Computer Science Journal of Moldova, vol.**10**, no.3(30), 2002

On enhanced time-varying distributed H systems

Sergey Verlan

Abstract

An enhanced time-varying distributed H system (ETVDH system) is a slightly different definition of the time-varying distributed H system (TVDH system) [9] and it was proposed by M. Margenstern and Yu. Rogozhin in [4] under the name of "extended time-varying distributed H system". The main difference is that the components of the ETVDH system are H systems and therefore splicing rules may be applied more than once as it is done in TVDH systems. This leads to difficulties in investigating the behavior of such systems because they have a higher level of parallelism. It is proved that ETVDH systems of degree 2 (*i.e.* with 2 components) generate all recursively enumerable languages in a sequential way [7] and that ETVDH systems of degree 4 generate all recursively enumerable languages in a "parallel" way, modelling a formal type-0 grammar [11]. In this paper we improve the last result and we present an ETVDH system of degree 3 which generates all recursively enumerable languages modelling type-0 formal grammars. The problem of the existence of ETVDH systems of degree 2 which generate all recursively enumerable languages in a "parallel" way is left open.

1 Introduction

Head splicing systems (H systems) were the first theoretical model of biomolecular computing (DNA-computing) and they were introduced by T. Head. [2, 3]. Inspired from biology this model is a mathematical formalisation of some biological processes which may be used to perform computations.

The molecules from biology are replaced by words over a finite alphabet and the chemical reactions are replaced by a *splicing* operation. An H system specifies a set of rules used to perform splicing and a set of initial words or axioms. The computation is done by applying iteratively the rules to the set of words which we have until no more new words can be generated. This corresponds to a bio-chemical experiment where we have enzymes (splicing rules) and initial molecules (axioms) which are put together in a tube and we wait until the reaction stops.

Unfortunately H systems are not very powerful and a lot of other models introducing additional control elements were proposed. One of these well-known models is time-varying distributed H system (TVDH systems) recently introduced in [9] as another theoretical model of biomolecular computing, based on splicing operations. This model introduces *components*, which cannot all be used at the same time but one after another, periodically.

This aims at giving an account of real biochemical reactions where the work of enzymes essentially depends on the environment conditions. In particular, at any moment, only a subset of all available rules is in action. If the environment is periodically changed, then the active enzymes change also periodically.

One of the important features of TVDH systems is that we apply rules of a particular component to the molecules which we have only once and only the result of this application is passed to the next component.

We can take off this condition and we can permit an iterative application of rules of the same component to the molecules being already generated in a similar way as it is done in H systems. This leads to a higher degree of parallelism compared to TVDH systems because inside each component the splicing rules may be iteratively applied to copies of the molecules already generated.

The obtained model was first considered by M. Margenstern and Yu. Rogozhin in [4]. Originally it was called “extended time-varying distributed H systems”. This name was later changed to “enhanced time-varying distributed H systems”.

Now it is interesting to prove similar universality results as it is done for the case of TVDH systems: TVDH systems of degree 1 generate all recursively enumerable languages in a *sequential* way modelling Turing machines [6] and TVDH systems of degree 2 generate all recursively enumerable languages in a "*parallel*" way, modelling type-0 formal grammars [8].

Let us clarify some points.

Recall that a *sequential* process is defined by the presence of a clock and by actions which are performed in such a way that at most one action is performed at each top of the clock. It is plain that a *deterministic* Turing machine performs a sequential computation. In [6], the proof consists namely in simulating such a deterministic machine and so, it is a sequential computation, not a parallel one.

By contrast, a *parallel* computation is performed by several processes which either produce actions independently of any clock or, produce them at each top of a clock: in this case, it is possible that several processes perform an action at the same top of the clock and that the performed actions are different from one another. When the processes perform their actions at the tops of the same clock, we speak of a *synchronised parallel* computation. According to these definitions, a sequential computation is a very particular case of a synchronised parallel computation.

The rules of a type-0 grammar may contain several times the same non-terminal symbol in their right part. Accordingly, the description of all possible computations starting from the axiom leads to a tree, whose branches are the different possible computations. Now, it is not needed that, when a branch is followed by some process, the clock of this process be the same as the clock of another one which is defined by another process. And so, we can see the simulation of any type-0 formal grammar as a good criterion for a parallel computation.

For ETVDH systems the following results are obtained: ETVDH systems with one component may produce only regular languages [4], ETVDH systems of degree 2 generate all recursively enumerable languages in a *sequential* way [7] and ETVDH systems of degree 4 generate all recursively enumerable languages in a "*parallel*" way [11].

In this paper we improve the last result and we show that ETVDH systems of degree 3 generate all recursively enumerable languages in a "*parallel*" way modelling type-0 formal grammars.

The problem of the existence of "*parallel*" ETVDH systems of degree 2 which generate any recursively enumerable languages is left open.

This result was presented at the DNA8 conference held on June 10-13, 2002 in Sapporo, Japan, and it was published as an abstract in the conference preproceedings [12]. Here we present the full version of the paper.

2 Basic definitions

We recall some notions. An *alphabet* V is a finite, non-empty set whose elements are called *letters*. A *word* (over some alphabet V) is a finite (possibly empty) concatenation of letters (from V). The empty concatenations of letters is also called the *empty word* and is denoted by ε. The set of all words over V is denoted by V^*. A *language* (over V) is a set of words (over V).

A *formal grammar* G is a tuple $G = (N, T, P, S)$ of an alphabet N of so-called *non-terminal* letters, an alphabet T of so-called *terminal* letters, with $N \cap T = \emptyset$, an *initial letter* S from N, and a finite set P of *rules* of the form $u \to v$ with $u, v \in (N \cup T)^*$ and u contains at least one letter from N. Any rule $u \to v \in P$ is a substitution rule allowing to substitute any occurrence of u in some word w by v.

Formally, we write $w \Rightarrow_G w'$ if there is a rule $u \to v$ in P and words $w_1, w_2 \in (N \cup T)^*$ with $w = w_1 u w_2$ and $w' = w_1 v w_2$. We denote by $\Rightarrow_G^*$ the reflexive and transitive closure of $\Rightarrow$. I.e., $w \Rightarrow_G^* w'$ means that there is an integer n and words $w_1, \dots, w_n$ with $w = w_1, w' = w_n$ and $w_i \Rightarrow_G w_{i+1}$ for all $i, 1 \le i < n$.

The sequence $w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_n$ is also called a computation (from w_1 to w_n of length $n - 1$). A *terminal word* is a word in T^*; all terminal words computable from the initial letter S form the language $L(G)$ generated by G. More formally, $L(G) \overset{\text{def}}{=} \{w \in T^*;\ S \Rightarrow^* w\}$.

An *(abstract) molecule* is simply a word over some alphabet. A *splicing rule* (over alphabet V), is a quadruple (u_1, u_2, u'_1, u'_2) of words $u_1, u_2, u'_1, u'_2 \in V^*$, which is often written in a two dimensional way as follows: $\frac{u_1 \mid u_2}{u'_1 \mid u'_2}$.

A splicing rule $r = (u_1, u_2, u'_1, u'_2)$ is applicable to two molecules m_1, m_2 if there are words $w_1, w_2, w'_1, w'_2 \in V^*$ with $m_1 = w_1u_1u_2w_2$ and $m_2 = w'_1u'_1u'_2w'_2$, and produces two new molecules $m'_1 = w_1u_1u'_2w'_2$ and $m'_2 = w'_1u'_1u_2w_2$. In this case, we also write $(m_1, m_2) \vdash_r (m'_1, m'_2)$.

A pair $h = (V, R)$, where V is an alphabet and R is a finite set of splicing rules, is called an *splicing scheme* or an H *scheme*.

For an H scheme $h = (V, R)$ and a language $L \subseteq V^*$ we define:

$\sigma_h(L) = \sigma_{(V,R)}(L) \overset{\text{def}}{=} \{w, w' \in V^* | \exists w_1, w_2 \in L : \exists r \in R : (w_1, w_2) \vdash_r (w, w')\}$.

A *Head-splicing-system* [2, 3], or H *system*, is a construct:

$H = (h, A) = ((V, R), A)$ of an alphabet V, a set $A \subseteq V^*$ of initial molecules over V, the *axioms*, and a set $R \subseteq V^* \times V^* \times V^* \times V^*$ of splicing rules. H is called finite if A and R are finite sets. For any H scheme h and language $L \in V^*$ we define:

$\sigma_h^0(L) = L,$
$\sigma_h^{i+1}(L) = \sigma_h^i(L) \cup \sigma_h(\sigma_h^i(L)),$
$\sigma_h^*(L) = \cup_{i \geq 0} \sigma_h^i(L).$

The language generated by H system H is:

$L(H) \overset{\text{def}}{=} \sigma_h^*(A).$

Thus, the language generated by H system H is the set of all molecules that can be generated in H starting with A as initial molecules by iteratively applying splicing rules to copies of the molecules already generated.

We define the operation $\tilde{\sigma}_h(L)$ as follows [4]:

$\tilde{\sigma}_h(L) = \tilde{\sigma}_h(L' \cup L'') \overset{\text{def}}{=} \sigma_h^*(L')$, where

$L' = \{w_1 \in L | \exists w_2 \in L : \exists w, w' \in V^* : \exists r \in R : (w_1, w_2) \vdash_r (w, w')$ or $(w_2, w_1) \vdash_r (w, w')\}$, $L'' = L \setminus L'$, and $h = (V, R)$.

(We note that every language $L \subseteq V^*$ for every H scheme h can be split recursively into two subsets L' and L'').

So, in order to obtain $\tilde{\sigma}_h(L)$ we take the molecules from L which may enter a splicing rule from h (L') and after that we apply σ_h^* to this set and we obtain all possible iterated splicings of these molecules as well as the original molecules (from L') as a result.

Enhanced time-varying distributed H *system* [4] (of degree $n, n \geq 1$), (ETVDH system) is a construct:

$$E = (V, T, A, R_1, R_2, \ldots, R_n),$$

where V is an alphabet, $T \subseteq V$ is the *terminal alphabet*, $A \subseteq V^*$ is the finite set of *axioms*, R_i are *components*, i.e. finite sets of splicing rules over V, $1 \leq i \leq n$.

At each moment $k = n \cdot j + i$, for $j \geq 0$, $1 \leq i \leq n$, only component R_i is used for splicing the currently available strings.

$L_1 = A,$
$L_{k+1} = \tilde{\sigma}_{h_i}(L_k),$ for $i \equiv k(mod\ n),\ k \geq 1, 1 \leq i \leq n, h_i = (V, R_i).$

So the components R_i, $1 \leq i \leq n$, of a ETVDH system of degree n first apply a filter to their contents, *i.e.* eliminate the molecules which can not enter a splicing rule, and after that they work as corresponding H systems.

We say that a component R_i of a ETVDH system rejects the word w if w cannot enter any splicing rule from R_i. In this case we write $w \uparrow_{\mathrm{R}_i}$. We may omit R_i if the context allows us to do so.

The language generated by E is:

$$L(E) = (\cup_{k \geq 1} L_k) \cap T^*.$$

3 ETVDH systems of degree 3

Theorem *For any type-0 formal grammar $G = (N, T, P, S)$ there is an ETVDH system $E_G = (V, T, A, R_1, R_2, R_3)$ of degree 3 which simulates G, i.e. $L(G) = L(E_G)$.*

In order to prove the theorem we shall prove the following inclusions:

(i) $L(G) \subseteq L(E_G)$ and
(ii) $L(G) \supseteq L(E_G)$.

It seems that **(i)** is the most difficult part of the demonstration and we will focus our attention on it.

We define $E_G = (V, T, A, R_1, R_2, R_3)$ as follows.

Let $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(B = a_n)$.

In what follows we will assume the following:
$1 \leq i \leq n,\ 1 \leq j \leq n-1,\ 2 \leq l \leq n,\ 1 \leq k \leq 5, \mathbf{a} \in N \cup T \cup \{B\}$.

Alphabet: $V = N \cup T \cup \{B\} \cup \{X, Y, X_i, Y_i, X'_i, Y'_i, X''_j, Y''_j, X', Y', X'', Y'', X''', Y''', Y^{IV}, Z, Z_E, C, C_1, C_2, D, D_1, D_2, Z^k_E\}$

The terminal alphabet T is the same as for the formal grammar G.

Axioms: $A = \{XSBY, X'_i Z^3_E, Z^4_E Y'_i, ZY_i, X''Z, ZY^{IV}, X_i a_i Z, ZY''_j, X'Z, ZY', X_j Z, ZY, ZY''', XZ, X''_j Z, ZY'', X'''Z, ZY, CZ^k_E, CZ_E, Z^k_E D, Z_E D, C_2 Z'', D_2, Z'C_1, D_1\} \cup \{ZvY : \exists u \to v \in P\}$

Component R_1:

1.1 : $\dfrac{\varepsilon \mid uY}{Z \mid vY}$, $\exists u \to v \in P$; 1.2 : $\dfrac{\varepsilon \mid a_i Y}{Z \mid Y_i}$; 1.3 : $\dfrac{X_i \mid \mathbf{a}}{X'_i \mid Z_E}$;

1.4 : $\dfrac{\mathbf{a} \mid Y''_j}{Z \mid Y_j}$; 1.5 : $\dfrac{X' \mid \mathbf{a}}{X'' \mid Z}$; 1.6 : $\dfrac{\mathbf{a} \mid Y'''}{Z \mid Y^{IV}}$;

1.7 : $\dfrac{X'_i \mid Z_E}{C \mid Z^1_E}$; 1.8 : $\dfrac{X'_i \mid Z^3_E}{C \mid Z^4_E}$; 1.9 : $\dfrac{Z^1_E \mid Y'_i}{Z^2_E \mid D}$;

1.10 : $\dfrac{Z^4_E \mid Y'_i}{Z^5_E \mid D}$; 1.11 : $\dfrac{Z' \mid C_1}{D_1 \mid \varepsilon}$;

Component R_2:

2.1 : $\dfrac{X \mid \mathbf{a}}{X_i a_i \mid Z}$; 2.2 : $\dfrac{\mathbf{a} \mid Y'_l}{Z \mid Y''_{l-1}}$; 2.3 : $\dfrac{X'_1 \mid \mathbf{a}}{X' \mid Z}$; 2.4 : $\dfrac{\mathbf{a} \mid Y'_1}{Z \mid Y'}$;

2.5 : $\dfrac{X''_j \mid \mathbf{a}}{X_j \mid Z}$; 2.6 : $\dfrac{\mathbf{a} \mid Y''}{Z \mid Y'''}$; 2.7 : $\dfrac{\mathbf{a} \mid BY''}{Z' \mid \varepsilon}$; 2.8 : $\dfrac{X''' \mid \mathbf{a}}{X \mid Z}$;

2.9 : $\dfrac{X'_i \mid Z^1_E}{C \mid Z^2_E}$; 2.10 : $\dfrac{X'_i \mid Z^4_E}{C \mid Z^5_E}$;

2.11 : $\dfrac{Z^2_E \mid Y'_i}{Z^3_E \mid D}$; 2.12 : $\dfrac{Z^5_E \mid Y'_i}{Z_E \mid D}$; 2.13 : $\dfrac{C_2 \mid Z''}{\varepsilon \mid D_2}$;

Component R_3:

$$3.1:\ \begin{array}{c|c} \mathbf{a} & Y_i \\ \hline Z_E & Y_i' \end{array};\quad 3.2:\ \begin{array}{c|c} X_l' & \mathbf{a} \\ \hline X_{l-1}'' & Z \end{array};\quad 3.3:\ \begin{array}{c|c} \mathbf{a} & Y' \\ \hline Z & Y'' \end{array};\quad 3.4:\ \begin{array}{c|c} X'' & \mathbf{a} \\ \hline X''' & Z \end{array};$$

$$3.5:\ \begin{array}{c|c} X'' & \mathbf{a} \\ \hline \varepsilon & Z'' \end{array};\quad 3.6:\ \begin{array}{c|c} \mathbf{a} & Y^{IV} \\ \hline Z & Y \end{array};$$

$$3.7:\ \begin{array}{c|c} X_i' & Z_E^2 \\ \hline C & Z_E^3 \end{array};\quad 3.8:\ \begin{array}{c|c} X_i' & Z_E^5 \\ \hline C & Z_E \end{array};\quad 3.9:\ \begin{array}{c|c} Z_E & Y_i' \\ \hline Z_E^1 & D \end{array};\quad 3.10:\ \begin{array}{c|c} Z_E^3 & Y_i' \\ \hline Z_E^4 & D \end{array};$$

Note

Each component contains also rules $\begin{array}{c|c} \alpha & \varepsilon \\ \hline \alpha & \varepsilon \end{array}$ for all axioms α except $XSBY$, $X_iZ_E^3$ and $Z_E^4Y_i$.

Proof of (i)

Notations

We shall use the following notation:

$$\begin{array}{c|c} w_1 & w_2 \\ \hline w_1' & w_2' \end{array} \quad \vdash_r \quad \frac{w_1w_2'^{\,*}}{w_1'w_2}, \quad w_1w_2$$

where $*$ indicates a possible occurrence of $\uparrow$. In the left part an application of the rule r on w_1w_2 and $w_1'w_2'$ is indicated and the right part contains the resulting two molecules w_1w_2' and $w_1'w_2$. We will use also the following convention: the upper part on both sides will contain the molecules we are interested in and the lower part will contain either an axiom or a molecule containing Z and which does not alter the further computation. The $\uparrow$ indicates the rejection of the considered molecule by the next component as defined above. The optional term of the right side in the formula, w_1w_2, is omitted if that molecule, which in principle enters the next component, is rejected by the next component. In the other case it is written.

"Rotate-and-simulate" method

The system uses the method of words rotation [1, 10]. Let us recall them briefly. For any word $w = w'w'' \in (N \cup T)^*$ of a formal grammar

G the word $Xw''Bw'Y$ $(X, Y, B \notin N \cup T)$ of the ETVDH system E_G, is called a "rotation version" of the word w. The system E_G models the formal grammar G as follows. It rotates the word Xw_1uw_2BY into Xw_2Bw_1uY and applies a splicing rule $\frac{\varepsilon \mid uY}{Z \mid vY}$. So we model the application of a rule $u \to v$ of formal grammar G by a single scheme-rule. E_G rotates the word Xwa_iY $(a_i \in (N \cup T \cup \{B\})$ "symbol by symbol", i.e., the word Xa_iwY will be obtained after some steps of working of the system E_G.

The rotation technique is as follows [4, 10]. We start with the word Xwa_iY in component R_1. Component R_2 receives the word XwY_i from component R_1. Component R_3 receives words $X_ja_jwY_i$ $(1 \leq j \leq n)$ from component R_2. After that point the system works in a cycle where indexes i and j decrease. If $j \neq i$, then the words with these indexes are eliminated. When $i = j$ we get the word $X_1a_iwY_1$ after some steps. After this we obtain the word Xa_iwY (so we rotated the word Xwa_iY) and the word a_iw' (if $a_iw = a_iw'B$) as possible result.

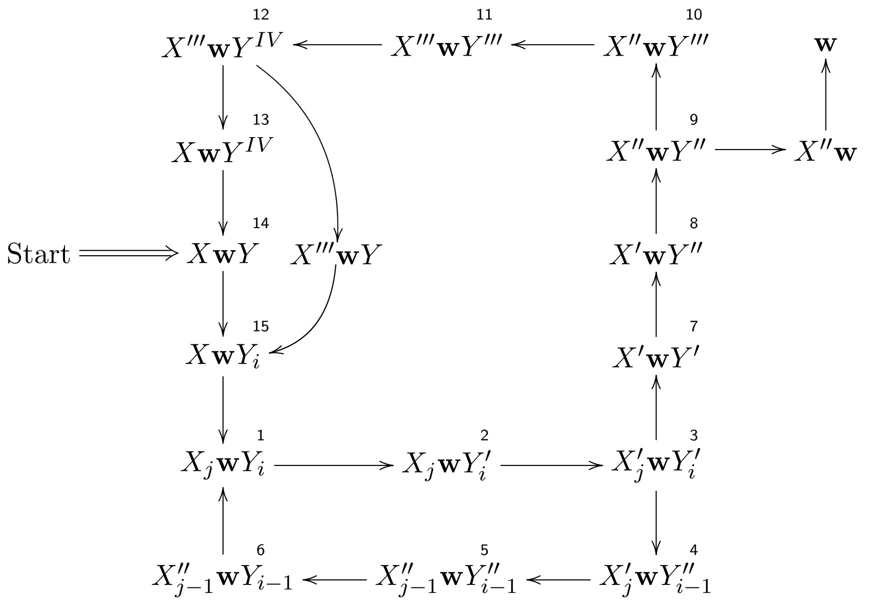
The system is made in a such way that molecules $X_i'Z_E$ appear in the first component only during an even step of computation. This means that if we have these molecules in the first component then the next time that we will be in this component (in 3 steps) these molecules $(X_i'Z_E)$ will no more exist. This is a very important feature of the system and it permits a correct simulation to be performed. A similar thing happens for molecules Z_EY_i'. They appear in the third component only if we are in the odd step.

The molecule Z' is produced in the first component and it appears only in the second component and the molecule Z'' is produced in the second component and it appears only in the third.

Computation

The computation follows the flow-chart shown in the figure 1. The vertices of the flow-chart show a configuration of molecules during the computation. We enumerate all configurations and their numbers are in the upper right corner. In configurations, **w** is treated as a variable, and it may have different values in different configurations. For example, if in configuration 14 $\mathbf{w}$ is equal to $w'a_i$ then in configuration 15 $\mathbf{w}$ may be w'. We will show that the computation follows the flow-chart i.e. all molecules produced from one configuration will be eliminated except molecules from the next configuration.

Because the length of both cycles is an even number we have the same parity of the computation step for all molecules in a particular configuration. Additionally it is easy to check that molecules from a configuration always arrive into the component with the same number. So we can say that each configuration has a component number and a parity of the step number associated with it.



$$\begin{tikzcd}
 & \overset{12}{X'''\mathbf{w}Y^{IV}} \arrow[d] \arrow[dr, bend left] & & \overset{11}{X'''\mathbf{w}Y'''} \arrow[ll] & \overset{10}{X''\mathbf{w}Y'''} \arrow[l] & \mathbf{w} \\
 & \overset{13}{X\mathbf{w}Y^{IV}} \arrow[d] & & & \overset{9}{X''\mathbf{w}Y''} \arrow[u] \arrow[r] & X''\mathbf{w} \arrow[u] \\
\text{Start} \arrow[r, Rightarrow] & \overset{14}{X\mathbf{w}Y} \arrow[d] & X'''\mathbf{w}Y \arrow[dl, bend left] & & \overset{8}{X'\mathbf{w}Y''} \arrow[u] & \\
 & \overset{15}{X\mathbf{w}Y_i} \arrow[d] & & & \overset{7}{X'\mathbf{w}Y'} \arrow[u] & \\
 & \overset{1}{X_j\mathbf{w}Y_i} \arrow[rr] & & \overset{2}{X_j\mathbf{w}Y'_i} \arrow[r] & \overset{3}{X'_j\mathbf{w}Y'_i} \arrow[u] \arrow[d] & \\
 & \overset{6}{X''_{j-1}\mathbf{w}Y_{i-1}} \arrow[u] & & \overset{5}{X''_{j-1}\mathbf{w}Y''_{i-1}} \arrow[ll] & \overset{4}{X'_j\mathbf{w}Y''_{i-1}} \arrow[l] &
\end{tikzcd}$$

Figure 1:The flow-chart of the computation

Because we deal with an ETVDH system, there are two types of molecules which pass to the next step: the generated molecules and the molecules which generated them. We will show that the last ones will be generally eliminated during the next step (including other molecules that may be produced from them during this next step) and we will discuss in detail other cases in order to show that we perform a correct simulation.

We will show several steps of the computation and after that we will discuss each configuration in detail.

We start with the word $XwY = Xw'a_iY$, $1 \le i \le n$.

$$\begin{array}{c|c} Xw' & a_iY \\ \hline Z & Y_i \end{array} \quad \vdash_{1.2} \quad \frac{Xw'Y_i}{Za_iY}, \quad Xw'a_iY$$

So $Xw'a_iY$ and $Xw'Y_i$ go to the next component.

We consider the evolution of $Xw'a_iY$:

$$\begin{array}{c|c} X & w'a_iY \\ \hline X_ka_k & Z \end{array} \quad \vdash_{2.1} \quad \frac{X_ka_kw'a_iY \uparrow}{XZ}, \quad k \in \{1, \dots, n\}$$

So it does not produce any new molecules. For $Xw'Y_i$ we have:

$$\begin{array}{c|c} X & w'Y_i \\ \hline X_ka_k & Z \end{array} \quad \vdash_{2.1} \quad \frac{X_ka_kw'Y_i}{XZ}, \quad Xw'Y_i\,, \quad k \in \{1, \dots, n\}$$

So $Xw'Y_i$ and $X_ka_kw'Y_i$ go to the next component.

We are on an odd step. So Z_EY_i' exist and we may apply the following rule:

$$\begin{array}{c|c} Xw' & Y_i \\ \hline Z_E & Y_i' \end{array} \quad \vdash_{3.1} \quad \frac{Xw'Y_i' \uparrow}{Z_EY_i}$$

So, there is no further evolution of $Xw'Y_i$. For $X_ka_kw'Y_i$ we get:

$$\begin{array}{c|c} X_ka_kw' & Y_i \\ \hline Z_E & Y_i' \end{array} \quad \vdash_{3.1} \quad \frac{X_ka_kw'Y_i'}{Z_EY_i}$$

So $X_ka_kw'Y_i$ and $X_ka_kw'Y_i'$ go to the next component.

And so on...

Now we will discuss each configuration and we will group configurations which have the same behavior.

Group 1 Configurations $4, 5, 6, 7, 11, 13, 15$ (plain).

We will discuss in detail configuration 5.

$$\begin{array}{c|c} X''_{j-1}w & Y''_{i-1} \\ \hline Z & Y_{i-1} \end{array} \quad \vdash_{1.4} \quad \frac{X''_{j-1}wY_{i-1}}{ZY''_{i-1}}, \quad X''_{j-1}wY''_{i-1}\,, \quad 2 \le i, j \le n$$

The molecule $X''_{j-1}wY_{i-1}$ is in configuration 6.

The molecule $X''_{j-1}wY''_{i-1}$ will be eliminated during the next step.

$$\begin{array}{c|c} X''_{j-1} & wY''_{i-1} \\ \hline X_{j-1} & Z \end{array} \quad \vdash_{2.5} \quad \frac{X_{j-1}wY''_{i-1}\uparrow}{X''_{j-1}Z}$$

Group 2 Configuration 14 ($u \to v$).

We may perform computations which are similar to the group 1. We may also apply the rule 1.1 thus we model the application of the rule $u \to v$ of G.

Group 3 Configurations 8 and 10 (Z', Z'').

We may perform computations which are similar to the group 1. We may also apply the rule 2.7 (3.5 respectively) but this leads to nothing. The detailed computation of this application for configuration 10 is:

$$\begin{array}{c|c} X'' & wY''' \\ \hline \varepsilon & Z'' \end{array} \quad \vdash_{3.5} \quad \frac{wY'''}{X''Z''}, \quad X''wY'''$$

$$\begin{array}{c|c} X''w & Y''' \\ \hline Z & Y^{IV} \end{array} \quad \vdash_{1.6} \quad \frac{X''wY^{IV}\uparrow}{ZY'''}$$

$$\begin{array}{c|c} w & Y''' \\ \hline Z & Y^{IV} \end{array} \quad \vdash_{1.6} \quad \frac{wY^{IV}\uparrow}{ZY'''}$$

Group 4 Configuration 9 (branch to result).

We may perform computations which are similar to the group 3. We may also apply rules 2.7 and 3.5 consecutively. Thus we may obtain the result w if $w \in T^*$.

$$\begin{array}{c|c} X''w'' & BY'' \\ \hline Z' & \varepsilon \end{array} \quad \vdash_{2.7} \quad \frac{X''w''}{Z'BY''}, \quad X''w''BY''$$

$$\begin{array}{c|c} X'' & w \\ \hline \varepsilon & Z'' \end{array} \quad \vdash_{3.5} \quad \frac{\mathbf{w}\uparrow}{X''Z''}$$

Group 5 Configurations 1 and 2 (junk).

We may perform computations which are similar to the group 1. We may also have the following computation (for configuration 1):

$$\begin{array}{c|c} X_1w & Y_i \\ \hline Z_E & Y'_i \end{array} \quad \vdash_{3.1} \quad \frac{X_1wY'_i}{Z_EY_i}, \quad X_1wY_i\,, \quad 1 \le i \le n$$

The molecule X_1wY_i' is in configuration 2.

$$\begin{array}{c|c} X_1 & wY_i \\ \hline X_1' & Z_E \end{array} \vdash_{1.3} \frac{X_1'wY_i}{X_1Z_E}$$

This is one of the cases when the generating molecule produce a molecule which is not eliminated immediately.

$$\begin{array}{c|c} X_1' & wY_i \\ \hline X' & Z \end{array} \vdash_{2.3} \frac{X'wY_i \uparrow}{X_1'Z}$$

The molecules $X_1'wY_i$ and $X'wY_i$ are rejected by the next component because we can not apply the rule 3.1 as the next step is an even step and the molecules Z_EY_i' do not exist.

And for configuration 2:

$$\begin{array}{c|c} X_k & wY_1' \\ \hline X_k' & Z_E \end{array} \vdash_{1.3} \frac{X_k'wY_1'}{X_kZ_E}, \quad X_kwY_1', \quad 1 \leq k \leq n$$

The molecule $X_k'wY_1'$ is in configuration 3.

$$\begin{array}{c|c} X_kw & Y_1' \\ \hline Z & Y' \end{array} \vdash_{2.4} \frac{X_kwY'}{ZY_1'}$$

$$\begin{array}{c|c} X_kw & Y' \\ \hline Z & Y'' \end{array} \vdash_{3.3} \frac{X_kwY'' \uparrow}{ZY'}$$

The molecules X_kwY' and X_kwY'' are rejected by the next component because we can not apply the rule 1.3 as the next step is an odd step and the molecules $X_i'Z_E$ do not exist.

Group 6 Configuration 3 (choice).

There are 4 cases depending on the value of i and j with respect to 1:

a) $X_j'wY_1'$, b) $X_1'wY_i'$, c) $X_1'wY_1'$, d) $X_j'wY_i'$, $2 \leq i, j \leq n$

a) $\mathbf{X_j'wY_1'}$ case.

$$\begin{array}{c|c} X_j'w & Y_1' \\ \hline Z & Y' \end{array} \vdash_{2.4} \frac{X_j'wY'}{ZY_1'}, \quad X_j'wY_1'$$

$$\begin{array}{c|c} X_j' & wY_1' \\ \hline X_{j-1}'' & Z \end{array} \vdash_{3.2} \frac{X_{j-1}''wY_1' \uparrow}{X_j'Z}$$

For the molecule X'_jwY' we may apply two rules, namely 3.2 and 3.3. We get the following:

$$\begin{array}{c|c} X'_jw & Y' \\ \hline Z & Y'' \end{array} \quad \vdash_{3.3} \quad \frac{X'_jwY'' \uparrow}{ZY''}$$

$$\begin{array}{c|c} X'_j & wY' \\ \hline X''_{j-1} & Z \end{array} \quad \vdash_{3.2} \quad \frac{X''_{j-1}wY' \uparrow}{X'_jZ}$$

We may apply one more time the rules 3.3 and 3.2 to the molecules from the previous two lines and we get the following:

$$\begin{array}{c|c} X''_{j-1}w & Y' \\ \hline Z & Y'' \end{array} \quad \vdash_{3.3} \quad \frac{X''_{j-1}wY'' \uparrow}{ZY'}$$

So this computation does not lead to new results.

b) $\mathbf{X'_1wY'_i}$ case.

For the molecule $X'_1wY'_i$ we may apply two rules, namely 2.2 and 2.3. We get the following:

$$\begin{array}{c|c} X'_1w & Y'_i \\ \hline Z & Y''_{i-1} \end{array} \quad \vdash_{2.2} \quad \frac{X'_1wY''_{i-1} \uparrow}{ZY'_i}$$

$$\begin{array}{c|c} X'_1 & wY'_i \\ \hline X' & Z \end{array} \quad \vdash_{2.3} \quad \frac{X'wY'_i \uparrow}{X'_1Z}$$

We may apply one more time the rules 2.3 and 2.2 to the molecules from the previous two lines and we get the following:

$$\begin{array}{c|c} X'w & Y'_i \\ \hline Z & Y''_{i-1} \end{array} \quad \vdash_{2.2} \quad \frac{X'wY''_{i-1} \uparrow}{ZY'_i}$$

So this computation does not lead to new results.

c) $\mathbf{X'_1wY'_1}$ case.

For the molecule $X'_1wY'_1$ we may apply two rules, namely 2.4 and 2.3. We get the following:

$$\begin{array}{c|c} X'_1w & Y'_1 \\ \hline Z & Y' \end{array} \quad \vdash_{2.4} \quad \frac{X'_1wY'}{ZY'_1}$$

$$\begin{array}{c|c} X'_1 & wY'_1 \\ \hline X' & Z \end{array} \quad \vdash_{2.3} \quad \frac{X'wY'_1 \uparrow}{X'_1Z}$$

We may apply one more time the rules 2.3 and 2.4 to the molecules from the previous two lines and we get the following:

$$\begin{array}{c|c} X'_1 & wY' \\ \hline X' & Z \end{array} \quad \vdash_{2.3} \quad \frac{\mathbf{X'wY'}}{X'_1Z}, \quad X'_1wY'$$

The molecule $X'wY'$ is in configuration 7.

$$\begin{array}{c|c} X_1'w & Y' \\ \hline Z & Y'' \end{array} \quad \vdash_{3.3} \quad \frac{X_1'wY''\uparrow}{ZY'}$$

d) $\mathbf{X_j'wY_i'}$, $\mathbf{i,j>1}$ case.

$$\begin{array}{c|c} X_k'w & Y_i' \\ \hline Z & Y_{i-1}'' \end{array} \quad \vdash_{2.2} \quad \frac{\mathbf{X_k'wY_{i-1}''}}{ZY_i'}, \quad X_k'wY_i'$$

The molecule $X_k'wY_{i-1}''$ is in configuration 4.

$$\begin{array}{c|c} X_k' & wY_i' \\ \hline X_{k-1}'' & Z \end{array} \quad \vdash_{3.2} \quad \frac{X_{k-1}''wY_i'\uparrow}{X_k'Z}$$

Group 7 Configuration 12 (parallel branch).

$$\begin{array}{c|c} X''' & wY^{IV} \\ \hline X & Z \end{array} \quad \vdash_{2.8} \quad \frac{XwY^{IV}}{X'''Z}, \quad X'''wY^{IV}$$

The molecule XwY^{IV} is in configuration 13.

$$\begin{array}{c|c} X'''w & Y^{IV} \\ \hline Z & Y \end{array} \quad \vdash_{3.6} \quad \frac{X'''wY}{ZY^{IV}}$$

$$\begin{array}{c|c} X'''w' & a_iY \\ \hline Z & Y_i \end{array} \quad \vdash_{1.2} \quad \frac{X'''w'Y_i}{Za_iY}, \quad X'''w'a_iY\ , \quad 1\le i\le n$$

$$\begin{array}{c|c} X''' & wY \\ \hline X & Z \end{array} \quad \vdash_{2.8} \quad \frac{XwY}{X'''Z}$$

$$\begin{array}{c|c} X''' & w'Y_i \\ \hline X & Z \end{array} \quad \vdash_{2.8} \quad \frac{Xw'Y_i}{X'''Z}$$

The molecules obtained XwY and $Xw'Y_i$ are the same to those obtained following the flow-chart. So they already exist during this step.

$$\begin{array}{c|c} X'''w' & Y_i \\ \hline Z_E & Y_i' \end{array} \quad \vdash_{3.1} \quad \frac{X'''w'Y_i'\uparrow}{Z_EY_i}$$

Proof of (ii)

The proof follows directly from the construction of the system. We can see that E_G correctly simulates the rule $u\to v\in P$ using rotation and the rule 1.1 as described before. The letters X, Y and B are removed only when we have B at the end of the word. This means that we use the right “rotational variant” of the word in order to obtain the corresponding terminal string. So, if $w \in L(E_G)$ then $w \in L(G)$. $\square$

Acknowledgments The author wants to thank M. Margenstern and Yu. Rogozhin for their very helpful comments and for their attention for the present work. The author acknowledges also the NATO project PST.CLG.976912, the “MolCoNet” IST-2001-32008 project and the *Laboratoire d'Informatique Théorique et Apliquée de Metz* which provided him best conditions for producing the present result. And finally the author acknowledges the Ministry of Education of France for the financial support of his PhD.

References

[1] Ferretti C., Mauri G., Zandron C., *Nine test tubes generate any RE language,* Theoretical Computer Science, **231**, N2, 171–180 (2000).

[2] Head T., *Formal language theory and DNA: an analysis of the generative capacity of recombinant behaviors.* Bulletin of Mathematical Biology, **49**, 737–759 (1987).

[3] Head T., Păun Gh., Pixton D., *Language theory and molecular genetics. Generative mechanisms suggested by DNA recombination,* chapter 7 in vol.2 of G.Rozenberg, A.Salomaa, eds., *Handbook of Formal Languages*, 3 volumes, Springer-Verlag, Heidelberg, 1997.

[4] Margenstern M., Rogozhin Yu., *About Time-Varying Distributed H Systems.* Lecture Notes in Computer Science, Springer, vol. **2054**, 2001, pp.53–62. (Proceedings of DNA6, Leiden, The Netherlands, June 13-17, 2000).

[5] Margenstern M., Rogozhin Yu., *A universal time-varying distributed H system of degree 1.* Proceedings of the DNA7, Seventh International Meeting on DNA Based Computers, Univ. of South Florida, U.S.A., June 10-13, 2001, (2002) - LNCS (to appear).

[6] Margenstern, M., Rogozhin, Yu., *Time-varying distributed H systems of degree 1 generate all recursively enumerable languages,* in vol. Words, Semigroups and Transductions (M. Ito, Gh. Paun, S. Yu, eds), World Scientific, Singapore, 2001, p. 329-340.

[7] Margenstern M., Rogozhin Yu., *Extended Time-Varying Distributed H Systems - Universality Result.* In Proceedings of The 5th World Multi-Conference on Systemics, Cybernetics and Informatics, Industrial Systems, SCI 2001, Orlando, Florida USA, July 22-25, 2001, vol.IX.

[8] Margenstern, M., Rogozhin, Yu., Verlan, S., *Time-Varying Distributed H Systems of Degree 2 Can Carry Out Parallel Computations.* Preproceedings of DNA8, Sapporo, Japan, 10-13 June 2002, p. 271–280.

[9] Păun G., *DNA computing: distributed splicing systems.* in *Structures in Logic and Computer Science. A Selection of Essays in honor of A. Ehrenfeucht*, Lecture Notes in Computer Science **1261**, Springer-Verlag, 353–370 (1997).

[10] Păun G., *DNA Computing Based on Splicing: Universality Results.* Theoretical Computer Science, **231**, N2, 275-296 (2000).

[11] Verlan S., *On extended time-varying distributed H systems.* In Proceedings of 6th International Workshop on DNA-Based Computers, DNA 2000, Leiden, The Netherlands, June 13-17, 2000, p. 281

[12] Verlan S., *On enhanced time-varying distributed H systems.* Preproceedings of DNA8, Sapporo, Japan, 10-13 June 2002, p. 339.

Sergey Verlan,

Received July 31, 2002

Laboratoire d'Informatique Théorique et Appliquée
Université de Metz, France,
E–mail:$verlan@lita.univ-metz.fr$