S'(y) = S(y) + 1, S'(\bullet z) = S(\bullet z) + 1$, and $S'(u) = S(u)$ for all $u \notin \{x, y, \bullet z\}$;

(2) there exist $x', \bullet y' \in S, z' \in S'$ such that

- $\{x', \bullet y'\} \Longrightarrow_{\mathsf{ins}} \{z'\}$ according to an intermolecular recombination step in R,
- $S'(x') = S(x') - 1, S'(\bullet y') = S(\bullet y') - 1, S'(z') = S(z') + 1$, and $S'(u) = S(u)$ for all $u \notin \{x', y', \bullet z'\}$.

Those strands which, by repeated recombinations with initial and intermediate strands eventually produce the axiom, form the language of the guided recombination system. Formally,

$$L_a^k(R) = \{w \in \Sigma^* \mid \{(w, k)\} \Longrightarrow_R^* S \text{ and } t \in S\}$$

((w, k) indicates the fact that the multiplicity of w equals k).

The guided recombination systems are proved in [8] to be equivalent to Turing machine:

Theorem 1 ([8]) *Let L be a language over T^* accepted by a Turing machine $TM = (S, \Sigma \cup \{\#\}, P)$. Then there exist an alphabet Σ', a sequence $\pi \in \Sigma'^*$, depending on L, and a recombination system R such that a word w over T^* is in L iff $\#^6 s_0 w \#^6 \pi$ belongs to $L_a^k(R)$ for some $k \geq 1$.*

In line with this result, we define acceptance for guided recombination systems as follows.

Definition 3 *We say a guided recombination system $R = (\Sigma, \sim, t)$* accepts *a string w iff there exists a $k \geq 1$ such that $w \in L_a^k(R)$.*

In other words, a guided recombination system accepts a string w if it generates the axiom, when starting with some (sufficient) amount of copies of w.

We now consider the parallelism for the guided recombination model. Intuitively, a number of operations can be applied in parallel to a string if the applicability of each operation is independent of the applicability of the other operations.

In this paper we use a notion of parallelism following [6], which is the *maximally parallel* application of a rule to a string.

First, we define the working places of a operation $\pi \in \widetilde{P}$ on a given string where π is applicable.

Definition 4 *Let w be a string. The working places of a operation $\pi \in \widetilde{P}$ with respect to a multiset S for w is a set of substrings of w written as $Wp(\pi(w))$ and defined by*

$$\begin{aligned} Wp(\mathsf{del}_p(w)) &= \{upw'pv \in Sub(w) \mid \{upw'pv\} \Longrightarrow_{\mathsf{del}_p} \{upv, \bullet w'p\}\}, \\ Wp(\mathsf{ins}_p(w)) &= \{upv \in Sub(w) \mid \{upv, \bullet w'p\} \Longrightarrow_{\mathsf{ins}_p} \{upw'pv\} \\ &\qquad \textit{for some } \bullet w'p \in S\}. \end{aligned}$$

Definition 5 *Let w be a string. The smallest working places of a operation $\pi \in \widetilde{P}$ for w is a subset of $Wp(\pi(w))$ written as $Wp_s(\pi(w))$ and defined by*

$$\begin{aligned} Wp_s(\pi(w)) &= \{w_1 \in Wp(\pi(w)) \mid \text{ for all } w_1' \in Sub(w_1) \\ &\qquad \text{and } w_1' \neq w_1, w_1' \notin Wp(\pi(w))\}. \end{aligned}$$

Definition 6 *Let Σ be a finite alphabet and $\widetilde{P}$ the set of rules defined above. Let $\pi \in \widetilde{P}$ and $u \in \Sigma^*$. We say that $v \in \Sigma^*$ is obtained from u by applying π in a maximally parallel way, denoted $u \Longrightarrow_{\pi}^{max} v$, if*

$$u = \alpha_1 u_1 \alpha_2 u_2 \dots \alpha_k u_k \alpha_{k+1}, \text{ and } v = \alpha_1 v_1 \alpha_2 v_2 \dots \alpha_k v_k \alpha_{k+1},$$

where $u_i \in Wp_s(\pi)(u)$, $v_i \in \Sigma^*$ for all $1 \leq i \leq k$, and also, $\alpha_i \notin Wp(\pi(u))$, for all $1 \leq i \leq k+1$.

Example 1 *Let del_p be the contextual deletion operation applied in the context $(x_1x_2, p, x_3) \sim (x_3, p, x_1)$, and consider the string $u = x_1x_2px_3px_1x_2p\ x_3px_1$. The unique correct result obtained by maximally parallel application of del_p to u is:*

$$x_1x_2px_3px_1x_2px_3px_1 \Longrightarrow_{\mathsf{del}_p}^{max} x_1x_2px_1x_2px_1.$$

Finally, if in a guided recombination system $R = (\Sigma, \sim, t)$ for some multiplicity k $\{(w,k)\} \Longrightarrow_R^n S$, with $t \in S$, we say that R accepts a string w in time n.

4 Efficiency of guided recombination systems

In this section, we use guided recombination systems as decision problem solvers. A possible correspondence between decision problems and languages can be done via an encoding function which transforms an instance of a given decision problem into a word, see, e.g., [3].

Definition 7 *We say that a decision problem X is solved in time $O(t(n))$ by guided recombination systems if there exists a family $\mathcal{A}$ of guided recombination systems such that the following conditions are satisfied:*

1. *The encoding function of any instance x of X having size n can be computed by a deterministic Turing machine in time $O(t(n))$.*
2. *For each instance x of size n of the problem one can effectively construct, in time $O(t(n))$, an intermolecular guided recombination system $G(x) \in \mathcal{A}$ which accepts, again in time $O(t(n))$, the word encoding the instance x if and only if the solution to the given instance of the problem is* `YES`.

Moreover, we say that a solution is *uniform* if all instances of the same size are solved by the same guided recombination system.

Theorem 2 `SAT` *can be solved uniformly and deterministically by a guided recombination system in time* $O(n \cdot m)$*, where* n *denotes the number of variables and* m *the number of clauses.*

Proof. Let us consider a propositional formula ϕ of m clauses over n variables in the conjunctive normal form. Thus $\phi = C_1 \wedge \cdots \wedge C_m$, such that each clause $C_j, 1 \leq j \leq m$, is of the form $C_j = \langle y_{j,1} \vee \cdots \vee y_{j,k_j} \rangle, k_j \geq 1,$ where $y_{j,k} \in \{x_i, \bar{x}_i \mid 1 \leq i \leq n\}, 1 \leq k \leq k_j$.

We encode each clause C_j as a string bounded by $\$_j$ in the following form:

$$c_j = \$_j \dagger C_j \dagger \$_j.$$

The instance ϕ is encoded as follows:

$$\hat{\phi} = c_1 \ldots c_m \$_{m+1} \dagger\dagger x_1 \dagger\dagger \bar{x}_1 \dagger\dagger \ldots \dagger\dagger x_n \dagger\dagger \bar{x}_n \dagger\dagger \$_{m+1} \$_{m+2}.$$

It is easily seen that the size of the encoding is linear in n and m.

The string appended to the formula contains both values for all variables in ϕ. We design a guided recombination system which solves the encoded instance of `SAT` in the following steps.

1. Excise the variable values.
2. Insert a valued variable after each clause.
3. Check if the inserted variable satisfies the clause.
4. Check if the inserted variables are consistent.
5. Generate the axiom if and only if both checks are successful.

Specifically, given a boolean formula ϕ with m clauses and over n variables, we construct a guided recombination system

$$G = (\Sigma, \sim, \$),$$

with

$$\begin{aligned} \Sigma &= \{\$_i \mid 1 \leq i \leq m+2\} \cup \{x_i, \bar{x}_i \mid 1 \leq i \leq n\} \cup \{\vee, \langle, \rangle, \dagger\}, \\ \$ &= \$_1 \$_2 \ldots \$_m \$_{m+1} \$_{m+2}. \end{aligned}$$

The relation $\sim$ is defined as follows, where $x \in \{x_i, \bar{x}_i \mid 1 \leq i \leq n\}$, $b \in \{\vee, \langle\}$, $e \in \{\vee, \rangle\}$ and $1 \leq j \leq m$. Also $\bar{x} = \bar{x}_i$ if $x = x_i$ and $\bar{x} = x_i$ if $x = \bar{x}_i$.

$$(\dagger, \dagger, x) \sim (x, \dagger, \dagger), \tag{1}$$
$$(\rangle, \dagger, \$_j \$_{j+1}) \sim (x, \dagger, x), \tag{2}$$
$$(b, x, e) \sim (\rangle \dagger, x, \dagger), \tag{3}$$
$$(bx, \dagger, \lambda) \sim (b\bar{x} \dagger \$_j \$_{j+1}, \dagger, \langle), \tag{4}$$
$$(\dagger \$_m, \$_{m+1}, \lambda) \sim (\lambda, \$_{m+1}, \$_{m+2}), \tag{5}$$
$$(\lambda, \$_j, \dagger) \sim (bx \dagger, \$_j, \$_{j+1} \$_{j+2}). \tag{6}$$

The size of this system is $O(n \cdot m)$ and it is not hard to see that it can be constructed by a deterministic Turing machine in time $O(n \cdot m)$.

We will show that G decides SAT for a given input ϕ. That is, that G accepts encoding $\hat{\phi}$ if and only if ϕ is satisfiable.

For the *if*-part, consider the input string

$$\$_1 \dagger C_1 \dagger \$_1 \ldots \$_m \dagger C_m \dagger \$_m \$_{m+1} \dagger\dagger x_1 \dagger\dagger \bar{x}_1 \dagger\dagger \ldots \dagger\dagger x_n \dagger\dagger \bar{x}_n \dagger\dagger \$_{m+1} \$_{m+2}.$$

To this word we can apply the operation $\mathsf{del}_\dagger$ using contexts of (1). In fact, we apply $2n$ $\mathsf{del}_\dagger$-operations in parallel, giving

$$\$_1 \dagger C_1 \dagger \$_1 \ldots \$_m \dagger C_m \dagger \$_m \$_{m+1} \dagger^{2n+2} \$_{m+1} \$_{m+2}$$

as well as the circular strings $\bullet x \dagger$ for all $x \in \{x_i, \bar{x}_i \mid 1 \leq i \leq n\}$.

In the next step, these circular strings can be inserted after each encoding of a clause of ϕ using contexts (2). Again, this is done in parallel for all clauses, so with m $\mathsf{ins}_\dagger$-operations we obtain a string of the form

$$\$_1 \dagger C_1 \dagger z_1 \dagger \$_1 \ldots \$_m \dagger C_m \dagger z_m \dagger \$_m \$_{m+1} \dagger^{2n+2} \$_{m+1} \$_{m+2}$$

with each $z_j, 1 \leq j \leq m$ in $\{x_i, \bar{x}_i \mid 1 \leq i \leq n\}$. We interpret these inserted variables as an assignment, where the variable inserted after each clause verifies this clause. It is important to note that the same variable can be inserted more than once, up to m times, into the same string, since we also have at our disposal circular strings excised from other copies of the input string. Recall that by Definition 3 we can assume that the input word is present in the multiplicity needed to generate all possible assignments of a verifying variable to a clause. If $m > 2n$, extra multiplicity is needed to provide enough variables to insert.

If the formula is satisfiable, there is at least one inserted assignment in which all inserted variables effectively verify the clause preceding it. In this case, we can apply the contexts of (3) to perform m del_x-operations. This gives m strings of the form $\bullet \vee \cdots \vee y_{j,k_j} \rangle \dagger z_i$ and a string

$$\$_1 \dagger \langle y_{1,1} \vee \cdots \vee z_1 \dagger \$_1 \ldots \$_m \dagger \langle y_{m,1} \vee \cdots \vee z_m \dagger \$_m \$_{m+1} \dagger^{2n+2} \$_{m+1} \$_{m+2}.$$

Recall that each clause $C_j, 1 \leq j \leq m$, is of the form $C_j = \langle y_{j,1} \vee \cdots \vee y_{j,k_j} \rangle, k_j \geq 1$, where $y_{j,k} \in \{x_i, \bar{x}_i \mid 1 \leq i \leq n\}, 1 \leq k \leq k_j$.

Again, if the formula is satisfiable, there is at least one inserted assignment which verifies all clauses and is consistent, in the sense that if a variable x_i is inserted in some place, no variable $\bar{x}_i$ is inserted after another clause. This means no context of (4) can apply to the string, since this requires the simultaneous presence of x_i and $\bar{x}_i$ in the string.

Now we can apply $\mathsf{del}_{\$_{m+1}}$ using context (5). In this and the following steps, no parallel applications are possible, so the derivation goes on sequentially. First, $\mathsf{del}_{\$_{m+1}}$ yields $\bullet \dagger^{2n+2} \$_{m+1}$ and

$$\$_1 \dagger \langle y_{1,1} \vee \cdots \vee z_1 \dagger \$_1 \ldots \$_m \dagger \langle y_{m,1} \vee \cdots \vee z_m \dagger \$_m \$_{m+1} \$_{m+2}.$$

From here, we apply successively $\mathsf{del}_{\$_m}$ to $\mathsf{del}_{\$_1}$ using the contexts of (6). For instance, $\mathsf{del}_{\$_m}$ gives $\bullet\dagger\langle y_{m,1} \vee \cdots \vee z_m\dagger\$_m$ and

$$\$_1\dagger\langle y_{1,1} \vee \cdots \vee z_1\dagger\$_1 \ldots \$_{m-1}\dagger\langle y_{m-1,1} \vee \cdots \vee z_{m-1}\dagger\$_{m-1}\$_m\$_{m+1}\$_{m+2},$$

after which $\mathsf{del}_{\$_{m-1}}$ can be applied. This process goes on until $\mathsf{del}_{\$_1}$ yields the axiom

$$\$_1\$_2 \ldots \$_{m+1}\$_{m+2}.$$

This means G accepts $\hat{\phi}$.

For the *only if*-part, assume that ϕ is not satisfiable. In this case, the first two steps go on exactly as described above, giving

$$\$_1\dagger C_1\dagger z_1\dagger\$_1 \ldots \$_m\dagger C_m\dagger z_m\dagger\$_m\$_{m+1}\dagger^{2n+2}\$_{m+1}\$_{m+2}.$$

However, ϕ is not satisfiable. This means that for all inserted assignments at least one of the following holds:

1. Not all clauses are verified by the variable inserted after them.
2. The inserted assignment is inconsistent.

For case 1, assume that only clause C_l is not verified by z_l. Then we cannot apply the m parallel del_x-operations as before. In fact, we can only apply $m-1$ operations giving

$$\$_1\dagger\langle y_{1,1} \vee \cdots \vee z_1\dagger\$_1 \ldots \$_l\dagger\langle \ldots\rangle\dagger z_l\dagger\$_l \ldots \$_m\dagger\langle y_{m,1} \vee \cdots \vee z_m\dagger\$_m\$_{m+1}\dagger^{2n+2}\$_{m+1}\$_{m+2}.$$

Alternatively, if $m > 2n$, z_l may not have been inserted. Also then del_{z_l} cannot be applied and we get

$$\$_1\dagger\langle y_{1,1} \vee \cdots \vee z_1\dagger\$_1 \ldots \$_l\dagger\langle \ldots\rangle\dagger\$_l \ldots \$_m\dagger\langle y_{m,1} \vee \cdots \vee z_m\dagger\$_m\$_{m+1}\dagger^{2n+2}\$_{m+1}\$_{m+2}.$$

In both of these cases, if z_{l+1} happens to verify clause l, we could apply $\mathsf{del}_{z_{l+1}}$ differently, resulting in $\bullet \vee \ldots\rangle(\dagger z_l)\dagger\$_l\$_{l+1}\dagger \ldots\rangle\dagger z_l$ and

$$\$_1\dagger\langle y_{1,1}\vee\cdots\vee z_1\dagger\$_1 \ldots \$_l\dagger\langle \ldots z_{l+1}\dagger\$_{l+1} \ldots \$_m\dagger\langle y_{m,1}\vee\cdots\vee z_m\dagger\$_m\$_{m+1}\dagger^{2n+2}\$_{m+1}\$_{m+2}.$$

A similar situation can arise if z_l happens to verify clause $l-1$. Then, instead of $\mathsf{del}_{z_{l-1}}$ we could also apply del_{z_l} , resulting in $\bullet \vee \ldots\rangle\dagger z_{l-1}\dagger\$_{l-1}\$_l\dagger \ldots\rangle\dagger z_l$ and

$$\$_1\dagger\langle y_{1,1} \vee \cdots \vee z_1\dagger\$_1 \ldots \$_{l-1}\dagger\langle \ldots z_l\dagger\$_l \ldots \$_m\dagger\langle y_{m,1} \vee \cdots \vee z_m\dagger\$_m\$_{m+1}\dagger^{2n+2}\$_{m+1}\$_{m+2}.$$

By our maximality requirement, these are the only possibilities. We suppose that no $\mathsf{del}_\dagger$ using contexts (4) can be applied (if it can, this is treated under case 2). Now, $\mathsf{del}_{\$_{m+1}}$ using context (5) is applied as before. Also $\mathsf{del}_{\$_j}$ by (6) until arriving at $\mathsf{del}_{\$_l}$ (or $\mathsf{del}_{\$_{l+1}}$). None of the strings obtained above satisfy the context of (6) at that point, so the derivation of the axiom cannot continue. No other operations can take place, so G does not accept $\hat{\phi}$ by this assignment. If more than one variables

do not satisfy their clause, the situation is the same, except that we can get more substrings of the form $\$_l\dagger\langle\ldots\rangle(\dagger z_l)\dagger\$_l$ or $\$_k\dagger\langle\ldots z_l\dagger\$_l, k < l$.

For case 2, assume all variables satisfy their clauses. In this case, m del_x-operations apply as before, giving

$$\$_1\dagger\langle y_{1,1} \vee \cdots \vee z_1\dagger\$_1 \ldots \$_m\dagger\langle y_{m,1} \vee \cdots \vee z_m\dagger\$_m\$_{m+1}\dagger^{2n+2}\$_{m+1}\$_{m+2}.$$

Now suppose that z_p and z_q are inconsistent, for $p < q$. Now, in the same step as $\mathsf{del}_{\$_{m+1}}$ we also apply $\mathsf{del}_\dagger$ by context (4). This gives $\bullet\$_p \cdots \vee z_q\dagger\$_q\$_{q+1}\dagger$ and

$$\$_1\dagger\langle y_{1,1} \vee \cdots \vee z_1\dagger\$_1 \cdots \vee z_p\dagger\langle\ldots\rangle z_{q+1}\dagger\$_{q+1} \ldots\ldots \$_m\dagger\langle y_{m,1} \vee \cdots \vee z_m\dagger\$_m\$_{m+1}\$_{m+2}.$$

Note that if case 1 holds for any z_l unequal to z_p and z_q, the same $\mathsf{del}_\dagger$ will still take place. As in case 1, the created string does not satisfy the context of $\mathsf{del}_{\$_q}$, so the axiom cannot be derived.

Since for all possible assignments at least one case holds, the axiom is not generated, so G does not accept $\hat{\phi}$.

Finally, since each context can induce both a del and an ins-operation, we should say a few words about the operations not mentioned before.

- $\mathsf{ins}_\dagger$ by context (1) is not possible since we never have any circular string with two consecutive symbols †.
- $\mathsf{del}_\dagger$ by (2) cannot happen because we never have a substring $x\dagger x$ in a circular string.
- ins_x by (3) is impossible because none of the circular strings we obtain has substring $\rangle\dagger x\dagger$.
- $\mathsf{ins}_\dagger$ by (4) cannot take place since none of the circular strings contains substring $\vee x\$_j\$_{j+1}\dagger\langle$ or $\langle x\$_j\$_{j+1}\dagger\langle$.
- $\mathsf{ins}_{\$_{m+1}}$ by (5) is not possible because no circular string contains $\$_{m+1}\$_{m+2}$ (the circular strings generated by del_x (3) and $\mathsf{del}_\dagger$ (4) only contain $\$_1$ to $\$_m$).
- $\mathsf{ins}_{\$_{m+1}}$ by (6) needs a circular string containing $\$_j\$_{j+1}\$_{j+2}$, which is never created.

Our algorithm has a linear running time. Excision of the variables takes at most m steps, since we need to excise from at most m copies of the input string. If the formula is satisfiable, we obtain the axiom after $m + 3$ additional steps, giving a total running time of $2m + 3$ steps. Finally, the system G we constructed solves all instances of SAT of m clauses over n variables, thus making our solution uniform. This concludes the proof. □

5 Conclusion

In this paper we considered a model of gene rearrangement in ciliates. We showed that this model can be used as an efficient problem solving device by presenting an algorithm for solving `SAT` in time $O(n \cdot m)$. One especially interesting feature of our algorithm is that we show that using small local contexts one can perform correctness and consistency checks over arbitrarily large distances. We believe that the study of the gene rearrangement process in ciliates and its formal modelling is not only interesting from a biological point of view, but can also be beneficial for the study of computation.

Acknowledgments. The work of T.-O.I. is supported by the Center for International Mobility (CIMO) Finland, grant TM-06-4036 and by Academy of Finland, project 203667. The work of R.L. was supported by Research Grant BES-2004-6316 of the Spanish Ministry of Education and Science. The work of I.P. is supported by Academy of Finland, project 108421.

References

[1] Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D. M., and Rozenberg, G., *Computation in Living Cells: Gene Assembly in Ciliates*, Springer, 2003.

[2] Ehrenfeucht, A., Prescott, D. M., and Rozenberg, G., Computational aspects of gene (un)scrambling in ciliates. In: L. F. Landweber, E. Winfree (eds.) *Evolution as Computation*, Springer, Berlin, Heidelberg, New York, 216–256, 2001.

[3] Garey, M., Jonhson, D., *Computers and Intractability. A Guide to the Theory of NP-completeness*, Freeman, San Francisco, CA, 1979.

[4] Harju, T., Petre, I., and Rozenberg, G., Two models for gene assembly in ciliates, *Theory is forever, LNCS* 3113:89–101, 2004.

[5] Head, T., Formal Language Theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. *Bull. Math. Biology* 49: 737–759, 1987.

[6] Ishdorj, T.-O, and Petre, I., Computing through gene assembly, accepted in *Int. Conf. UC'07*, 13–17, August, 2007, Kingston, Canada.

[7] Ishdorj, T.-O, Petre, I., and Rogojin, V., Computational power of intramolecular gene assembly, *International Journal of Foundations of Computer Science*, to appear, 2007.

[8] Kari, L., and Landweber, L. F., Computational power of gene rearrangement. In: E. Winfree and D. K. Gifford (eds.) *Proceedings of DNA Based Computers, V* American Mathematical Society, 207–216, 1999.

[9] Kari, L., and Thierrin, G., Contextual insertion/deletions and computability. *Information and Computation* 131:47–61, 1996.

[10] Landweber, L. F., and Kari, L., The evolution of cellular computing: Nature's solution to a computational problem. In: *Proceedings of the 4th DIMACS Meeting on DNA-Based Computers*, Philadelphia, PA, 3–15, 1998.

[11] Loos, R., Martín-Vide, C., and Mitrana, V., Solving `SAT` and `HPP` with accepting splicing systems, *PPSN IX, LNCS* 4193:771–777, 2006.

[12] Papadimitriou, Ch. P., *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[13] Păun, Gh., Rozenberg, G., Salomaa, A., *DNA Computing - New computing paradigms*, Springer-Verlag, Berlin, 1998.

[14] Prescott, D. M., Ehrenfeucht, A., and Rozenberg, G., Molecular operations for DNA processing in hypotrichous ciliates. *Europ. J. Protistology* **37** (2001) 241–260.

[15] Salomaa, A., *Formal Languages*, Academic Press, New York, 1973.