TUCS

Tseren-Onolt Ishdorj | Vladimir Rogojin | Ion Petre

Computational Power of Intramolecular Gene Assembly

Turku Centre for Computer Science

TUCS Technical Report
No 815, March 2007

Computational Power of Intramolecular Gene Assembly

Tseren-Onolt Ishdorj
Åbo Akademi University, Department of Information Technologies
Lemminkäisenkatu 14 A, 20520 Turku, Finland
tishdorj@abo.fi
Sevilla University, Department of Computer Science and Artificial Intelligence
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

Vladimir Rogojin
Åbo Akademi University, Department of Information Technologies
Turku Centre for Computer Science
Lemminkäisenkatu 14 A, 20520 Turku, Finland
vrogojin@abo.fi

Ion Petre
Åbo Akademi University, Department of Information Technologies
Turku Centre for Computer Science
Lemminkäisenkatu 14 A, 20520 Turku, Finland
ipetre@abo.fi
Academy of Finland

TUCS Technical Report
No 815, March 2007

Abstract

The process of gene assembly in ciliates, an ancient group of organisms, is one of the most complex instances of DNA manipulation known in any organism. Three molecular operations (ld, hi, and dlad) have been postulated for the gene assembly process, [3], [1]. We propose in this paper a mathematical model for contextual variants of ld and dlad on strings: recombinations can be done only if certain contexts are present. We prove that the proposed model is Turing-universal.

Keywords: Turing universality, gene assembly in ciliates

TUCS Laboratory
Computational Biomodelling Laboratory

1 Introduction

Ciliates are an ancient group of eukariotes (about 2.5 billion years old). They are known to be the most complex unicellular organisms on the Earth. Their main special feature which differs them from other eukariotes is nuclear duality: ciliates have two types of nuclei (micronucleus and macronucleus) performing completely different functions. Micronuclei are used mainly to store genetical information for future generations, while macronuclei contain genes used to produce proteins during the life-time of a cell. Genomes are stored in these two types of nuclei in two completely different ways: micronuclear genes are highly fragmented and shuffled, fragments (coding blocks) are separated from each other by non-coding blocks, while in macronuclei each DNA-molecule contains usually one gene stored in assembled (non-fragmented) way. During sexual reproduction coding blocks from micronuclei get assembled into macronuclear genes. For details related to ciliates and the gene assembly process we refer to [6], [14], [15].

Two models were proposed for the gene assembly process in ciliates: the intermolecular model in [7], [9], [10] and the intramolecular model in [3] and [16]. 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 coding block E starts with a pointer-sequence repeating exactly the pointer-sequence in the end of that coding block preceding E in the assembled gene. It is currently believed that the pointers guide the alignment of coding blocks during the gene assembly process.

The bulk of the research on the intermolecular model concentrates on the computational power of the model, in various formulations. E.g., in [7], the so-called guided recombination systems were introduced, defining a context-based applicability of the model. The authors proved that this intermolecular guided recombination system with *insertion/deletion* operations is computationally universal. For this, they constructed for each Turing machine a guided recombination system, so as for each computation of the Turing machine, there is a corresponding sequence of recombinations in the guided recombination system. Crucially, the input of the recombination system has to be given in a large enough number of copies.

Most of the research on the intramolecular model concentrates on the combinational properties of the gene assembly process, including the number and the type of operations used in the assembly, parallelism, or invariants.

In this paper we initiate a study of the intramolecular model from the perspective of the computability theory. Using a similar approach as in [7], we introduce a context-based version of the intramolecular model and prove that it is Turing universal. We prove that any Turing machine may be simulated through intramolecular recombination systems: for any Turing machine M there exists a recombination system G such that for any word w, w is accepted by M, if and only if $\varphi(w)$ is accepted by G, for a suitable encoding φ. Unlike in the intramolecular case, no multiplicities are needed in this case, since the intramolecular model conjectures that all useful (genetic) information is preserved on a single molecule throughout the assembly.

2 Preliminaries

We assume the reader to be familiar with the basic elements of formal languages and Turing computability [17], DNA computing, [13]. 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. Given two words x and y, the concatenation of x and y (denoted as xy) is defined as the word z consisting of all symbols of x followed by all symbols of y, thus $|z| = |x| + |y|$. The concatenation of a word x with itself k times is denoted as x^k, and $x^0 = \lambda$.

We denote by $|x|_S$ the number of letters from the subset $S \subseteq \Sigma$ occurring in the word x and by $|x|_a$ the number of letters a in x.

If $w = xy$, for some $x, y \in \Sigma^*$, then x is called a *prefix* of w and y is called a *suffix* of w; if $w = xyz$ for some $x, y, z \in \Sigma^*$, then y is called a *substring* of w.

A rewriting system $M = (S, \Sigma \cup \{\#\}, P)$ is called a *Turing machine* (we use also abbreviation TM), [17], 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) $s_i ac \longrightarrow a s_j c$

(3) $s_i a\# \longrightarrow a s_j \sqcup \#$

(4) $c s_i a \longrightarrow s_j ca$

(5) $\# s_i a \longrightarrow \# s_j \sqcup a$

(6) $s_f a \longrightarrow s_f$

(7) $a s_f \longrightarrow s_f$

where s_i and s_j are states in S, $s_i \neq s_f$, and a, b, c are in Σ.

A Turing machine M is called *deterministic* if:

- each word $s_i a$ from the left side of the rule (1) is not a subword of the left sides from rules (2)–(5), and

- each subword $s_i a$ from the left side of rules (2) and (3) is not subword from the left side of rules (4) and (5), and viceversa, each subword $s_i a$ from the left side of rules (4) and (5) is not subword of the left side of rules (2) and (3), and

- for each left side u_i of the rules (1)–(5) it corresponds exactly one right side v_i.

A configuration of the Turing machine M is presented as a word $\#w_1 s_i w_2\#$ over $\Sigma \cup \{\#\} \cup S$, where $w_1 w_2 \in \Sigma^*$ represents the contents of the tape, $\#$-s are the boundary markers, and the position of the state symbol s_i indicates the position of the read/write head on the tape: if s_i is positioned at the left of a letter a, this indicates that the read/write head is placed over the cell containing a. 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.

Using an approach developed in a series of works (see [11], [12], [4], and [8]) we use *contexts* to restrict the application of molecular recombination operations, [13], [1].

First, we give the formal definition of splicing rules. Consider an alphabet Σ and two special symbols, $\#, \$$, not in Σ. 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 $u_1 u_2 = \lambda$ or $u_3 u_4 = \lambda$ 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 $x = x_1 u_1 u_2 x_2$, $y = y_1 u_3 u_4 y_2$, $z = x_1 u_1 u_4 y_2$, for some $x_1, x_2, y_1, y_2 \in \Sigma^*$. We say that we *splice* x, y at the *sites* $u_1 u_2$, $u_3 u_4$, 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 $(\alpha', p, \beta') \sim (\alpha, p, \beta)$.

2.1 Intramolecular Gene Assembly Operations

The intramolecular operations excise non-coding blocks from the micronuclear DNA-molecule, interchange positions of some portions of the molecule or invert them, so as to obtain after some rearrangements the DNA-molecule containing a continuous succession of coding blocks, i.e., the assembled gene. Contrary to the intermolecular model, all the molecular operations in the intramolecular model are performed within a single molecule.

We recall bellow the three intramolecular operations conjectured in [3] and [16] for the gene assembly, which were proved to be complete [2], i.e., any sequence of coding and non-coding blocks can be assembled to the macronuclear gene by means of these operations (for details related to the intramolecular model we refer to [1]):

- ld excises a non-coding block flanked by the two occurrences of a same pointer in the form of a circular molecule, as shown in Figure 1.
- hi inverts part of the molecule flanked by the two occurrences of a same pointer, where one pointer is the inversion of the other, as shown in Figure 2.
- dlad swaps two parts of the molecule delimited by the same pair of pointers, as shown in Figure 3.

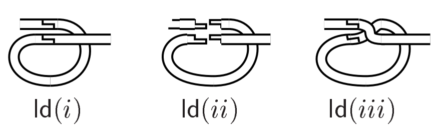

Figure 1: Loop Recombination: (i) the molecule folds on itself aligning pointers in the direct repeat to form the loop, (ii) enzymes cut on the pointer sites, (iii) hybridization happens. As the result, a portion of the molecule in the loop is excised in the form of a circular molecule.

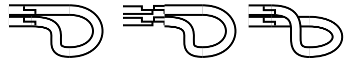

Figure 2: Hairpin Recombination: (i) the molecule folds on itself aligning pointers in the inverted repeat to form the hairpin, (ii) enzymes cut on the pointer sites, (iii) hybridization happens. As the result, a portion of the molecule in the hairpin is inverted.

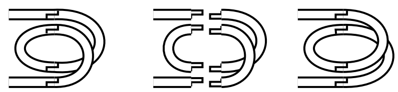

Figure 3: Double-Loop Recombination: (i) the molecule folds on itself aligning equal pointers from the repeated pair to form a double loop, (ii) enzymes cut on the pointer sites, (iii) hybridization happens. As the result, portions of the molecule in the loops interchange their places.

3 The Contextual Intramolecular Operations

We define the contextual intramolecular *translocation* and *deletion* operations as the generalization of dlad and ld operations, respectively. We follow here the style of contextual intermolecular recombination operations used in [7].

We consider a splicing scheme $R = (\Sigma, \sim)$.

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 $u \Rightarrow_{\mathsf{trl}\, p,q} v$ and say that u is recombined to v by $\mathsf{trl}_{p,q}$ operation.

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* $y = \beta' y'$*.*

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 the non-empty word u we get by del_p word v, we write $u \Rightarrow_{\mathsf{del}_p} v$ and say that the word u is recombined to v by del_p operation.

We define the set of all contextual intramolecular operations under the guiding of $\sim$ as follows:

$$\begin{aligned}\widetilde{R} \;\; = \;\; & \{\mathsf{trl}_{p,q}, \mathsf{del}_p \mid (\alpha, p, \beta) \sim (\alpha', p, \beta'), (\gamma, q, \delta) \sim (\gamma', q, \delta') \\ & \text{for some } \alpha, \alpha', \beta, \beta', \gamma, \gamma', \delta, \delta', p, q \in \Sigma^+\}.\end{aligned}$$

Now, we define an intramolecular recombination (AIR) system as the *language accepting device* that captures series of dispersed homologous recombination events on a single micronuclear molecule with a scrambled gene.

Definition 3 *An accepting intramolecular recombination system is a quadruple $G = (\Sigma, \sim, \alpha_0, w_t)$, where $R = (\Sigma, \sim)$ is the splicing scheme, $\alpha_0 \in \Sigma^*$ is the start word, and $w_t \in \Sigma^+$ is the target word.*

The language accepted by G is defined as $L(G) = \{w \in \Sigma^* \mid \alpha_0 w \Rightarrow^*_{\widetilde{R}} w_t\}$.

To illustrate the definitions above we give the following examples.

Here we show some examples of application of contexts and of the recombination operations trl and del.

(i) Consider the word $w_1 = abccbccba$ and the context $(a, b, c) \sim (c, b, a)$. The context is applicable to w_1 only in the following way: $\underline{a}\widehat{b}\underline{c}cbc\underline{c}\widehat{b}\underline{a}$, where by *underline* we marked the context and by *hat* we marked the pointers. Deletion del_b is applicable to w_1 in the context from above, i.e., $\underline{a}\widehat{b}\underline{c}cbc\underline{c}\widehat{b}\underline{a} \Rightarrow_{\mathsf{del}_b} aba$.

(ii) Consider the word $w_2 = abcabccba$ and the context $(a, b, c) \sim (c, b, a)$. This context we can apply to w_2 in two different ways: either $\underline{a}\widehat{b}\underline{c}abc\underline{c}\widehat{b}\underline{a}$, or $abc\underline{a}\widehat{b}\underline{c}\,\underline{c}\widehat{b}\underline{a}$. In this way del_b being applied to w_2 produces two different results in the context $(a, b, c) \sim (c, b, a)$: $\underline{a}\widehat{b}\underline{c}abc\underline{c}\widehat{b}\underline{a} \Rightarrow_{\mathsf{del}_b} aba$ and $abc\underline{a}\widehat{b}\underline{c}\,\underline{c}\widehat{b}\underline{a} \Rightarrow_{\mathsf{del}_b} abcaba$.

(iii) Here we show that contexts and pointers can have length greater than one. Consider the string $w_3 = bababababaaaabaa$ and the context $(b, aba, ba) \sim (aaa, aba, a)$. The context is applicable to w_3 in the following two ways: either $\underline{b}\widehat{aba}\underline{ba}b\underline{aaa}\widehat{aba}\underline{a}$ or $ba\underline{b}\widehat{aba}\underline{b\underline{a}aa}\widehat{aba}\underline{a}$. In this way by applying del_{aba} to the string w_3 in the context $(b, aba, ba) \sim (aaa, aba, a)$ we get the following two results: $\underline{b}\widehat{aba}\underline{ba}b\underline{aaa}\widehat{aba}\underline{a} \Rightarrow_{\mathsf{del}_{aba}} babaa$ and $ba\underline{b}\widehat{aba}\underline{b\underline{a}aa}\widehat{aba}\underline{a} \Rightarrow_{\mathsf{del}_{aba}} bababaa$.

(iv) Consider the string $w_4 = abcaabcabcbabcaab$ and the contexts $\sigma_1 = (a, b, c) \sim (c, b, a)$ and $\sigma_2 = (bc, a, abc) \sim (c, a, a)$. Context σ_1 is applicable to w_4 in three different ways: either $\underline{a}\widehat{b}\underline{c}aabcab\underline{c}\widehat{b}\underline{a}bcaab$ or $abca\underline{a}\widehat{b}\underline{c}ab\underline{c}\widehat{b}\underline{a}bcaab$ or $abcaabc\underline{a}\widehat{b}\underline{\underline{c}}\widehat{b}\underline{a}bcaab$. Context σ_2 can be applied to w_4 only in one way $a\underline{bc}\widehat{a}\underline{abc}abcba\overline{b}\underline{c}\widehat{a}\underline{a}b$. In this way, we can apply to w_4 reduction del_b either in one of the three different ways or reduction del_a or $\mathsf{trl}_{b,a}$ reduction in the contexts both σ_1 and σ_2. One can see, that $\mathsf{trl}_{a,b}$ is applicable to w_4 in the contexts σ_1 and σ_2 only in a single way: $\underline{a}\widehat{\overline{\overline{b}}}\overbrace{\ \underline{\overline{c}}\ }\widehat{a}\overline{abc}ab\underline{c}\widehat{b}\overbrace{\underline{a}b\overline{c}}\widehat{a}\overline{a}b \Rightarrow_{\mathsf{trl}_{b,a}} a\widehat{b}\overbrace{abc}\widehat{a}abcabc\widehat{b}\overbrace{\ c\ }\widehat{a}ab$. By underline we marked the context for the pointer b and by overline we marked the context for the pointer a.

As the summary of the example from above, to the same string the same context can be applied in many different ways and as the result, different words can be obtained from the same word by applications of the same operations.

In the next example we illustrate a recombination system.

We define an intramolecular recombination system accepting words of the form $a^n b^n$, where $n \geq 2$. $G = (\{\$, \#, 0, 1\}, \sim, \$1\#0\#1, \$10\#\#)$. We will define the splicing scheme $R = (\Sigma, \sim)$ so as for each word of the form $0^n 1^n 0\#\#$ we would obtain the target $w_t = \$10\#\#$:

$\alpha_0 0^n 1^n 0\#\# = \$1\#0\#10^n 1^n 0\#\# = \$1\#0\#\widehat{1}0\widehat{0}00^{n-3}1^{n-3}1\widehat{1}1\widehat{0}\#\# \Rightarrow_{\mathsf{trl}_{1,0}}$ $\$1\#0\#1\widehat{1}0\widehat{0}00^{n-4}1^{n-4}1\widehat{1}1\widehat{0}0\#\# \Rightarrow_{\mathsf{trl}_{1,0}} \cdots\cdots \Rightarrow_{\mathsf{trl}_{1,0}} \$1\#0\#11^{k-1}\widehat{1}0\widehat{0}00^{n-k-3}$ $1^{n-k-3}1\widehat{1}1\widehat{0}0^{k-1}0\#\# \Rightarrow_{\mathsf{trl}_{1,0}} \cdots\cdots \Rightarrow_{\mathsf{trl}_{1,0}} \$\widehat{1}\#\widehat{0}\#11^{n-3}\widehat{1}0011\widehat{0}0^{n-3}0\#\# \Rightarrow_{\mathsf{trl}_{1,0}}$ $\$100110\widehat{\#}1^{n-1}\widehat{\#}0^{n-1}0\#\# \Rightarrow_{\mathsf{del}_\#} \$100110\widehat{\#}0^{n-1}\widehat{\#}\# \Rightarrow_{\mathsf{del}_\#} \$1\widehat{0}011\widehat{0}\#\# \Rightarrow_{\mathsf{del}_0}$ $\$10\#\# = w_t$.

In this way, we need the following contexts in our splicing scheme:

(a) $(\#, 1, 0) \sim (1, 1, 10)$	(f) $(1\#, 0, \#) \sim (10011, 0, 0)$
(b) $(1, 1, 0) \sim (1, 1, 10)$	(g) $(0, \#, 1) \sim (1, \#, 0)$
(c) $(10, 0, 0) \sim (1, 0, \#)$	(h) $(0, \#, 0) \sim (0, \#, \#)$
(d) $(10, 0, 0) \sim (1, 0, 0)$	(i) $(\$, 10, 0) \sim (1, 0, \#\#)$
(e) $(\$, 1, \#0) \sim (1, 1, 00110)$	

In these contexts the recombination steps from the word $\$1\#0\#10000^{n-3}1^{n-3}$ $1110\#\#$ is looking like this (for each line i the first column from the left contains notation of the word w_i, in the second column there are shown applicable contexts, the third column contains the word w_i, the forth column from the left contains contexts of the recombination operation, the fifth column contains the recombination operation of the string, context of the left pointer of the trl_0 operation in the string is marked by *underline*, context of the right pointer is marked by *overline*, context for the del_0 operation is marked by *underline*, pointers are marked by the *hat*).

w_1	(a)(c)	$\$1\#0\underline{\#}\overline{\hat{1}\underline{0}}\hat{0}\overline{0}0^{n-3}1^{n-3}\underline{1}\hat{1}\underline{\overline{1}\hat{0}}\overline{\#}\#$	(a)(c)	$\mathsf{trl}_{1,0}$
w_2	(b)(d)	$\$1\#0\#\underline{1}\overline{\hat{1}\underline{0}}\hat{0}\overline{0}0^{n-4}1^{n-4}\underline{1}\hat{1}\underline{\overline{1}\hat{0}}\overline{0}\#\#$	(b)(d)	$\mathsf{trl}_{1,0}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
w_k	(b)(d)	$\$1\#0\#11^{k-2}\underline{1}\overline{\hat{1}\underline{0}}\hat{0}\overline{0}0^{n-k-3}1^{n-k-3}$		
		$\underline{1}\hat{1}\underline{\overline{1}\hat{0}}\overline{0}0^{k-2}0\#\#$	(b)(d)	$\mathsf{trl}_{1,0}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
w_{n-2}	(e)(f)	$\underline{\$}\overline{\hat{1}}\underline{\overline{\#}\hat{0}}\overline{\#}11^{n-4}\underline{1}\overline{\hat{1}}\underline{\overline{0011}\hat{0}}\overline{0}0^{n-4}0\#\#$	(e)(f)	$\mathsf{trl}_{1,0}$
w_{n-1}	(g)	$\$10011\underline{0}\hat{\#}\underline{1}1^{n-3}\overline{1}\hat{\#}\overline{0}0^{n-1}0\#\#$	(g)	$\mathsf{del}_{\#}$
w_n	(h)	$\$10011\underline{0}\hat{\#}\underline{0}0^{n-3}\overline{0}\hat{\#}\overline{\#}$	(h)	$\mathsf{del}_{\#}$
w_{n+1}	(i)	$\underline{\$1}\hat{0}\underline{0}1\overline{1\hat{0}\#\#}$	(i)	del_0
w_{n+2}		$\$10\#\#$		

In this way, each word of the form $0^n1^n0\#\#$ is accepted by our recombination system G. Words of the form $0^n1^m0\#\#$, where $n \neq m$ are not accepted.

Indeed, assume $m < n$. To the word w_1 from the table above only the contexts (a) and (c) are applicable and so, we can use only $\mathsf{trl}_{1,0}$ operation which can produce only the single result. After application of either del_1 or del_0 to w_1 it is not possible to reach the target. Only the contexts (b) and (d) are applicable to the words w_i with $2 \leq i \leq m-2$. Operation $\mathsf{trl}_{1,0}$ applied to w_i, $2 \leq i \leq m-2$ can produce only the single result. After application of either del_1 or del_0 to those words we cannot reach the target. In this way we get the string $w_{m-2} = \$1\#0\#11^{m-4}110000^{n-m-1}11000^{m-4}0\#\#$. Only the context (d) is applicable to w_{m-2} and in this way, only del_0 is applicable, but after that we cannot reach the target. The case when $m > n$ is proved in the same way.

4 The Computational Power of Intramolecular Contextual Recombinations

Here we show, that by using intramolecular contextual operations one can express any deterministic Turing machine. We prove that for any Turing machine M over an alphabet Σ, we associate a recombination system R over an alphabet Σ'. Also, for any $w \in \Sigma^*$, we associate a word $w' \in \Sigma'^*$ such that $w \in L(M)$ iff $w' \in L(R)$. Intuitively, R simulates M in the following way: w' encodes both the word w, as well as all rules of M in a large enough number of copies. It is important to have a large number of copies because in every step of the simulation, R "consumes" one rule of M, which is then never "recovered".

Theorem 1 *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 $w\#^5\pi_M^{k_w}\#^2 \in L(G_M)$.

Proof. Consider a deterministic Turing machine $M = (S, \Sigma \cup \{\#\}, P)$ containing m rewriting rules in P. Each rule of P we identify uniquely by an integer $1 \leq i \leq m$, and a rule identified as i we represent as $i : u_i \to v_i$. The configuration of the Turing machine can be represented by the string $\#\ w_l s_q a w_r \#$, where $a \in \Sigma$, $s_q \in S$ and $w_l, w_r \in \Sigma^*$.

We define a recombination system

$$G_M = (\Sigma', \sim, \alpha_0, w_t)$$

and a string π_M for the Turing machine M in the following way:

$$\begin{aligned}
\Sigma' &= S \cup \Sigma \cup \{\#\} \cup \{\$_i \mid 0 \leq i \leq m+1\}, \\
\alpha_0 &= \#^4 s_0, \\
w_t &= \#^4 s_f \#^3, \\
\pi_M &= \$_0 (\prod_{\substack{1 \leq i \leq m \\ p,q \in \Sigma \cup \{\#\}}} \$_i p v_i q \$_i) \$_{m+1}.
\end{aligned}$$

For a rewriting rule $i : u_i \to v_i$ of the Turing machine M and all $c_1, c_2, d_1, d_2, d_3, p, q \in \Sigma \cup \{\#\}$ we define the relations:

(i) $(c_1c_2, p, u_i q d_1 d_2 d_3) \sim (\$_i, p, v_i q \$_i)$ and

(ii) $(c_1c_2 p u_i, q, d_1 d_2 d_3) \sim (\$_i p v_i, q, \$_i)$.

Also we define the relation

(iii) $(\#\#\# s_f \#, \#, \#\#\#\$_0) \sim (\$_{m+1}, \#, \#)$.

We have to prove the following claim: a word $w \in \Sigma^*$ is accepted by M if and only if there is such k_w, that word $w\#\#\#\#\#\pi_M^{k_w}\#\#$ is accepted by G_M.

Let a word w be accepted by the given *Turing machine* M, by the derivations $\#s_0w\# \Rightarrow_{i_1} \#w_{l_1}s_{j_1}w_{r_1}\# \Rightarrow_{i_2} \#w_{l_2}s_{j_2}w_{r_2}\# \Rightarrow_{i_3} \ldots \Rightarrow_k \#w_{l_k}s_{j_k}w_{r_k}\# \Rightarrow_{k+1} \ldots \Rightarrow_n \#s_f\#$. We prove that there is an integer k_w big enough such that the word $w\#^5\pi_M^{k_w}\#\#$ is accepted by the recombinations $\alpha_0 w\#^5\pi_M^{k_w}\#\# = \#\#\#\#s_0w\#\#\#\#\#\pi_M^{k_w}\#\# \Rightarrow_{\mathsf{trl}_{p_1,q_1}} \#\#\#\#w_{l_1}s_{j_1}w_{r_1}\#\#\#\#\#\pi_1\#\# \Rightarrow_{\mathsf{trl}_{p_2,q_2}} \#\#\#\#w_{l_2}s_{j_2}w_{r_2}\#\#\#\#\#\pi_2\#\# \Rightarrow_{\mathsf{trl}_{p_3,q_3}} \ldots \Rightarrow_{\mathsf{trl}_{p_k,q_k}} \#\#\#\#w_{l_k}s_{j_k}w_{r_k}\#\#\#\#\#\pi_k\#\# \Rightarrow_{\mathsf{trl}_{p_{k+1},q_{k+1}}} \ldots \Rightarrow_{\mathsf{trl}_{p_n,q_n}} \#\#\#\#s_f\#\#\#\#\#\pi_n\#\#$, where $w_{l_i}, w_{r_i} \in \Sigma^*$, $s_{j_i} \in S$ and $\pi_i \in \Sigma'^*$ for all $1 \leq i \leq n$ and π_{i+1} differs from π_i only by a substring u_i which replaces substring v_i in π_i.

Since for each $k < n$ there is a rule i_k applicable to $\#w_{l_k}s_{j_k}w_{r_k}\#$, then $\#w_{l_k}s_{j_k}w_{r_k}\# = w'_{l_k}pu_{i_k}qw'_{r_k}$, where s_{j_k} is in u_{i_k}, $p, q \in \Sigma \cup \{\#\}$ and $w'_{l_k}, w'_{r_k} \in (\Sigma \cup \{\#\})^*$. We suppose, that the string π_k contains at least one copy of the substring $pv_{i_k}q$, i.e., $\pi_k = \$_0\$_1\omega'\$_{i_k}pv_{i_k}q\$_{i_k}\omega''\$_m\$_{m+1}$. Then there are two relations in our recombination system such as $\mathsf{trl}_{p,q}$ operation is applicable to the string $\#\#\#w'_{l_k}pu_{i_k}qw'_{r_k}\#\#\#\#\pi_k\#\#$.

Indeed, these relations are (i) $(c_1c_2, p, u_{i_k}qd_1d_2d_3) \sim (\$_{i_k}, p, v_{i_k}q\$_{i_k})$ and (ii) $(c_1c_2pu_{i_k}, q, d_1d_2d_3) \sim (\$_{i_k}pv_{i_k}, q, \$_{i_k})$. In this way, we can obtain the string $w''_{l_k}pv_{i_k}qw''_{r_k}\$_0\$_1\omega'\$_{i_k}pu_{i_k}q\$_{i_k}\omega''\$_{m+1}\#\# = \#^4w_{l_{k+1}}s_{j_{k+1}}w_{r_{k+1}}\#^5\pi_{k+1}\#^2$ from the string of $\#\#\#w'_{l_k}pu_{i_k}qw'_{r_k}\#\#\#\#\pi_k\#\# = w''_{l_k}pu_{i_k}qw''_{r_k}\$_0\$_1\omega'\$_{i_k}pv_{i_k}q\$_{i_k}\omega''$ $\$_{m+1}\#\#$, where $w''_{l_k} = \#\#\#w'_{l_k} = w'''_{l_k}c_1c_2$ and $w''_{r_k} = w'_{r_k}\#\#\#\# = d_1d_2d_3w'''_{r_k}$.

In this way, for each derivation step $\#w_k\# \Rightarrow_{i_k} \#w_{k+1}\#$ from the Turing machine M we have the corresponded recombination step $\#\#\#\#w_k\#\#\#\#\#\pi_k\#$ $\# \Rightarrow_{\mathsf{trl}_{p_k,q_k}} \#\#\#\#w_{k+1}\#\#\#\#\#\pi_{k+1}\#\#$, in the recombination system G_M.

Now, we have to provide the number k_w of copies of the π_M big enough, so as for each derivation $\#w_k\# \Rightarrow_{i_k} \#w_{k+1}\#$ we would have at least a copy of the substring v_{i_k} in the substring π_k. Such number k_w exists and it is Turing computable. Indeed, this can be for instance $k_w \geq n$, i.e., the number of derivations of M in order to accept the word w.

In this way, if w is accepted by M by the derivations $\#s_0w\# \Rightarrow \ldots \Rightarrow \#s_f\#$, then we can have recombination of $\#\#\#\#s_0w\#\#\#\#\#\pi_M^{k_w}\#\#$ to $\#\#\#\#s_f\#$ $\#\#\#\#\pi_n\#\#$ by trl operations in G_M. In order to accept $w\#\#\#\#\#\pi_M^{k_w}\#\#$ in G_M, we have to recombine $\#\#\#\#s_f\#\#\#\#\#\pi_n\#\#$ to the target $w_t = \#\#\#\#s_f\#\#\#$. This can be done by the deletion operation in the relation (iii): $\#\#\#\#s_f\#\ \#\#\#\#\pi_n\#\# \Rightarrow_{\mathsf{del}_\#} \#\#\#\#s_f\#\#\#$.

Now, we prove, that for each word $\#\#\#\#s_0w\#\#\#\#\#\pi_M^{k_w}\#\#$ accepted by the recombination system G_M, Turing machine M accepts word w too.

Assume, that there is such $w \in \Sigma$, that $\#\#\#\#s_0w\#\#\#\#\#\pi_M^{k_w}\#\#$ is accepted by G_M for some $k_w > 0$, but it is not accepted by M. That means, there are recombination operations possible which do not correspond to the derivation rules from M, i.e., there are possible recombinations of the form $\#\#\#\#w'\#\#\#\#\#\pi'$ $\#\# \Rightarrow_{\widetilde{R}} w''$, where $w' \in (\Sigma \cup S)^*$, $|w'|_S = 1$, $\pi', w'' \in \Sigma'^*$ and the recombination is not of the form $\#\#\#\omega'''pu_iq\omega^{iv}\#\#\#\#\omega^v\$_ipv_iq\$_i\omega^{vi}\#\# \Rightarrow_{\mathsf{trl}_{p,q}}$ $\#\#\#\omega'''pv_iq\omega^{iv}\#\#\#\#\omega^v\$_ipu_iq\$_i\omega^{vi}\#\#$, where $\omega''', \omega^{iv} \in (\Sigma\cup\{\#\})^*$, ω^v, ω^{vi} $\in \Sigma'^*$ and $p, q \in \Sigma \cup \{\#\}$. Such recombinations exist.

Assume, relation of the form (i) or (ii) is applicable to the string $\#\omega^{vii}c_1c_2pu_iq$ $d_1d_2d_3\omega^{viii}\#\$_0\omega^{ix}\$_ipv_iq\$_i\omega^x\$_{m+1}\#\#$, where $\omega^{vii}, \omega^{viii} \in (\Sigma\cup\{\#\})^*$, $\omega^{ix}, \omega^x \in \Sigma'^*$ and $c_1, c_2, d_1, d_2, d_3, p, q \in \Sigma\cup\{\#\}$, which is obtained from $\#\#\#\#s_0w\#\#\#$ $\#\#\$_0\pi_M^{k_w}\#\#$ only by translocations corresponding to the rules from P. Relation (ii) is not applicable to the string because we do not have substring $\#\#\#s_f\#$ in $\#\omega^{vii}c_1c_2pu_iqd_1d_2d_3\omega^{viii}\#\$_0\omega^{ix}\$_ipv_iq\$_i\omega^x\$_{m+1}\#\#$. Here we may have either:

Case del**:** $\varpi = \#\omega^{vii}c_1c_2pu_iqd_1d_2d_3\omega^{viii}\#\$_0\omega^{ix}\$_ipv_iq\$_i\omega^x\$_{m+1}\#\# \Rightarrow_{\mathsf{del}_p} \#$

$\omega^{vii}c_1c_2pv_iq\$_i\omega^x\$_{m+1}\#\# = \varpi'$ in the relation of the type (i), or $\#\omega^{vii}c_1c_2pu_iqd_1d_2d_3\omega^{viii}\#\$_0\omega^{ix}\$_ipv_iq\$_i\omega^x\$_{m+1}\#\# \Rightarrow_{\mathsf{del}_q} \#\omega^{vii}c_1c_2pu_iq\$_i\omega^x\$_{m+1}\#\# = \varpi''$ in the relation of the type (ii). Since relations of types (i) and (ii) both consider pair of pointers, one of which is from the left side and another one is from the right side of the substring $\#\#\#\#\#\$_0$ of string ϖ, substring $\#\#\#\#\#\$_0$ is deleted, we obtain either ϖ' or ϖ'' and after that it is not possible to reach by the recombination the string where the relation (iii) is applicable. Moreover, after the deletion operation either in the relation (i) or relation (ii), it is not possible to remove from the string symbol $\$_{m+1}$ in the relations (i) and (ii). Indeed, in any recombination in the relations (i) and (ii) of strings ϖ' and ϖ'' the suffix $\$_{m+1}\#\#$ is not affected.

Case trl**:** $\varpi = \#\omega^{vii}c_1c_2pu_iqd_1d_2d_3\omega^{viii}\#\$_0\omega^{xi}\$_ipv_iq\$_i\omega^{xii}\$_ipv_iq\$_i\omega^{xiii}\$_{m+1}\#\# \Rightarrow_{\mathsf{trl}_{p,q}} \#\omega^{vii}c_1c_2pv_iq\$_i\omega^{xii}\$_ipv_iqd_1d_2d_3\omega^{viii}\#\$_0\omega^{xi}\$_ipu_iq\$_i\omega^{xiii}\$_{m+1}\#\# = \varpi'''$ in the relations (i) and (ii), where $\omega^{xi}, \omega^{xii}, \omega^{xiii} \in \Sigma'^*$. Assume u_j is the substring of pv_iq. There is no context applicable to the string ϖ'''. Indeed, according to the definition of the Turing machine from above, the maximal length of the suffix containing S-symbol as the prefix in the right side of a derivation rule is 3 (type (3) $v_i = a_is_{j_i} \sqcup \#$ or type (5) $v_i = \#s_{j_i} \sqcup a_i$, we represent v_i as $v_i = a'_is_{i_j}a''_ia'''_i$, where $a'_i, a''_i, a'''_i \in (\Sigma \cup \{\#\})$) and in the rule of the type (7) ($as_f \to s_f$) S-symbol is the rightmost-symbol in the left side of the rule. There are no other types of rules where S-symbol is the rightmost in the left side of the rule. In this way, we consider that $s_{j_i} = s_f$. I.e., we have substring $pv_iq\$_i = pa'_is_fa''_ia'''_iq\$_i$. Relations (i) and (ii) are not applicable. Indeed, to the right from S-symbol we need to have at least 4 symbols not equal to $\$_i$ in order to satisfy the left condition of (i) and (ii) (i.e., $(c_1c_2, p, u_iqd_1d_2d_3)$ and $(c_1c_2pu_i, q, d_1d_2d_3)$). Similarly, we can show that to the left from S-symbol we need to have at least 3 symbols not $\$_i$ in order to satisfy the left conditions of the relations (i) and (ii). There are no other places in the string ϖ''' where left conditions of (i) and (ii) are satisfied, i.e., relations (i) and (ii) are not applicable as soon as the translocation involving symbols $\$_i$ is used.

There are no other recombinations possible in the relations (i), (ii) and (iii). It follows then, that as soon as we have recombination not corresponding to a rule from P, the target w_t cannot be reached, i.e., word $w\#\#\#\#\#\pi_M^{k_w}\#\#$ is accepted by G_M if and only if w is accepted by M. □

5 Final Remarks

In [7] the equivalence between a Turing machine language and a set of multisets of words was explored. Since we are working with the intramolecular model, we can prove here a universality result in a standard way, showing the equivalence of two families of languages.

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., Petre, I., Prescott, D. M., and Rozenberg, G., Universal and simple operations for gene assembly in ciliates. In: V. Mitrana and C. Martin-Vide (eds.) *Words, Sequences, Languages: Where Computer Science, Biology and Linguistics Meet*, Kluwer Academic, Dortrecht, (2001) pp. 329–342.

[3] 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 (2001) pp. 216–256.

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

[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] Jahn, C. L., and Klobutcher, L. A., Genome remodeling in ciliated protozoa. *Ann. Rev. Microbiol.* **56** (2000), 489–520.

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

[8] Kari, L., and Thierrin, G., Contextual insertion/deletions and computability. *Information and Computation* 131 (1996) pp. 47–61.

[9] 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.

[10] Landweber, L. F., and Kari, L., Universal molecular computation in ciliates. In: L. F. Landweber and E. Winfree (eds.) *Evolution as Computation*, Springer, Berlin Heidelberg New York (2002).

[11] Marcus, S., Contextual grammars, *Revue Roumaine de Matématique Pures et Appliquées*, 14 (1969), pp. 1525–1534.

[12] Păun, Gh., *Marcus Contextual Grammars*, Kluwer, Dordrecht, (1997).

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

[14] Prescott, D. M., The DNA of ciliated protozoa. *Microbiol. Rev.* **58**(2) (1994), 233–267.

[15] Prescott, D. M., Genome gymnastics: unique modes of DNA evolution and processing in ciliates. *Nat. Rev. Genet.* 1(3) (2000), 191–198.

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

[17] Salomaa, A., *Formal Languages*, Academic Press, New York (1973).

Turku
Centre *for*
Computer
Science

Lemminkäisenkatu 14 A, 20520 Turku, Finland | www.tucs.fi

University of Turku

- Department of Information Technology
- Department of Mathematics

Åbo Akademi University

- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration

- Institute of Information Systems Sciences

ISBN 978-952-121-876-7
ISSN 1239-1891