About Time-Varying Distributed H Systems

Maurice Margenstern[1] and Yurii Rogozhin[2]

[1] Institut Universitaire de Technologie,
Université de Metz, Metz, France
LITA, EA 3097
(Laboratoire d'Informatique Théorique et Appliquée)
margens@antares.iut.univ-metz.fr

[2] Institute of Mathematics and Computer Science of the
Academy of Sciences of Moldova
rogozhin@math.md

Abstract. A time-varying distributed H system (TVDH system) is a splicing system which has the following feature: at different moments one uses different sets of splicing rules (these sets are called components of TVDH system). The number of components is called the degree of the TVDH system. The passing from a component to another one is specified in a cycle. It was proved by both authors (1999) that TVDH systems of degree 2 generate all recursively enumerable languages. It was made by modelling Turing machines and, in that modelling, every language is generated "step by step" or "word by word". This solution is not a fully parallel one. A.Paun (1999) presented a complete parallel solution for TVDH systems of degree 4 by modelling *type* 0 formal grammars. Now we improved A.Paun's result by reducing the number of components of such TVDH systems down to 3. This question is open for 2 components, i.e. is it possible to construct TVDH systems of degree 2 which completely uses the parallel nature of molecular computations based on splicing operations (say model *type* 0 formal grammars)?
We consider also original G.Paun's definition of TVDH systems and suggest a slightly different definition of TVDH systems based on the definition of H systems – extended time-varying distributed H systems (ETVDH systems). For this new definition we proved that ETVDH systems with one component generate exactly the set of all regular languages and that with two components, they generate exactly the set of all recursively enumerable languages.

1 Introduction

Starting from [1], a grounding paper on splicing computations, a lot of studies were devoted to various extensions of H systems originating from [3]. In particular, a lot of them point at the possible universality power of such systems. Extending the original definition, paper [11] defined the notion of *test tube* and proved the universality of an extended H system with a finite number of test tubes. But no indication on the number of test tubes needed in order to obtain universality was given in [11]. That number was first set to 9 [2], then to 6 [12], and finally established to 3 in [16]. The latter result is very near to the real frontier between decidability and undecidability or, in other terms, between the possibility and the impossibility of predicting the eventual behaviour of such systems, depending on the number of test tubes. Indeed, as it is known that for a single test tube, generated languages are regular [15], it remains to examine the case of two test tubes, which is still open up to now.

Time-varying distributed H systems were recently introduced in [12], [13, 14] as another theoretical model of biomolecular computing (DNA-computing), based on splicing operations. Instead of considering *test tubes*, these models introduce *components*, later see the formal definition, 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 are in action. If the environment is periodically changed, then the active enzymes change also periodically.

In [13], it is proved that 7 different *components* are enough in order to generate any recursively enumerable language. Recently, see [6, 7], both authors proved that two components are enough to construct a *universal* time-varying distributed H system, *i.e.* a time-varying distributed H system, capable of simulating the computation of any Turing machine. Universality of computation and generating any recursively enumerable language are equivalent properties, but it is *a priori* not necessarily true, that the universality of some time-varying distributed H system with two components entails that time-varying distributed H systems generate all recursively enumerable languages, with only two components. We proved in [8] that this is the case: 2 different components are enough in order to generate any recursively enumerable language. That result was presented by both authors at FBDU'98, satellite workshop of MFCS/CSL'98 Federal Conference, August 22 - 30, 1998, Brno, Czech Republic. The proof was made by modelling Turing machines and, whithin that frame, every language is generated "step by step" or "word by word".

In the meanwhile in [9], it was proved that time-varying distributed H systems with 4 components generate all recursively enumerable languages. That solution by A.Paun is considered as a "parallel" one. Indeed, although a formal definition of a parallel DNA computation is not known, it is generally agreed that if a given family of systems is able to mimic any *type* 0 formal grammar, such a computation can be considered as a paradigm of parallel computation. So, A.Paun's result about TVDH systems of degree 4 can be considered as a universal parallel DNA computation.

Now we improved A.Paun's result by reducing the number of components of TVDH systems which model *type* 0 formal grammars down to 3. The corresponding question for 2 components remains open.

We consider also G.Paun's original definition of TVDH systems and suggest a slightly different definition of such systems based on the definition of H systems (so-called extended TVDH systems or ETVDH systems). For ETVDH systems we proved that these systems with one component generate exactly the set of all regular languages and with two components exactly the set of all recursively enumerable languages.

Accordingly, our paper settles a frontier between decidability and undecidability for extended time-varying distributed H systems in terms of the number of their components: two is the smallest required number of components in order to generate all recursively enumerable sets.

The question of the status of time-varying distributed H system of degree one is still open.

We consider the "parallel" and "nonparallel" aspects also for ETVDH systems. Our solution for ETVDH systems of degree 2 is the same as for TVDH systems of degree 2 and so it is not a "parallel" one. Therefore there is a question: how many components are needed for ETVDH systems to generate any recursively enumerable languages in a parallel fashion? A partial answer is given in [19]: ETVDH systems of degree 4 generate any recursively enumerable language by modelling *type* 0 formal grammar. Accordingly, that solution is "parallel".

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* (of *type* 0) 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 $u, v \in (N \cup T)^*$. Note that u contains at least one symbol 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 $w_1, \cdots, w_n$ 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, $L(G) \stackrel{\text{def}}{=} \{w \in T^*;\, S \Rightarrow^* w\}$.

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: $\frac{u_1 | u_2}{u_1' | u_2'}$.

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 $m_1 = w_1 u_1 u_2 w_2$ and $m_2 = w_1' u_1' u_2' w_2'$, and produces two new molecules $m_1' = w_1 u_1 u_2' w_2'$ and $m_2' = w_1' u_1' u_2 w_2$. 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 a *splicing scheme* or an H *scheme*.

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

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

A *Head-splicing-system* [3,4], or H *system*, is a construct:

$$H = (h, A) = ((V, R), A),$$

which consists 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 any language $L \in V^*$ we define:

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

The language generated by H system H is defined as $L(H) \stackrel{\text{def}}{=} \sigma_h^*(A)$.

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.

A time-varying distributed H *system* (of degree n, $n \geq 1$), (TVDH system) is a construct:

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

where V is *an alphabet*, $T \subseteq V$ is *a terminal alphabet*, $A \subseteq V^*$ is a finite set of *axioms*, and *components* R_i are 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. Specifically, we define

$$\begin{aligned}
&L_1 = A, \\
&L_{k+1} = \sigma_{h_i}(L_k), \text{ for } i \equiv k(mod\ n),\ k \geq 1, 1 \leq i \leq n,\ h_i = (V, R_i).
\end{aligned}$$

Therefore, from a step k to the next step, $k + 1$, one passes only the result of splicing the strings in L_k according to the rules in R_i for $i \equiv k(mod\ n)$; the strings in L_k that cannot enter a splicing are removed.

The language generated by D is, by definition:

$$L(D) \stackrel{\text{def}}{=} (\cup_{k \geq 1} L_k) \cap T^*.$$

We denote by REG the set of all regular languages, by RE the set of all recursively enumerable languages, by VDH_n, $n \geq 1$, the family of languages generated by time-varying distributed H systems of degree at most n, and by VDH_* the family of all languages of this type.

3 TVDH Systems of Degree 3

Theorem 1. *For any type 0 formal grammar $G = (N, T, P, S)$ there is a TVDH system $D_G = (V, T, A, R_1, R_2, R_3)$ of degree 3 which simulates G and $L(G) = L(D_G)$.*

Proof. We construct $D_G = (V, T, A, R_1, R_2, R_3)$ as follows. Let be $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(a_n = B)$ and $B \notin \{N \cup T\}$.

Alphabet $V = N \cup T \cup \{B\} \cup \{X, Y, Z, X_i, Y_i, X'_j, Y'_j, X''_j, Y''_j, X', Y', Z', Z''\}$, $1 \leq i \leq n$, $1 \leq j \leq n-1$.

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

Axioms $A = \{XSBY\} \cup \{ZvY_i, \exists u : u \to va_i \in P,\ i \in \{1, \ldots, n-1\}\} \cup \{ZY_i, X'_jZ, X'Z, X_ia_iZ, ZY''_j, X_jZ, XZ, ZY'_j, ZY', X''_jZ, ZY, Z', Z'', 1 \leq i \leq n, 1 \leq j \leq n-1\}$.

Component R_1.

$1.1 : \dfrac{\varepsilon \mid uY}{Z \mid vY_i}$ $\quad \exists u : u \to va_i \in P,\ i \in \{1, \ldots, n-1\}$,

$1.2 : \dfrac{\varepsilon \mid a_iY}{Z \mid Y_i}$ $\quad 1 \leq i \leq n$,

$1.3 : \dfrac{X_j \mid \varepsilon}{X'_{j-1} \mid Z}$ $\quad 2 \leq j \leq n$, $\quad 1.4 : \dfrac{X_1 \mid \varepsilon}{X' \mid Z}$

$1.5 : \dfrac{\varepsilon \mid Y''_j}{Z \mid Y_j}$ $\quad 1 \leq j \leq n-1$,

$1.6 : \dfrac{\varepsilon \mid a_iuY}{Z \mid Y_i}$ $\quad \exists u : u \to \varepsilon \in P,\ 1 \leq i \leq n$,

Component R_2.

$2.1 : \dfrac{X \mid \varepsilon}{X_ia_i \mid Z}$ $\quad 1 \leq i \leq n$, $\quad 2.2 : \dfrac{\varepsilon \mid Y'_j}{Z \mid Y''_j}$ $\quad 1 \leq j \leq n-1$,

$2.3 : \dfrac{X''_j \mid \varepsilon}{X_j \mid Z}$ $\quad 1 \leq j \leq n-1$, $\quad 2.4 : \dfrac{X' \mid \varepsilon}{X \mid Z}$ $\quad 2.5 : \dfrac{X' \mid \varepsilon}{\varepsilon \mid Z'}$

Component R_3.

$3.1 : \dfrac{\varepsilon \mid Y_j}{Z \mid Y'_{j-1}}$ $\quad 2 \leq j \leq n$, $\quad 3.2 : \dfrac{X'_j \mid \varepsilon}{X''_j \mid Z}$ $\quad 1 \leq j \leq n-1$,

$3.3 : \dfrac{\varepsilon \mid Y_1}{Z \mid Y'}$ $\quad 3.4 : \dfrac{\varepsilon \mid Y'}{Z \mid Y}$ $\quad 3.5 : \dfrac{\varepsilon \mid BY'}{Z'' \mid \varepsilon}$

Note.

Components R_1, R_2 and R_3 also contain the following rules:

$\dfrac{\alpha \mid \varepsilon}{\alpha \mid \varepsilon}$ $\quad$ for each axiom $\alpha \in A$, except $XSBY$.

Details.

Our simulation is based on the method of rotating words [10] and the corresponding technique that are proposed by G.Paun [13]. Let us recall them briefly.

For any word $w = w'w'' \in (N \cup T)^*$ of formal grammar G say that word $Xw''Bw'Y$ $(X, Y, B \notin N \cup T)$ of TVDH system D_G is a "rotation version" of word w. Say also that going from a rotation version of the word to another one is rotating the word. $TVDH$ system D_G models the formal grammar G as follows. System D_G rotates the word Xw_1uw_2BY into Xw_2Bw_1uY. And so rotating is used to put the occurrence of u to the end of the word. After that D_G applies a splicing rule $\frac{\varepsilon \mid uY}{Z \mid vY_i}$. So we model the application of a rule $u \to va_i$ of a formal grammar G by a single scheme-rule. $TVDH$ system D_G rotates the word Xwa_iY $(a_i \in (N \cup T \cup \{B\})$ "symbol by symbol", i.e. word Xa_iwY is obtained after performing a few steps of system D_G. Rotating a word can be done as follows. We start with word Xwa_iY in component R_1. Component R_2 receives the word XwY_i from component R_1. Component R_3 receives words $X_ja_jwY_i$ $(1 \le j \le n)$ from component R_2. After that point system D_G works in a cycle in which indexes i and j decrease. If $j \ne i$, then derived words with these indexes will be ruled out. When $i = j$ we obtain word $X_1a_iwY_1$. Then, from that word we obtain word Xa_iwY (so we rotated the word Xwa_iY) and word a_iw' (if $a_iw = a_iw'B$). So, if $a_iw' \in T^*$ then $a_iw' \in L(G)$ and $a_iw' \in L(D_G)$.

(i) $L(G) \subseteq L(D_G)$.

We start with word $XwY = Xw'a_iY$ $(w \in (N \cup T \cup \{B\})^*,\ 1 \le i \le n)$.

$(Xw|uY, Z|vY_i) \vdash_{1.1} (XwvY_i, ZuY),\ i \in \{1, \dots, n-1\}$.

String ZuY cannot enter a splicing in R_2 and therefore is removed. So we model the application of rule $u \to va_i$.

$(Xw'|a_iuY, Z|Y_i) \vdash_{1.6} (Xw'Y_i, Za_iuY),\ 1 \le i \le n$.

String Za_iuY cannot enter a splicing in R_2 and therefore is removed. So we model the application of rule $u \to \varepsilon$.

$(Xw'|a_iY, Z|Y_i) \vdash_{1.2} (Xw'Y_i, Za_iY),\ 1 \le i \le n$.

String Za_iY cannot enter a splicing in R_2 and therefore is removed. So we start to rotate the word $w'a_i$.

$(X|w'Y_i, X_ja_j|Z) \vdash_{2.1} (\mathbf{X_ja_jw'Y_i}, XZ),\ 1 \le i, j \le n$.

String XZ is an axiom.

$(X_ja_jw'|Y_i, Z|Y'_{i-1}) \vdash_{3.1} (X_ja_jw'Y'_{i-1}, ZY_i),\ 1 \le j \le n,\ 2 \le i \le n$.

String ZY_i is an axiom.

$(X_j|a_jw'Y'_{i-1}, X'_{j-1}|Z) \vdash_{1.3} (X'_{j-1}a_jw'Y'_{i-1}, X_jZ),\ 2 \le i, j \le n$.

String X_jZ is an axiom.

$(X'_{j-1}a_jw'|Y'_{i-1}, Z|Y''_{i-1}) \vdash_{2.2} (X'_{j-1}a_jw'Y''_{i-1}, ZY'_{i-1}),\ 2 \le i, j \le n$.

String ZY'_{i-1} is an axiom.

$(X'_{j-1}|a_jw'Y''_{i-1}, X''_{j-1}|Z) \vdash_{3.2} (X''_{j-1}a_jw'Y''_{i-1}, X'_{j-1}Z),\ 2 \le i, j \le n$.

String $X'_{j-1}Z$ is an axiom.

$(X''_{j-1}a_jw'|Y''_{i-1}, Z|Y_{i-1}) \vdash_{1.5} (X''_{j-1}a_jw'Y_{i-1}, ZY''_{i-1}),\ 2 \le i, j \le n$.

String ZY''_{i-1} is an axiom.

$(X''_{j-1}|a_jw'Y_{i-1}, X_{j-1}|Z) \vdash_{2.3} (\mathbf{X_{j-1}a_jw'Y_{i-1}}, X''_{j-1}Z),\ 2 \le i, j \le n$.

String $X''_{j-1}Z$ is an axiom.

And so on.

There are 3 cases:

1) $\mathbf{X_1 a_j w' Y_k}$ 2) $\mathbf{X_k a_j w' Y_1}$, $2 \leq k \leq n$, and 3) $\mathbf{X_1 a_i w' Y_1}$.

Case 1):

$(X_1 a_j w'|Y_k, Z|Y'_{k-1}) \vdash_{3.1} (X_1 a_j w' Y'_{k-1}, ZY_k)$, $2 \leq k \leq n$.
String ZY_k is an axiom.

$(X_1|a_j w' Y'_{k-1}, X'|Z) \vdash_{1.4} (X' a_j w' Y'_{k-1}, X_1 Z)$, $2 \leq k \leq n$.
String $X_1 Z$ is an axiom.

$(X' a_j w'|Y'_{k-1}, Z|Y''_{k-1}) \vdash_{2.2} (X' a_j w' Y''_{k-1}, ZY'_{k-1})$, $2 \leq k \leq n$.
String ZY'_{k-1} is an axiom. String $X' a_j w' Y''_{k-1}$ cannot enter a splicing in R_3 and therefore is removed.

$(X'|a_j w' Y'_{k-1}, X|Z) \vdash_{2.4} (X a_j w' Y'_{k-1}, X'Z)$, $2 \leq k \leq n$.
String $X'Z$ is an axiom. String $X a_j w' Y'_{k-1}$ cannot enter a splicing in R_3 and therefore is removed.

$(X'|a_j w' Y'_{k-1}, |Z') \vdash_{2.5} (a_j w' Y'_{k-1}, X'Z')$, $2 \leq k \leq n$.
Strings $a_j w' Y'_{k-1}$ and $X'Z'$ cannot enter a splicing in R_3 and therefore are removed.

So, this computation is "mortal", i.e. it completes without results.

Case 2):

$(X_k a_j w'|Y_1, Z|Y') \vdash_{3.3} (X_k a_j w' Y', ZY_1)$, $2 \leq k \leq n$.
String ZY_1 is an axiom.

$(X_k|a_j w' Y', X'_{k-1}|Z) \vdash_{1.3} (X'_{k-1} a_j w' Y', X_k Z)$, $2 \leq k \leq n$.
String $X_k Z$ is an axiom.
String $X'_{k-1} a_j w' Y'$ cannot enter a splicing in R_2 and therefore is removed.

So, this computation also is "mortal".

Case 3):

$(X_1 a_i w'|Y_1, Z|Y') \vdash_{3.3} (X_1 a_i w' Y', ZY_1)$, $1 \leq i \leq n$.
String ZY_1 is an axiom.

$(X_1|a_i w' Y', X'|Z) \vdash_{1.4} (X' a_i w' Y', X_1 Z)$, $1 \leq i \leq n$.
String $X_1 Z$ is an axiom.

$(X'|a_i w' Y', X|Z) \vdash_{2.4} (X a_i w' Y', X'Z)$, $1 \leq i \leq n$.
String $X'Z$ is an axiom.

$(X'|a_i w' Y', |Z') \vdash_{2.5} (a_i w' Y', X'Z')$, $1 \leq i \leq n$.
String $X'Z'$ cannot enter a splicing in R_3 and therefore is removed.

$(X a_i w'|Y', Z|Y) \vdash_{3.4} (X a_i w' Y, ZY')$, $1 \leq i \leq n$.
String ZY' is an axiom.

And now, $X a_i w' Y$ is the expected result. We rotated symbol a_i from the right side of the developed word to the left side one.

$(Xa_iw''|BY', Z''|) \vdash_{3.5} (Xa_iw'', Z''BY')$, $1 \le i \le n$.
Strings Xa_iw'' and $Z''BY'$ cannot enter a splicing in R_1 and therefore are removed.

$(a_iw''|BY', Z''|) \vdash_{3.5} (a_iw'', Z''BY')$, $1 \le i \le n$.
String $Z''BY'$ cannot enter a splicing in R_1 and therefore is removed.

And now, a_iw'' may be the expected result (a pure terminal string). If $a_iw'' \in T^*$ and so $a_iw'' \in L(G)$, then $a_iw'' \in L(D_G)$ also and this part of theorem 1 is proved.

D_G continues its computations.
$(a_iw'|Y', Z|Y) \vdash_{3.4} (a_iw'Y, ZY')$, $1 \le i \le n$.
String ZY' is an axiom.

Rule 1.1 may be applied:
$(a_iw''|uY, Z|vY_k) \vdash_{1.1} (a_iw''vY_k, ZuY)$, $1 \le i \le n$, $k \in \{1, \ldots, n-1\}$.
Strings $a_iw''vY_k$ and ZuY cannot enter a splicing in R_2 and therefore are removed.

Rule 1.6 may be applied:
$(a_iw''|a_luY, Z|Y_l) \vdash_{1.6} (a_iw''Y_l, Za_luY)$, $1 \le i, l \le n$.
Strings $a_iw''Y_l$ and Za_luY cannot enter a splicing in R_2 and therefore are removed.
$(a_iw''|a_lY, Z|Y_l) \vdash_{1.2} (a_iw''Y_l, Za_lY)$, $1 \le i, l \le n$.
Strings $a_iw''Y_l$ and Za_lY cannot enter a splicing in R_2 and therefore are removed.

So, these computations are "mortal" too and D_G cannot produce any other molecules than a_iw''.

(ii) $L(D_G) \subseteq L(G)$.

Let us examine the behaviour of D_G. First of all, we can see that D_G correctly simulates the use of any rule $u \to v \in P$, because all "superfluous" molecules are ruled out. We note that the subscripts of X, Y can be removed only when $i = j$ during the simulation of rule $u \to v \in P$. More generally, occurring nonterminal letters can be removed to obtain the "pure" terminal word only when we simulates the rule $u \to v \in P$. So, if $w \in L(D_G)$ then $w \in L(G)$. □

4 Modified Definition of TVDH Systems – Extended TVDH Systems

We extend the definition of H system to components of TVDH system, so the components $R_i, 1 \le i \le n$ of TVDH system of degree n will work as corresponding H systems.

Let us define operation $\tilde{\sigma}_h(L)$ as follows:

$\tilde{\sigma}_h(L) = \tilde{\sigma}_h(L' \cup L'') \stackrel{\text{def}}{=} \sigma_h^*(L')$, 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, $\tilde{\sigma}_h(L)$ works as H system $(h, L') = ((V, R), L')$.

An extended time-varying distributed H *system* (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*, and 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 $i \equiv k (mod\ n),\ k \geq 1, 1 \leq i \leq n, h_i = (V, R_i)$.

The language generated by E is:

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

Let us denote by $EVDH_n,\ n \geq 1$, the family of languages generated by extended time-varying distributed H systems of degree at most n, and $EVDH_*$ the family of all languages of this type.

Theorem 2. (see [8].) *a)* $EVDH_1 = REG$*, b)* $EVDH_2 = EVDH_* = RE$*.*

Proof. Let be $E = (V, T, A, R_1)$. As there is a single component, we can denote it simply R and, accordingly, $\tilde{\sigma}_{h_1}$ can also be denoted by $\tilde{\sigma}$ and $\sigma^*_{h_1}$ by σ^*. According to the definitions, $L_1 = A$. We have that $L_2 = \tilde{\sigma}(A)$.

Let us now split L_2 into two subsets: L'_2 will denote the words which enter the rules of the component and L''_2, will denote those which do not enter the rules. Accordingly, $L_3 = \tilde{\sigma}(L_2) = \sigma^*(L'_2)$. By construction, we have that $L'_2 \subseteq L_2$. Split L_3 into two subsets L'_3 and L''_3 defined in the same way as for L_2. We have that $L_3 = \tilde{\sigma}(L_2) = \sigma^*(L'_2)$ and, as $L'_2 \subseteq L_2$, that $L'_3 \subseteq L'_2$.

Now, $L_4 = \tilde{\sigma}(L_3) = \sigma^*(L'_3) \subseteq \sigma^*(\sigma^*(L'_2))$. But, by the definition of σ^*, it is plain that $\sigma^*(\sigma^*(L)) = \sigma^*(L)$ for any language L. Accordingly, $L_4 \subseteq L_3$. And so, due to the definition of $L(E)$, $L(E) = (L_1 \cup L_2 \cup L_3) \cap T^*$.

By [15], we know that extended H systems with a finite set of axioms and a finite set of rules generate regular languages. Accordingly, L_2 is a regular language. For each rule r of R, let us define L_r as the set of words upon V^* which do not enter r. As it is plain that the complement of L_r in V^* is recognizable by a finite automaton, L_r is a regular language and so, as R is finite and as $L''_2 = \bigcup_{r \in R}(L_2 \cap L_r)$, L''_2 is also a regular language. As L'_2 is the complement of L''_2 in L_2 which is also a regular language, we obtain that L'_2 is a regular language. By [11], extended H systems with a regular set of axioms and with a finite number of rules generate regular languages. Accordingly, L_3 is a regular language, and so is $L(E)$. This proves that $L(E) \subseteq REG$. Similar as [13], it is easy to prove that $REG \subset EVDH_1$, so the proof of theorem 2 *a*) is completed.

The part b) is true immediately after a small modification of the construction of theorem 1b [8]. $\square$

Acknowledgement. The authors acknowledge the very helpful contribution of *INTAS project* 97-1259 for enhancing their cooperation, giving the best conditions for producing the present result. For the same reasons they also acknowledge the help of the University of Metz (France).

References

1. Csuhaj-Varjù, E., Kari, L., Păun, G.: Test Tube distributed system based on splicing. Computer and AI. 2–3 (1996) 211–232
2. Ferretti, C., Mauri, G., Zandron, C.: Nine test tubes generate any RE language. TCS. **231**, no.2 (2000) 171–180
3. Head, T.: Formal language theory and DNA: an analysis of the generative capacity of recombinant behaviors. Bulletin of Mathematical Biology **49** (1987) 737–759
4. 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)
5. Margenstern, M.: Frontier between decidability and undecidability: a survey. TCS. **231**, no.2 (2000) 217–251
6. Margenstern, M., Rogozhin, Yu.: A universal time-varying distributed H-system of degree 2. In Preliminary proceedings, Fourth International Meeting on DNA Based Computers, June 15-19, 1998, University of Pennsylvania (1998) 83–84
7. Margenstern, M., Rogozhin, Yu.: A universal time-varying distributed H-system of degree 2. Biosystems **52** (1999) 73–80
8. Margenstern, M., Rogozhin, Yu.: Generating all recursively enumerable languages with a time-varying distributed H-system of degree 2. Publications du GIFM, no. 99-102, I.U.T., Metz (France), ISBN 2-9511539-5-3 (1999)
9. Păun, A.: On Time-Varying H Systems. Bulletin of EATCS. **67** (1999) 157–164
10. Păun, G.: Regular extended H systems are computationally universal. Journal of Automata, Languages and Combinatorics **1**, no.1 (1996) 27–36
11. Păun, G., Rozenberg, G., Salomaa, A.: Computing by splicing. TCS. **168**, no.2 (1996) 321–336
12. Păun, G.: DNA computing: distributed splicing systems. In *Structures in Logic and Computer Science. A Selection of Essays in honor of A. Ehrenfeucht*, LNCS. **1261** (1997) 353–370
13. Păun, G.: DNA Computing Based on Splicing: Universality Results. TCS. **231**, no.2 (2000) 275–296
14. Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing: New Computing Paradigms. Springer, Heidelberg (1998)
15. Pixton, D.: Regularity of splicing languages. Descrete Applied Mathematics **69** (1996) 101–124
16. Priese, L., Rogozhin, Yu., Margenstern, M.: Finite H-Systems with 3 Test Tubes are not Predictable. In Proceedings of Pacific Symposium on Biocomputing, Kapalua, Maui, January 1998 (R.B.Altman, A.K.Dunker, L.Hunter, T.E.Klein, eds), World Sci. Publ., Singapure (1998) 545–556
17. Rogozhin, Yu.: Small universal Turing machines. TCS. **168**, no.2 (1996) 215–240
18. Rogozhin, Yu.: A Universal Turing Machine with 22 States and 2 Symbols. Romanian Journal of Information Science and Technology **1**, no.3 (1998) 259–265
19. Verlan, S.: On extended time-varying distributed H systems. Poster at DNA6 conference, Leiden (2000)