Computing Through Gene Assembly

Tseren-Onolt Ishdorj[1,3] and Ion Petre[1,2]

[1]Computational Biomodelling Laboratory
Department of Information Technologies
Åbo Akademi University, Turku 20520 Finland
E-mail:{tishdorj,ipetre}@abo.fi
[2]TUCS, Turku 20520 Finland
[3]Research Group on Natural Computing
Department of CS and AI, Sevilla University
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

Abstract. The intramolecular gene assembly model, [1], uses three molecular recombination operations ld, dlad, and hi. A computing model with two contextual recombination operations del and trl, which are based on ld and dlad, respectively, is considered in [6] and its computational power is investigated. In the present paper, we expand the computing model with a new molecular operation such as cpy - *copy*. Then we prove that the extended contextual intramolecular gene assembly model is both computationally universal and efficient.

1 Introduction

There have been proposed two formal models to explain the gene assembly process in ciliates: intermolecular model, for instance in [8], and intramolecular model, for instance in [1]. They both are based on so called "pointers" - short nucleotide sequences (about 20 bp) lying on the borders between coding and non-coding blocks. Each next coding block starts with a pointer-sequence repeating exactly the pointer-sequence in the end of the preceding codding block from the assembled gene. It is supposed that the pointers guide the alignment of coding blocks during the gene assembly process. The intramolecular model proposes three operations: ld (loop with direct pointers), hi (hairpin loop with inverted pointers), dlad (double loop with alternating direct pointers).

The context sensitive variants of the intramolecular operations dlad and ld have been considered in [6]. The accepting contextual intramolecular recombination (AIR) system using the operations *translocation* based on (dlad) and *deletion* based on (ld) has been proved to be equivalent in power with Turing machine.

In the present paper, we expand the set of contextual intramolecular operations with a new operation: *copy* cpy_p. Intramolecular *copy* operation is first considered in [14]. The extended intramolecular recombination model with three types of contextual operations *translocation, deletion*, and *copy* computing along a single string is as powerful as Turing machine, and efficient enough to solve **NP**-complete problems (in particular, SAT) in polynomial time.

No ciliate-based efficient algorithm for intractable problems has been presented in the literature so far. This makes the efficiency result in this paper novel in the theory of ciliate computation.

A particularly interesting feature of the system is that the recombination operations are applied in a maximally parallel manner.

To solve an instance of SAT, we encode the problem into a string. Then during the recombination steps, the encoded problem (propositional formula) is duplicated by *copy* operation along the string. Meanwhile, the truth-assignments attached to the propositional formula are constructed. In the end, all possible truth-assignments are generated in a linear number of steps, and each truth-assignment is checked to see whether it satisfies the encoded formula.

The universality result of [6] uses a number of copies of a given substring along the working string in order to start a computation. By extension of the model with the *copy* operation, the necessary substrings will be generated during the computation instead of making the working string as full storage in advance as in [6]. This saves on the size and complexity of the encoding.

2 Preliminaries

We assume the reader to be familiar with the basic elements of formal languages and Turing computability [13], DNA computing [12], and computational complexity [10]. We present here only some of the necessary notions and notation.

Using an approach developed in a series of works (see [9], [11], [2], and [7]) we use *contexts* to restrict the application of molecular recombination operations, [12], [1]. First, we give the formal definition of splicing rules. Consider an alphabet Σ and two special symbols, $\#, \$ \notin \Sigma$. A *splicing rule* (over Σ) is a string of the form $r = u_1\#u_2\$u_3\#u_4$, where $u_1, u_2, u_3, u_4 \in \Sigma^*$. (For a maximal generality, we place no restriction on the strings u_1, u_2, u_3, u_4. The cases when $u_1u_2 = \lambda$ or $u_3u_4 = \lambda$ could be ruled out as unrealistic.)

For a splicing rule $r = u_1\#u_2\$u_3\#u_4$ and strings $x, y, z \in \Sigma^*$ we write $(x, y) \vdash_r z$ if and only if $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$, $z = x_1u_1u_4y_2$, for some $x_1, x_2, y_1, y_2 \in \Sigma^*$. We say that we *splice* x, y at the *sites* u_1u_2, u_3u_4, respectively, and the result is z. This is the basic operation of DNA molecule recombination.

A splicing scheme [5] is a pair $R = (\Sigma, \sim)$, where Σ is the alphabet and $\sim$, the pairing relation of the scheme, $\sim \subseteq (\Sigma^+)^3 \times (\Sigma^+)^3$. Assume we have two strings x, y and a binary relation between two triples of nonempty 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 rewriting system $M = (S, \Sigma \cup \{\#\}, P)$ is called a *Turing machine* (we use also abbreviation TM), [13], where: (i) S and $\Sigma \cup \{\#\}$, where $\# \notin \Sigma$ and $\Sigma \neq \emptyset$, are two disjoint sets referred to as the *state* and the *tape* alphabets; we fix a symbol from Σ, denote it as $\sqcup$ and call it "blank symbol"; (ii) Elements s_0 and s_f of S are the *initial* and the *final* states respectively; (iii) The productions (rewriting rules) of P are of the forms

(1) $s_i a \longrightarrow s_j b$; (2) $s_i ac \longrightarrow as_j c$; (3) $s_i a\# \longrightarrow as_j \sqcup \#$; (4) $cs_i a \longrightarrow s_j ca$; (5) $\#s_i a \longrightarrow \#s_j \sqcup a$; (6) $s_f a \longrightarrow s_f$; (7) $as_f \longrightarrow s_f$, where s_i and s_j are states in S, $s_i \neq s_f$, and a, b, c are in Σ.

The TM M changes from one configuration to another one according to its set of rules P. We say that the Turing machine M *halts* with a word w if there exists a computation such that, when started with the read/write head positioned at the beginning of w, the TM eventually reaches the final state, i.e., if $\#s_0 w\#$ derives $\#s_f\#$ by successive applications of the rewriting rules (1)–(7) from P. The language $L(M)$ *accepted* by the TM M is the set of words on which M halts. If TM is deterministic, then there is the only computation possible for each word. The family of languages accepted by Turing machines is equivalent to the family of languages accepted by deterministic Turing machines.

The satisfiability problem, SAT, is a well known **NP**-complete problem. It asks whether or not for a given propositional formula in the conjunctive normal form there is a truth-assignment of variables such that the formula assumes the value *true*. The details of SAT are considered in Section 4.

3 The Contextual Intramolecular Operations

The contextual intramolecular *translocation* and *deletion* operations are the generalizations of dlad and ld operations, respectively, as defined in [6]. We consider a splicing scheme $R = (\Sigma, \sim)$. Denote $core(R) = \{p \mid (\alpha, p, \beta) \sim (\gamma, p, \delta) \in R$ for some $\alpha, \beta, \gamma, \delta \in \Sigma^+\}$.

Definition 1. *The contextual intramolecular translocation operation with respect to R is defined as $\mathsf{trl}_{p,q}(xpuqypvqz) = xpvqypuqz$, where there are such relations $(\alpha, p, \beta) \sim (\alpha', p, \beta')$ and $(\gamma, q, \delta) \sim (\gamma', q, \delta')$ in R, that $x = x'\alpha$, $uqy = \beta u' = u''\alpha'$, $vqz = \beta' v'$, $xpu = x''\gamma$, $ypv = \delta y' = y''\gamma'$ and $z = \delta' z'$.*

We say that operation $\mathsf{trl}_{p,q}$ is applicable, if the contexts of the two occurrences of p as well as the contexts of the two occurrences of q are in the relation $\sim$. Substrings p and q we call *pointers*. In the result of application of $\mathsf{trl}_{p,q}$ strings u and v, each flanked by pointers p and q, are swapped. If from the non-empty word u we get by $\mathsf{trl}_{p,q}$ operation word v, we write $u \Longrightarrow_{\mathsf{trl}_{p,q}} v$ and say that v is obtained from u by $\mathsf{trl}_{p,q}$ operation. The *translocation* is depicted in Fig. 1.

Definition 2. *The contextual intramolecular deletion operation with respect to R is defined as $\mathsf{del}_p(xpupy) = xpy$, where there is a relation $(\alpha, p, \beta) \sim (\alpha', p, \beta')$ in R that $x = x'\alpha$, $u = \beta u' = u''\alpha'$, and $y = \beta' y'$.*

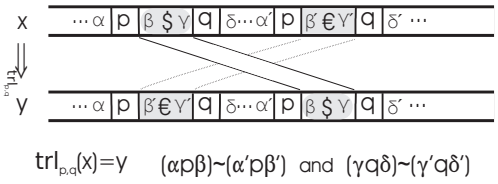

Fig. 1. Translocation operation.

In the result of applying del_p, the string u flanked by two occurrences of p is removed, provided that the contexts of those occurrences of p are in the relation $\sim$. If from a non-empty word u we obtain a word v by del_p, we write $u \Longrightarrow_{\mathsf{del}_p} v$ and say that the word v is obtained from u by del_p operation. See Fig. 2 A.

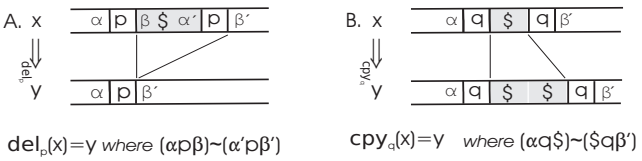

Fig. 2. A. Deletion and B. Copy operations.

We introduce here an additional operation *the contextual intramolecular copy.*

Definition 3. *The contextual intramolecular copy operation with respect to R is defined as $\mathsf{cpy}_q(xquqy) = xquuqy$, where there is a relation $(\alpha, q, \beta) \sim (\alpha', q, \beta')$ in R that $x = x'\alpha$, $u = \beta = \alpha'$, $y = \beta' y'$.*

As a result of applying cpy_q, the substring u flanked by two occurrences of q is duplicated. The substring u supposed to be duplicated itself must be also the following substring of the first pointer q and the preceding substring of the second occurrence of q. If from the non-empty word u we obtain a word v by cpy_q, we write $u \Longrightarrow_{\mathsf{cpy}_q} v$ and say that the word v is obtained from u by cpy_q operation. The *copy* operation is depicted in Fig. 2 B.

We use the next notations:

$$\begin{aligned} \mathsf{trl} &= \{\mathsf{trl}_{p,q} \mid p, q \in \mathit{core}(R)\}, \\ \mathsf{del} &= \{\mathsf{del}_p \mid p \in \mathit{core}(R)\}, \\ \mathsf{cpy} &= \{\mathsf{cpy}_p \mid p \in \mathit{core}(R)\}. \end{aligned}$$

Then we define the set of all contextual intramolecular operations as follows: $\widetilde{R} = \mathsf{trl} \cup \mathsf{del} \cup \mathsf{cpy}$.

Definition 4. *Let a triplet $R_{op} = (\Sigma, \sim, op)$ be a splicing scheme that an operation $op \in \widetilde{R}$ performs only in the contexts defined by the set of splicing relations in R_{op}. We say that R_{op} is an operation class of the splicing scheme $\sim$ for the operation $op \in \widetilde{R}$.*

We consider the parallelism for the intramolecular 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. The parallelism in intramolecular gene assembly was initially studied in a different setting in [4]. We recall the definition of parallelism following [4] with a small modification adapted to our model.

Definition 5. *Let $S \subseteq R$ be a set of k rules and let u be a string. We say that the rules in S can be applied in parallel to u if for any ordering $\varphi_1, \varphi_2, \ldots, \varphi_k$ of S, the composition $\varphi_k \circ \varphi_{k-1} \circ \cdots \circ \varphi_1$, is applicable to u.*

In our proof below we use a different notion of parallelism. We introduce it here is the form of the *maximally parallel* application of a rule to a string. First, we define the working places of a operation $\varphi \in \widetilde{R}$ on a given string where φ is applicable.

Definition 6. *Let w be a string. The working places of a operation $\varphi \in \widetilde{R}$ for w is a set of substrings of w written as $Wp(\varphi(w))$ and defined by*

$$
\begin{aligned}
Wp(\mathsf{trl}_{p,q}(w)) &= \{(w_1, w_2) \in Sub(w) \mid \ \mathsf{trl}_{p,q}(xw_1yw_2z) = xw_2yw_1z\}.\\
Wp(\mathsf{del}_{p}(w)) &= \{w_1 \in Sub(w) \mid \ \mathsf{del}_{p}(xpw_1py) = xpy\}.\\
Wp(\mathsf{cpy}_{p}(w)) &= \{w_1 \in Sub(w) \mid \ \mathsf{cpy}_{p}(xw_1y) = xw_1w_1y\}.
\end{aligned}
$$

Definition 7. *Let w be a string. The smallest working places of an operation $\varphi \in \widetilde{R}$ for w is a subset of $Wp(\varphi)(w)$ written as $Wp_s(\varphi(w))$ and defined by*

$$
\begin{aligned}
Wp_s(\mathsf{trl}_{p,q}(w)) = \{&(w_1, w_2) \in Wp(\mathsf{trl}_{p,q}(w)) \mid \ \textit{for all } w_1' \in Sub(w_1) \textit{ and}\\
&w_2' \in Sub(w_2) \textit{ and } (w_1', w_2') \neq (w_1, w_2),\\
&(w_1', w_2') \notin Wp(\mathsf{trl}_{p,q}(w))\}.\\
Wp_s(\mathsf{del}_{p}(w)) = \{&w_1 \in Wp(\mathsf{del}_{p}(w)) \mid \ \textit{for all } w_1' \in Sub(w_1),\\
&\textit{and } w_1' \neq w_1, w_1' \notin Wp(\mathsf{del}_{p}(w))\}.\\
Wp_s(\mathsf{cpy}_{p}(w)) = \{&w_1 \in Wp(\mathsf{cpy}_{p}(w)) \mid \ \textit{for all } w_1' \in Sub(w_1),\\
&\textit{and } w_1' \neq w_1, w_1' \notin Wp(\mathsf{cpy}_{p}(w))\}.
\end{aligned}
$$

Definition 8. *Let Σ be a finite alphabet and $\widetilde{R}$ the set of rules defined above. Let $\varphi \in R$ and $u \in \Sigma^*$. We say that $v \in \Sigma^*$ is obtained from u by applying φ in a maximally parallel way, denoted $u \Longrightarrow_{\varphi}^{max} v$, if $u = \alpha_1 u_1 \alpha_2 u_2 \ldots \alpha_k u_k \alpha_{k+1}$,* and *$v = \alpha_1 v_1 \alpha_2 v_2 \ldots \alpha_k v_k \alpha_{k+1}$, where $u_i \in Wp_s(\varphi)(w)$ for all $1 \leq i \leq k$, and also, $\alpha_i \notin Wp(\varphi(w))$, for all $1 \leq i \leq k+1$.*

Note that a rule $\varphi \in \widetilde{R}$ may be applied in parallel to a string in several different ways, as shown in the next example.

Example 1. Let $\mathsf{trl}_{p,q}$ be the contextual translocation operation applied in the context $(x_1, p, x_2) \sim (x_3, p, x_4)$ and $(y_1, q, y_2) \sim (y_3, q, y_4)$. We consider the string $u = x_1px_2\$_1y_1qy_2\$_2x_3px_4\$_3x_3px_4\$_4y_3qy_4$.

Note that there are two occurrences of p with context x_3 and x_4. We can obtain two different strings from u by applying $\mathsf{trl}_{p,q}$ in a parallel way as follows:

$$u \Longrightarrow^{max}_{\mathsf{trl}_{p,q}} v' \text{ where } v' = x_1px_4\$_3x_3px_4\$_4y_3qy_2\$_2x_3px_2\$_1y_1qy_4.$$
$$u \Longrightarrow^{max}_{\mathsf{trl}_{p,q}} v'' \text{ where } v'' = x_1px_4\$_4y_3qy_2\$_2x_3px_4\$_3x_3px_2\$_1y_1qy_4.$$

Here only the second case satisfies the definition of maximally parallel application of $\mathsf{trl}_{p,q}$ because it applies for the smallest working place.

Example 2. Let del_p be the contextual deletion operation applied in the relation of $(x_1x_2, p, x_3) \sim (x_3, p, x_1)$, 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 $x_1x_2px_3px_1x_2px_3px_1 \Longrightarrow^{max}_{\mathsf{del}_p} x_1x_2px_1x_2px_1$.

4 Computational Efficiency and Universality

Definition 9. *An extended accepting intramolecular recombination (eAIR) system is a tuple $G = (\Sigma, \sim, \widetilde{R}, \alpha_0, w_t)$ where $\widetilde{R}$ is the set of recombination operations, $\alpha_0 \in \Sigma^*$ is the start word, and $w_t \in \Sigma^+$ is the target word. If α_0 is not specified, we omit it. The operation classes of splicing scheme $R_{op} = (\Sigma, \sim, op), op \in \widetilde{R}$ are defined. Then the operations are applied in a sequentially or a maximally parallel manner according to the operation classes. The language accepted by G is defined by $L(G) = \{w \in \Sigma^* \mid \alpha_0 w \Longrightarrow^*_{R_{op\in\widetilde{R}}} w_t\}$.*

We use the extended accepting intramolecular recombination (eAIR) 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 10. *We say that a decision problem X is solved in time $O(t(n))$ by extended accepting intramolecular recombination systems if there exists a family $\mathcal{A}$ of extended AIR 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 extended accepting intramolecular recombination system $G(x) \in \mathcal{A}$ which decides, again in time $O(t(n))$, the word encoding the given instance. This means that the word is accepted if and only if the solution to the given instance of the problem is* `YES`.

Theorem 1. SAT *can be solved deterministically in linear time by an extended accepting intramolecular recombination system constructed in polynomial time in the size of the given instance of the problem.*

Proof. Let us consider a propositional formula in the conjunctive normal form, $\varphi = C_1 \wedge \cdots \wedge C_m$, such that each clause $C_i, 1 \leq i \leq m$, is of the form $C_i = y_{i,1} \vee \cdots \vee y_{i,k_i}, k_i \geq 1,$ where $y_{i,j} \in \{x_k, \bar{x}_k \mid 1 \leq k \leq n\}$.

We construct an extended accepting intramolecular recombination system

$$G = (\Sigma, \sim, \widetilde{R}, \texttt{YES}), \text{ where}$$

$$\begin{aligned}
\Sigma &= \{\$_i \mid 0 \leq i \leq m+1\} \cup \{x_i, x_{\bar{i}}, \langle_i, \langle_{\bar{i}}, \rangle_i, \rangle_{\bar{i}}, \dagger_i, \dagger_{\bar{i}} \mid 1 \leq i \leq n\} \\
&\quad \cup \{f_i \mid 0 \leq i \leq n+1\} \cup \{T, F, \vee, \texttt{Y}, \texttt{E}, \texttt{S}\}, \\
R &= (\Sigma, \sim), \\
\widetilde{R} &= \{\mathsf{trl}_{\langle_i,\rangle_i}, \mathsf{trl}_{\langle_{\bar{i}},\rangle_{\bar{i}}}, \mathsf{del}_{f_i}, \mathsf{del}_{\dagger_i}, \mathsf{del}_{\dagger_{\bar{i}}} \mid \langle_i, \rangle_i, \langle_{\bar{i}}, \rangle_{\bar{i}}, f_i, \dagger_i, \dagger_{\bar{i}} \in core(R), 1 \leq i \leq n\} \\
&\quad \cup \{\mathsf{cpy}_{f_i} \mid f_i \in core(R), 0 \leq i \leq n-1\} \cup \{\mathsf{del}_{\$_i} \mid \$_i \in core(R), 1 \leq i \leq m\} \\
&\quad \cup \{\mathsf{del}_{\texttt{E}} \mid \texttt{E} \in core(R)\},
\end{aligned}$$

and the operation classes R_{trl}, R_{del}, and R_{cpy} are defined below.

We encode each clause C_i as a string bounded by $\$_i$ in the following form: $c_i = \$_i \vee \langle_{\sigma(j_b)} x_{\sigma(j_b)} \rangle_{\sigma(j_b)} \vee \cdots \vee \langle_{n_{\sigma(j_b)}} x_{n_{\sigma(j_b)}} \rangle_{n_{\sigma(j_b)}} \vee \$_i$, where $b \in \{0,1\}, \sigma(j_1) = j, \sigma(j_0) = \bar{j}$, and x_j stands for variable x_j, while $x_{\bar{j}}$ stands for negated variable $\bar{x}_j$, $1 \leq j \leq n$, in the formula φ. The instance φ is encoded as:

$$\delta = \$_0 c_1 \ldots c_m \$_{m+1}.$$

In order to generate all possible truth-assignments for all variables $x_1, x_2, \ldots, x_n$ of the formula, we consider a string of the form

$$\gamma = \dagger_1 \langle_1 T \rangle_1 \dagger_1 \dagger_{\bar{1}} \langle_{\bar{1}} F \rangle_{\bar{1}} \dagger_{\bar{1}} \ldots \dagger_n \langle_n T \rangle_n \dagger_n \dagger_{\bar{n}} \langle_{\bar{n}} F \rangle_{\bar{n}} \dagger_{\bar{n}}.$$

Here T and F denote the truth-values $true$ and $false$, respectively. Then m copies of γ are attached to the end of the encoded formula, leading to $\beta = \delta\gamma^m$.

If we have a propositional formula $\varphi' = (x_1 \vee \bar{x}_2) \wedge (x_1 \vee x_2)$, then

$$\begin{aligned}
\delta' &= \$_0\$_1 \vee \langle_1 x_1 \rangle_1 \vee \langle_{\bar{2}} x_{\bar{2}} \rangle_{\bar{2}} \vee \$_1\$_2 \vee \langle_1 x_1 \rangle_1 \vee \langle_2 x_2 \rangle_2 \vee \$_2\$_3, \text{ and} \\
\gamma' &= \dagger_1 \langle_1 T \rangle_1 \dagger_1 \dagger_{\bar{1}} \langle_{\bar{1}} F \rangle_{\bar{1}} \dagger_{\bar{1}} \dagger_2 \langle_2 T \rangle_2 \dagger_2 \dagger_{\bar{2}} \langle_{\bar{2}} F \rangle_{\bar{2}} \dagger_{\bar{2}}.
\end{aligned}$$

We use the following notations:

$$\begin{aligned}
\gamma = \gamma_1\gamma_2\ldots\gamma_n \text{ where } \gamma_i &= \gamma_i^{(1)}\gamma_i^{(0)}, \gamma_i^{(1)} = \dagger_i\langle_i T\rangle_i\dagger_i, \gamma_i^{(0)} = \dagger_{\bar{i}}\langle_{\bar{i}} F\rangle_{\bar{i}}\dagger_{\bar{i}}. \\
\gamma_{i,n} = \gamma_i\gamma_{i+1}\ldots\gamma_n.\gamma_{i,j}^{(b_{i,j})} &= \gamma_i^{(b_i)}\gamma_{i+1}^{(b_{i+1})}\ldots\gamma_j^{(b_j)}, b_{i,j} = b_ib_{i+1}\ldots b_j \in \{0,1\}^+, \\
1 \leq i < j \leq n, \gamma_{1,1} &= \gamma_1, \gamma_{n,n} = \gamma_n, \gamma_{1,0} = \lambda, \gamma_{n+1,n} = \lambda, \Gamma \in Sub(\gamma), \\
\alpha_{i,n} = f_if_{i+1}\ldots f_n, \bar{\alpha}_{i,n} &= f_nf_{n-1}\ldots f_i, 0 \leq i \leq n, \\
\alpha_{n+1,n+1} = \bar{\alpha}_{n+1,n+1} = f_{n+1}, \sigma(i_1) = i,\ \sigma(i_0) &= \bar{i}, B \in \{T, F\}.
\end{aligned}$$

The input string π_1 (we call it in-string) containing the encoded propositional formula φ is of the form:

$$\pi_1 = \mathtt{YE}\bar{\alpha}_{1,n+1}f_0\alpha_{1,n+1}\beta\bar{\alpha}_{1,n+1}f_0\alpha_{1,n+1}\mathtt{ES}.$$

The size of the input is quadratic, $|\pi_1| \leq 4nm+14n+3m+12$. Hence the system is constructible by a deterministic Turing machine in time $O(nm)$ in the size of the given instance of the problem.

Roughly speaking, the main idea of the algorithm by which the eAIR system solves `SAT` is as follows: (i) We generate all possible truth-assignments on the in-string according to the variables of the given instance of the propositional formula, while the encoded propositional formula with the attached Γ is copied, in linear time. (ii) Then each truth-assignment is assigned to its attached formula in the maximally parallel way. Step (iii), we check the satisfiability of the formula with regard to the truth-assignments. (iv) Finally, eAIR system decides to accept or not the input string π_1. We stress here that the recombination steps are guided very much by the classes of splicing schemes and by the contexts of rules defined in the splicing relations. The phases (i), (ii), (iii), and (iv) of the computation are illustrated in Fig. 3.

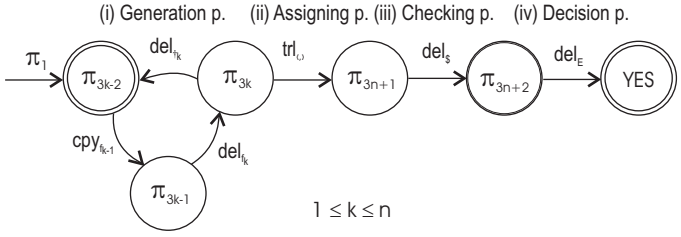

Fig. 3. A scheme of the algorithm.

Let us start the computation accomplishing the above steps.

(i) Generating truth-value assignments. The generation of all possible truth-assignments takes $3n$ steps. The computation starts applying operation cpy_{f_0} (it is easy to observe that no other operations are applicable on the in-string at the first step), and then $\mathsf{del}_{\dagger_1}$ and $\mathsf{del}_{\dagger_{\bar{1}}}$ are applied parallel, following this step operation del_{f_1} applies. The next combination of three modules are applied automatically one after other in a cyclic order:

$$\mathsf{cpy}_{f_i} \Longrightarrow \mathsf{del}_{\dagger_{i+1}}, \mathsf{del}_{\dagger_{\overline{i+1}}} \Longrightarrow \mathsf{del}_{f_{i+1}}, 0 \leq i \leq n-1.$$

We emphasize here that the repeated pointers used in cpy_{f_i} are the closest to each other, because of the maximal parallel application of the rule and the definition of the operation cpy_p.

(1) The copy operation cpy_{f_i} duplicates the substrings s_i flanked by the repeats of f_i.

$$\alpha f_i s_i f_i \alpha' \Longrightarrow^{max}_{\mathsf{cpy}_{f_i}} \alpha f_i s_i s_i f_i \alpha',\ \alpha, \alpha' \in \Sigma^*, 0 \leq i \leq n-1.$$

The contexts for the cpy_{f_i} operation to be applied are

– $(\bar{\alpha}_{i+1,n+1}, f_i, s_i) \sim (s_i, f_i, \alpha_{i+1,n+1}) \in R_{\mathsf{cpy}}$,
where $s_i = \alpha_{i+1,n+1}\delta(\gamma_{1,i}^{(b_{1,i})}\gamma_{i+1,n})^m\bar{\alpha}_{i+1,n+1}$.

At each $3k-2$th ($1 \leq k \leq n$) step, the in-string contains substrings of the form:

$$f_{n+1}f_n \dots f_{i+1}\underline{f_i}f_{i+1}\dots f_n f_{n+1}\delta(\gamma_{1,i}^{(b_{1,i})}\gamma_{i+1,n})^m f_{n+1}f_n\dots f_{i+1}\underline{f_i}f_{i+1}\dots f_n f_{n+1}.$$

Each substring s_i flanked by repeated f_i contains only one $\delta\Gamma^m$ part. The only possible rule to be applied in the context of this substring is cpy_{f_i} for the repeats $f_i, i = 3k-3$ (at this step there no rule is applicable from each R_{trl} and R_{del}, which can be checked out later on). No copy using $\mathsf{cpy}_{f_j}, j \neq 3k-3, j \geq 0$, may occur, because no $f_j, j < 3k-3$, has remained in the string; on the other hand, at this moment there is no pattern of the form $f_{j+1}f_jf_{j+1}$, $j > 3k-3$, to which cpy_{f_j} is applicable.

The copy cpy_{f_i} is applied in a maximally parallel way on the string for its smallest working places. As the result of this step, each substring flanked by f_i is copied as the following form:

$$f_{i+1}\underline{f_i}f_{i+1}\dots f_{n+1}\delta(\gamma_{1,i}^{(b_{1,i})}\gamma_{i+1,n})^m f_{n+1}\dots f_{i+1}$$
$$f_{i+1}\dots f_{n+1}\delta(\gamma_{1,i}^{(b_{1,i})}\gamma_{i+1,n})^m f_{n+1}\dots f_{i+1}\underline{f_i}f_{i+1}.$$

The substring flanked by f_i-s satisfies as the smallest working place for cpy_{f_i}, but cpy_{f_i} still can not apply to it because, on the one hand, by the definition of cpy_{f_i}, it applies to a substring s_i flanked by f_i-s and on the other hand, the substring s_i has to contain only one $\delta\Gamma^m$ part as the relations in R_{cpy} constrain it. But it is not the case with the above smallest working place. There are two $\delta\Gamma^m$ parts between the repeats of f_i. There are splicing relations for the repeats of f_i in R_{del} but we will check later that which are not satisfied on the current string.

(2) Thus, the next deletion operations $\mathsf{del}_{\dagger_i}$ and $\mathsf{del}_{\dagger_{\bar{i}}}$ apply to the current in-string. It is the $3k-1$th ($1 \leq k \leq n$) step. The deletion $\mathsf{del}_{\dagger}$ is applied maximally parallel on the 2^k copies of Γ^m. By deletion $\mathsf{del}_{\dagger_i}$ (resp. $\mathsf{del}_{\dagger_{\bar{i}}}$), the truth-values $\langle_i T\rangle_i$ (resp. $\langle_{\bar{i}} F\rangle_{\bar{i}}$) are deleted wherever the contexts of $\mathsf{del}_{\dagger}$ are satisfied.

$$\alpha\dagger_{\sigma(i_b)}\langle_{\sigma(i_b)}B\rangle_{\sigma(i_b)}\dagger_{\sigma(i_b)}\alpha' \Longrightarrow^{max}_{\mathsf{del}_{\dagger_{\sigma(i_b)}}} \alpha\dagger_{\sigma(i_b)}\alpha', \alpha, \alpha' \in \Sigma^*, 1 \leq i \leq n.$$

$\mathsf{del}_{\dagger_{\bar{i}}}$: $(f_{i-1}\alpha_{i,n+1}\delta(\gamma_{1,i-1}^{(b_{1,i-1})}\gamma_{i,n})^{j-1}\gamma_{1,i-1}^{(b_{1,i-1})}\gamma_i^1, \dagger_{\bar{i}}, \langle_{\bar{i}} F\rangle_{\bar{i}}) \sim$
$(\langle_{\bar{i}} F\rangle_{\bar{i}}, \dagger_{\bar{i}}, \gamma_{i+1,n}(\gamma_{1,i-1}^{(b_{1,i-1})}\gamma_{i,n})^{m-j}\bar{\alpha}_{i,n+1}\alpha_{i,n+1}) \in R_{\mathsf{del}}$ where $b_i \in \{0,1\}, 1 \leq j \leq m, 1 \leq i \leq n, \gamma_{1,1} = \gamma_1, \gamma_{n,n} = \gamma_n, \gamma_{1,0} = \gamma_{n+1,n} = \lambda$.

The context says that the substring $\dagger_{\bar{i}}\langle_{\bar{i}}F\rangle_{\bar{i}}$, which is going to be deleted, has to be preceded by a substring which is bordered by f_{i-1} in the left side, and followed by a substring of the form $f_{n+1}\dots f_i f_i \dots f_{n+1}$. One more crucial constraint is that all $\gamma_j, j < i$, have to be broken already and $\gamma_j, j > i$, have not been processed up to now. $\mathsf{del}_{\dagger}$ applies to m copies of Γ attached to δ, namely, it applies to all Γ in the maximally parallel way on the in-string which satisfies the contexts.

$\mathsf{del}_{\dagger_i}$: $(\bar{\alpha}_{i,n+1}\alpha_{i,n+1}\delta(\gamma_{1,i-1}^{(b_{1,i-1})}\gamma_{i,n})^{j-1}\gamma_{1,i-1}^{(b_{1,i-1})}, \dagger_i, \langle_i T\rangle_i) \sim$
$(\langle_i T\rangle_i, \dagger_i, \gamma_i^0\gamma_{i+1,n}(\gamma_{1,i-1}^{(b_{1,i-1})}\gamma_{i,n})^{m-j}\bar{\alpha}_{i,n+1}f_{i-1}) \in R_{\mathsf{del}}$,
where $b_i \in \{0,1\}, 1 \le j \le m, 1 \le i \le n, \gamma_{1,1} = \gamma_1$,
$\gamma_{n,n} = \gamma_n, \gamma_{1,0} = \lambda, \gamma_{n+1,n} = \lambda$.
A substring $\dagger_i\langle_i T\rangle_i$ with its preceding substring bordered by f_{i-1} in the right side is deleted by $\mathsf{del}_{\dagger_i}$.

Note that since there is no splicing relation for the repeats of † in the classes of the splicing schemes except R_{del}, only $\mathsf{del}_{\dagger}$ applies to the repeats of †. Up to now, 2^i truth-assignments of the form $\gamma_1^{(b_1)}\gamma_2^{(b_2)}\dots\gamma_i^{(b_i)}\gamma_{i+1}\dots\gamma_n, b_i \in \{0,1\}$ have been generated on the in-string.

(3) At the $3k-1$th ($1 \le k \le n$) step of the computation, the deletion operation del_{f_i} is allowed. It applies for the repeats of f_i and deletes one copy of f_i with one f_{i-1} if it is available between those f_i.

$$\alpha f_i f_{i-1} f_i \alpha' \Longrightarrow^{max}_{\mathsf{del}_{f_i}} \alpha f_i \alpha', \alpha, \alpha' \in \Sigma^*, 1 \le i \le n.$$

The next two contexts are in the splicing scheme for del_{f_i}:

- $(\bar{\alpha}_{i+1,n+1}, f_i, f_{i-1}) \sim (f_{i-1}, f_i, \alpha_{i+1,n+1}\delta(\gamma_{1,i}^{b_{1,i}}\gamma_{i+1,n})^m) \in R_{\mathsf{del}}$,
- $(\delta(\gamma_{1,i}^{(b_{1,i})}\gamma_{i+1,n})^m\bar{\alpha}_{i+1,n+1}, f_i, f_{i-1}) \sim (f_{i-1}, f_i, \alpha_{i+1,n+1}) \in R_{\mathsf{del}}$.

Remember that at the previous two steps del_{f_i} was not applicable because γ_i had not been operated. Since γ_i was operated at the previous step by $\mathsf{del}_{\dagger_i}$, now del_{f_i} can be applied. Here the subscripts are shifted from i to $i-1$.

$$\alpha f_i f_i \alpha' \Longrightarrow^{max}_{\mathsf{del}_{f_i}} \alpha f_i \alpha', \text{ where}$$

- $(\delta(\gamma_{1,i}^{(b_{1,i})}\gamma_{i+1,n})^m\bar{\alpha}_{i+1,n+1}, f_i, f_i) \sim (f_i, f_i, \alpha_{i+1,n+1}) \in R_{\mathsf{del}}$
  where $\alpha_{n+1,n+1} = \bar{\alpha}_{n+1,n+1} = f_{n+1}$.

After n repeats of the cyclic iteration (1) $\Longrightarrow$ (2) $\Longrightarrow$ (3), it ends at the $3n$th step and all truth-assignments have been generated completely. Now the in-string is of the form:

$$\pi_{3n} = \mathtt{YE}(f_{n+1}f_n f_{n+1}\delta\Gamma^m f_{n+1}f_n f_{n+1})^{2^n}\mathtt{ES}.$$

We enter to the next phase as follows.

(ii) Assigning the truth-values to the variables. The truth-values (truth-assignments) are assigned to each variable (to each clause) of the propositional formula by the next translocation operations:

$$\alpha'\langle_{\sigma(i_b)}x_{\sigma(i_b)}\rangle_{\sigma(i_b)}\alpha''\langle_{\sigma(i_b)}B\rangle_{\sigma(i_b)}\alpha''' \Longrightarrow^{max}_{\mathsf{trl}_{\langle_{\sigma(i_b)},\rangle_{\sigma(i_b)}}} \alpha'\langle_{\sigma(i_b)}B\rangle_{\sigma(i_b)}\alpha''\langle_{\sigma(i_b)}x_{\sigma(i_b)}\rangle_{\sigma(i_b)}\alpha''',$$

where $b \in \{0,1\}, \sigma(i_1) = i, \sigma(i_0) = \bar{i}, 1 \leq i \leq n, \alpha', \alpha'', \alpha''' \in \Sigma^*$. We have the next two splicing relations in R_{trl}.

- $(\vee, \langle_{\sigma(i_b)}, x_{\sigma(i_b)}\rangle_{\sigma(i_b)} \vee u\dagger_{\sigma(i_b)}) \sim (x_{\sigma(i_b)}\rangle_{\sigma(i_b)} \vee u\dagger_{\sigma(i_b)}, \langle_{\sigma(i_b)}, B\rangle_{\sigma(i_b)}\dagger_{\sigma(i_b)}v)$
- $(\langle_{\sigma(i_b)}x_{\sigma(i_b)}, \rangle_{\sigma(i_b)}, \vee u\dagger_{\sigma(i_b)}\langle_{\sigma(i_b)}B) \sim (\vee u\dagger_{\sigma(i_b)}\langle_{\sigma(i_b)}B, \rangle_{\sigma(i_b)}, \dagger_{\sigma(i_b)}v)$, where $u \neq u'f_{n+1}u'',\ v \neq v'f_{n+1}v'',\ u', u'', v', v'' \in \Sigma^*$.

By the application of $\mathsf{trl}_{\langle_i,\rangle_i}$, every variable x_i flanked by $\langle_i$ and $\rangle_i$ in each clause c_j and the corresponding truth-value T (*true*) flanked by $\langle_i$ and $\rangle_i$ is swapped if such correspondence exists. Similarly, for assigning the truth-value F ($false$) to $x_{\bar{i}}$, the contents of $\langle_{\bar{i}}x_{\bar{i}}\rangle_{\bar{i}}$ and $\langle_{\bar{i}}F\rangle_{\bar{i}}$ are swapped by $\mathsf{trl}_{\langle_{\bar{i}},\rangle_{\bar{i}}}$. Thus, all truth-assignments are assigned to their attached propositional formula at the same time in a single step. The truth-values assigned to a fixed formula should not be mixed from the different assignments. That is why the constraints $u \neq u'f_{n+1}u'',\ v \neq v'f_{n+1}v''$ are required in the context. It is the $3n+1$th step.

(iii) Checking the satisfiability of the propositional formula. If a clause c_k is satisfied by a truth-assignment, then at least one substring of type $\langle_{\sigma(i_b)}B\rangle_{\sigma(i_b)}$ exists in c_k as $\$_k x_k\langle_{\sigma(i_b)}B\rangle_{\sigma(i_b)}y_k\$_k$. However, if the propositional formula φ is satisfied by a truth-assignment, then each clause c_l of φ is of the form $\$_l x_l\langle_{\sigma(i_b)}B\rangle_{\sigma(i_b)}y_l\$_l, x_l, y_l \in \Sigma^*, 1 \leq l \leq m$. Then an encoding of the propositional formula satisfied by an assignment is of the form:

$\$_0\$_1x_1\langle_{\sigma(i_b)}B\rangle_{\sigma(i_b)}y_1\$_1\$_2x_2\langle_{\sigma(i_b)}B\rangle_{\sigma(i_b)}y_2\$_2\ldots\$_mx_m\langle_{\sigma(i_b)}B\rangle_{\sigma(i_b)}y_m\$_m\$_{m+1}$, where $\sigma(i_1) = i, \sigma(i_0) = \bar{i}, 1 \leq i \leq n$, $x_j, y_j \in \Sigma^*, 1 \leq j \leq m$.

The deletion operation $\mathsf{del}_{\$_i}$ applies to the in-string in maximally parallel, and deletes the clauses which contain at least a truth-value of the form $\$_i\alpha'\langle_{\sigma(j_b)}B\rangle_{\sigma(j_b)}\alpha''\$_i$ as follows:

$$\alpha\$_i\alpha'\langle_{\sigma(j_b)}B\rangle_{\sigma(j_b)}\alpha''\$_i\alpha''' \Longrightarrow^{max}_{\mathsf{del}_{\$_i}} \alpha\$_i\alpha''',$$

- $(\$_{i-1}, \$_i, x_i\langle_{\sigma(j_b)}B\rangle_{\sigma(j_b)}y_i) \sim (x_i\langle_{\sigma(j_b)}B\rangle_{\sigma(j_b)}y_i, \$_i, \$_{i+1}) \in R_{\mathsf{del}}$, $x_i, y_i \in \Sigma^*, 1 \leq i \leq m, 1 \leq j \leq n$.

The checking phase has been done at the $3n+2$th step of the computation.

(iv) Deciding. At the end of the computation, we obtain some sequences of the form $\$_0\$_1\$_2\ldots\$_m\$_{m+1}$ on the in-string if there exist truth-assignments which satisfy the formula φ. Then $\mathsf{del}_{\mathtt{E}}$ applies to the in-string $\mathtt{YE}u\$_1\$_2\ldots\$_{m-1}\$_mv\mathtt{ES}$ for the repeats of $\mathtt{E}$.

$$\mathtt{YE}u\$_1\$_2\ldots\$_{m-1}\$_mv\mathtt{ES} \Longrightarrow_{\mathsf{del}_{\mathtt{E}}} \mathtt{YES}, \text{ where}$$

– $(\mathtt{Y}, \mathtt{E}, \$) \sim (\$, \mathtt{E}, \mathtt{S}) \in R_{\mathsf{del}}$, $\$ = u\$_1\$_2 \ldots \$_{m-1}\$_m v$, for some $u, v \in \Sigma^*$.

Thus, at $3n + 3$th step, the target string YES is reached.

If no sequence $\$_1\$_2 \ldots \$_m$ is obtained with the in-string, then the computation just halts at the $3n + 2$th step since no rule is possible to apply from now on, hence, π_0 is not accepted by G. Thus, the problem φ is solved in a linear time. □

We recall here the following result of [6].

Theorem 2. *[6] For any deterministic Turing machine $M = (S, \Sigma \cup \{\#\}, P)$ there exists an intramolecular recombination system $G_M = (\Sigma', \sim, \alpha_0, w_t)$ and a string $\pi_M \in \Sigma'^*$ such that for any word w over Σ^* there exists $k_w \geq 1$ such that $w \in L(M)$ if and only if $w\#^5\pi_M^{k_w}\#^2 \in L(G_M)$.*

The next theorem proves that eAIR system is computationally universal.

Theorem 3. *For any deterministic Turing machine $M = (S, \Sigma \cup \{\#\}, P)$, there exists an extended intramolecular recombination system $G_M = (\Sigma', \sim, \widetilde{R}, \alpha_0, w_t)$, and a string $\pi_M \in \Sigma'^*$ for any word $w \in L(M)$ iff $w\#^3\pi_M\#^2 \in L(G_M)$.*

Proof. Since the string can be extended by cpy during the recombination processes in eAIR, with this proof we do not need to reserve as many copies of an encoded TM rule as we did in Theorem 2. Instead, an encoding of a rewriting rule is duplicated by cpy before it is used. Then one copy of the encoding is used by a translocation operation while another copy is reserved as initial state. In a manner similar to Theorem 2, for a Turing machine M we construct an extended intramolecular recombination system

$$
\begin{aligned}
G_M &= (\Sigma', \sim, \widetilde{R}, \alpha_0, w_t), \text{ where} \\
\Sigma' &= S \cup \Sigma \cup \{\#\} \cup \{\$_i \mid 0 \leq i \leq m+1\}, \\
\alpha_0 &= \#^3 s_0, w_t = \#^3 s_f \#^3, \\
\widetilde{R} &= \{\mathsf{trl}_{p,q}, \mathsf{del}_p, \mathsf{cpy}_p \mid p, q \in \mathit{core}(R)\}.
\end{aligned}
$$

We also consider the string

$$
\pi_M = \$_0(\prod_{\substack{1 \leq i \leq m \\ p,q \in \Sigma \cup \{\#\}}} \$_i p v_i q \$_i)\$_{m+1},\ \pi_M \in \Sigma'^*.
$$

The classes of splicing relations $R_{op \in \widetilde{R}} = (\Sigma', \sim, op)$ are constructed as follows:

$$
\begin{aligned}
R_{\mathsf{cpy}}\colon\ & (pu_iqw'\$_0w'', \$_i, pv_iq) \sim (pv_iq, \$_i, \$_{i+1}), && (1)\\
R_{\mathsf{trl}}\colon\ & (c, p, u_iqd) \sim (\$_i, p, v_iqp), && (2)\\
& (pu_i, q, d) \sim (\$_ipv_i, q, pv_iq\$_i), && (3)\\
& \text{where } c, d, p, q \in \Sigma \cup \{\#\}, \\
R_{\mathsf{del}}\colon\ & (\$_i, p, u_iq) \sim (u_iq, p, v_iq\$_i), && (4)\\
& (\#^3 s_f \#, \#, \#) \sim (\$_{m+1}, \#, \#). && (5)
\end{aligned}
$$

If a word $w \in \Sigma^*$ is accepted by Turing machine M, then associated eAIR G_M works as follows:

$$\#^3 s_0 w \#^3 \pi_M \#^2 \Longrightarrow^*_{R_{op \in \tilde{R}}} \#^3 s_f \#^3 \pi_M \#^2 \Longrightarrow_{\mathsf{del}_{\#}} \#^3 s_f \#^3.$$

Where we refer to the subsequence $\#^3 s_0 w \#^3$ as the "data", and to the subsequence $\pi_M \#^2$ as the "program". A rewriting rule $i : u \to v \in P$ of M is simulated in three subsequent steps ($\mathsf{cpy} \Longrightarrow \mathsf{trl} \Longrightarrow \mathsf{del}$) in G_M.

Step 1. When left-hand side of a rule $i : u_i \to v_i$ appears in the form pu_iq in data, the corresponding encoding of the right-hand side of the rule pv_iq flanked by $\$_i$-s in the program is copied ($\$_i pv_i qpv_i q\$_i$). The copy $\mathsf{cpy}_{\$_i}$ performs in the contexts defined by relation (1) as follows,

$$xpu_iqw'\$_0 w''\$_i pv_i q\$_i\$_{i+1} y \Longrightarrow_{\mathsf{cpy}_{\$_i}} xpu_iqw'\$_0 w''\$_i pv_i qpv_i q\$_i\$_{i+1} y.$$

Step 2. The u_i and the corresponding v_i each one is flanked by p and q are swapped by $\mathsf{trl}_{p,q}$ provided the relations (2) and (3) are satisfied. The translocation $\mathsf{trl}_{p,q}$ performs as follows,

$$xcpu_iqdw'\$_i pv_i qpv_i q\$_i\$_{i+1} y \Longrightarrow_{\mathsf{trl}_{p,q}} xcpv_iqdw'\$_i pu_i qpv_i q\$_i\$_{i+1} y.$$

Step 3. The substring pu_iq used in the swapping is deleted from the program in the context of relation (4) as follows,

$$xcpu_jqdw'\$_i pu_i qpv_i q\$_i\$_{i+1} y \Longrightarrow_{\mathsf{del}_p} xcpu_jqdw'\$_i pv_i q\$_i\$_{i+1} y.$$

The encoding of $\$_i pv_i q\$_i$ is still kept in program if u_i appears again in the data.

Thus, the rewriting rule $i : u_i \to v_i$ is simulated in three subsequent *Steps 1–3* in G_M. At step 3, while del_p performs, the next rewriting rule $j : u_j \to v_j$, $j \neq i$ simulation could start making a copy of the corresponding $\$_j pv_j q\$_j$ in program, unless u_i appears immediately in a subsequent step in data. If the last case happens, its simulation starts at the next step following del_p.

The rewriting rules of M are simulated in G_M correctly by the subsequent repeats of the recombination *Steps 1–3*, and the next string can be reached:

$$\#^3 s_0 w \#^3 \pi_M \#^2 \Longrightarrow^*_{R_{op \in R}} \#^3 s_f \#^3 \pi_M \#^2.$$

When the substring $\#^3 s_f \#^3$ contains the final state s_f is obtained in data, the contexts of the relation (5) are satisfied the operation $\mathsf{del}_{\#}$ to be applied:

$$\#^3 s_f \#^3 \pi_M \#^2 \Longrightarrow_{\mathsf{del}_{\#}} \#\#\# s_f \#\#\#.$$

Thus, the target $\#\#\# s_f \#\#\#$ is reached, G_M accepts the word $\#^3 s_0 w \#^3 \pi_M \#^2$.

For the converse implication, we have to claim that only a correct relation or a correct combination of the relations and nothing else is used in each recombination step. We start the claim with the relation (1): operation $\mathsf{cpy}_{\$_i}$ does not repeat immediately for the repeats of the pointer $\$_i$ following its application.

By the definition of cpy_p, the substring flanked by the repeat of a pointer p is copied if and only if it is the following context of the first occurrence of p, and the preceding context of the second occurrence of p in the splicing relation (see Definition 3). Since the substring $\$_i pv_i qpv_i q\$_i$ does not satisfy the context defined by relation (1), $\mathsf{cpy}_{\$_i}$ is not applicable to it. It is true that only relations (2) and (3) from R_{trl} among others are satisfied on the string at the next step of $\mathsf{cpy}_{\$_i}$ applied. When $\mathsf{trl}_{p,q}$ was applied at *Step 2* swapping u_i and v_i in the contexts (2) and (3), it cannot be applied immediately its following step to the same pointers in the same contexts as previous one. For instance, encoding of u_i would appear again in the data by $\mathsf{trl}_{p,q}$ as $xpu_i qw\$_i pu_i qpv_i q\$_i$, but $\mathsf{trl}_{p,q}$ is not applicable to p and q here. Because the preceding and following contexts of the second occurrence of the pointer q of the splicing relation (3) cannot be satisfied on the string since u_i and v_i can never be the same strings in the rules of Turing machine M. It completes the proof. □

5 Final Remarks

In the present paper we propose a computing model which is based on the ciliate intramolecular gene assembly model developed, for instance, in [1]. The model is mathematically elegant and biologically well-motivated because only three types of operations (two of them are formalizations of gene assembly process in ciliates) and a single string are involved. The context-sensitivity for string operations are already well-known in formal language theory, see [9, 7]. Moreover, parallelism is a feature characteristic of bio-inspired computing models, starting with DNA computing, which is also the case with our model. From a computer science point of view, in the model (eAIR system) is both as powerful as Turing machines and as efficient in solving intractable problems in feasible time. Investigating the other computability characteristics of eAIR system could be worthwhile.

It is important to note that the cpy operation we introduce in this model is purely theoretical. Although duplication mechanisms exists in nature, implementing a copy operation of the kind we consider has not been demonstrated yet.

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 I.P. is supported by Academy of Finland, project 108421.

We are grateful to Gheorghe Păun for useful discussions.

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. Galiukschov, B.S., Semicontextual grammars, *Mathematika Logica i Matematika Linguistika*, Talinin University (1981), 38–50 (in Russian).

3. Garey, M., Jonhson, D., *Computers and Interactability. A Guide to the Theory of NP-completeness*, Freeman, San Francisco, CA, 1979.
4. Harju, T., Petre, I., Li, C. and Rozenberg, G., Parallelism in gene assembly. In: *Proceedings of DNA-based computers 10*, Springer, to appear, 2005.
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, Petre, I., and Rogojin, V., Computational Power of Intramolecular Gene Assembly, to appear in *International Journal of Foundations of Computer Science*.
7. Kari, L., and Thierrin, G., Contextual insertion/deletions and computability. *Information and Computation* 131 (1996) pp. 47–61.
8. 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 (1998) pp. 3–15.
9. Marcus, S., Contextual grammars, *Revue Roumaine de Matématique Pures et Appliquées*, 14 (1969), pp. 1525–1534.
10. Papadimitriou, Ch. P., *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.
11. Păun, Gh., *Marcus Contextual Grammars*, Kluwer, Dordrecht, (1997).
12. Păun, Gh., Rozenberg, G., Salomaa, A., *DNA Computing - New computing paradigms*, Springer-Verlag, Berlin, (1998).
13. Salomaa, A., *Formal Languages*, Academic Press, New York (1973).
14. Searls, D. B., Formal language theory and biological macromolecules. *Series in Discrete Mathematics and Theoretical Computer Science* 47 (1999), pp. 117–140.