Time-varying distributed H systems of degree 1 generate all recursively enumerable languages

Maurice Margenstern[1] and Yurii Rogozhin[2]

[1] Institut Universitaire de Technologie,
Université de Metz, Metz, France
Laboratoire d'Informatique Théorique et Appliquée
margens@lita.univ-metz.fr

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

Abstract. Time-varying distributed H system (TVDH system shortly) of degree n is a well known model of splicing computations which has the following special feature: at different moments one uses different sets of splicing rules (the number of these sets of splicing rules is called the degree of the TVDH system). It is known that there is a universal TVDH system of degree 1. Now we prove that TVDH systems of degree 1 generate all recursively enumerable languages.

1 Introduction

Time-varying distributed H system are well known theoretical model of biocomputing based on splicing operations [18–20]. We refer the reader to [6–11, 16, 24, 12] for more details on the history of this model, its extensions and connections with previous models based on splicing computations. That model introduces *components*, later see the formal definition, which cannot all be used at the same time but one after another, periodically.

In [19] it is proved that 7 different *components* are enough in order to generate any recursively enumerable language. In [17], the number of components was reduced down to 4. Recently, see [12], both authors proved that one component is enough to construct a *universal* time-varying distributed H system, *i.e.* to construct a time-varying distributed H system, that is capable to simulate the computation of any Turing machine (for instance any universal Turing machine from [21–23]). 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 n components entails that there are time-varying distributed H systems which generate all recursively enumerable languages, with only n components. It is true for two components, because both authors proved that 2 different components are enough in order to generate any recursively enumerable language [11].

Now we complete the proof that it is true also for one component, i.e. that TVDH systems of degree 1 can generate any recursively enumerable language.

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 concatenation 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).

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 $m_1 = w_1u_1u_2w_2$ and $m_2 = w_1'u_1'u_2'w_2'$, and produces two new molecules $m_1' = w_1u_1u_2'w_2'$ and $m_2' = w_1'u_1'u_2w_2$. In this case, we also write

$$(w_1u_1|u_2w_2,\ w_1'u_1'|u_2'w_2') \vdash_r (w_1u_1u_2'w_2',\ w_1'u_1'u_2w_2)$$

or simply $(m_1, m_2) \vdash_r (m_1', m_2')$.

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* [1,2], 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 the H system H is defined as $L(H) \stackrel{\text{def}}{=} \sigma_h^*(A)$.

Thus, the language generated by the 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, $n \geq 1$), 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 the component R_i is used for splicing the currently available strings. Specifically, we define

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

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 rule are removed.

The language generated by D is, by definition:

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

Denote 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 Main result

Theorem 1. *$VDH_1 = VDH_* = RE$.*

Proof. We shall consider recursively enumerable sets of natural numbers instead of recursively enumerable languages. There are many well known methods to encode any formal languages into subsets of natural numbers. It will make our proof simpler.

It is a well known fact from theoretical computer science that for every recursively enumerable set $\mathcal{L}$ of natural numbers, there is a Turing machine $T_\mathcal{L}$ which generates $\mathcal{L}$ as follows. Let $f_\mathcal{L}$ be a total recursive function enumerating the elements of $\mathcal{L}$, *i.e.* $\mathcal{L} = \{f_\mathcal{L}(x)\ ;\ x \in I\!N\}$. It is well known, see [3], chap.XIII, § 68, Theorem XXVIII, how to construct a Turing machine $T_\mathcal{L}$ from the definition of $f_\mathcal{L}$ which would compute what f does, *i.e.* for every natural number x, machine $T_\mathcal{L}$ transforms any initial configuration $q_1 0 1^{x+1}$ into the final configuration $q_0 0 1^{f_\mathcal{L}(x)+1} 0 \ldots 0$. And more, machine $T_\mathcal{L}$ never visits cells on the left at the cell with 0 of the initial configuration (i.e. machine $T_\mathcal{L}$ has a tape that is semiinfinite to right) and no instruction of machine $T_\mathcal{L}$ is stationary, i.e. the head moves on every steps of the computation.

Let us assume for a while that we succeeded to simulate machine $T_\mathcal{L}$. It is not yet enough to prove the theorem. Indeed, starting from one word, say $q_1 0 1^{x+1}$ for some x, we would obtain an encoding of $f_\mathcal{L}(x)$ as $q_0 0 1^{f_\mathcal{L}(x)+1}$. But this is not what we need, because this gives us a single word.

It is not difficult to see that starting from machine $T_\mathcal{L}$, there is a Turing machine $T'_\mathcal{L}$ which computes $0 1^{x+2} q_0 0 1^{f_\mathcal{L}(x)+1} 0 \ldots 0$, starting from $q_1 0 1^{x+1}$. It will be possible for us to device a TVDH system of degree 1 Γ which, arriving to that stage of the computation, will split the obtained word into two parts: $1^{f_\mathcal{L}(x)+1}$ as a *generated* word and $q_1 0 1^{x+2}$ as a new starting configuration. This will allow us to obtain exactly $\mathcal{L}$ as $\mathcal{L}(\Gamma)$.

4 Details

In what follows, in order to facilitate the reading of the formulas, we shall write $f(x)$ instead of $f_{\mathcal{L}}(x)$ when it will be clear from the context that we mean that latter function.

Definition of Γ

And so, we assume that $\{0,1\}$ is the tape alphabet of Turing machine $T'_{\mathcal{L}}$ and let us number its states by $q_0, q_1, \ldots, q_n$, where q_1 is the initial state and q_0 the final one. From now on, we shall use latin small letters to denote tape symbols, greek letters for words on the alphabet of the tape symbols, bold latin small letters for states of the Turing machine.

Define TVDH system Γ as follows:

Its terminal alphabet is $T = \{1\}$. Let us introduce variables for $q_0, q_1, \ldots, q_n$, namely $\mathbf{a}$ and $\mathbf{b}$ with that additional assumption that $\mathbf{b}$ ranges over all q_i's but $\mathbf{a}$ ranges over all of them except q_0.

Denoting by $\mathcal{I}$ the set of instructions of $T'_{\mathcal{L}}$, the expressions $\mathbf{a},\, x \in \mathcal{I}$ as well as $\mathbf{a},\, x, \mathbf{b},\, y \in \mathcal{I}$ will be used for indicating an instruction by its constituting elements. In particular, we will use quantifiers with such expressions. Moreover, we shall specialise $\mathcal{I}$ by denoting by $\mathcal{I}_R$ the set of all instructions with a move to right, by $\mathcal{I}_L$ the set of all instructions with a move to left.

Define alphabet V by setting

$V = \{0,1\} \cup \{X, Y, Y', \mathbf{b}, t_{q0},\ t_{r1},\ t_{r2},\ t_{l1},\ t_{l2},\ R,\ L,\ Q,\ Z,\ \rightarrow,\ \leftarrow\}$, with $\mathbf{b}$ taking all their possible values.

Axioms A are given by

$A = \{Xq_101Y,\ Z\mathbf{a}0Y,\ Ly\mathbf{b}Y,\ Ly\mathbf{b}zR,\ L\mathbf{b}zyR,\ RQ \rightarrow t_{q0},\ t_{r1}1 \rightarrow t_{r2},\ Z \leftarrow 1,\ t_{l1} \leftarrow 1t_{l2},\ Y',\ \ \leftarrow Y,\ Xq_10Z\ \}$, with $\mathbf{a}, \mathbf{b}$ taking all their possible values, and $s, y, z \in \{0,1\}$.

Block "Right end of the tape"

$1.1 : \dfrac{st \,|\, \mathbf{a}Y}{Z \,|\, \mathbf{a}0Y}$ $\qquad \forall s \in \{0,1,X\};\ \forall \mathbf{a};$

$\forall t \in \{0,1\};$

Block "Right shift"

$2.1 : \dfrac{s \,|\, \mathbf{a}\, xz}{L \,|\, y\, \mathbf{b}\, zR}$ $\qquad \forall\, s \in \{0,1,X\};$

$\forall z \in \{0,1\};$
$\forall\, \mathbf{a}, x, y, \mathbf{b} \in \mathcal{I}_R;$

$$2.1' : \begin{array}{r|l} s & \mathbf{a}\,xY \\ \hline L & y\,\mathbf{b}\,Y \end{array} \qquad \forall\, s \in \{0,1,X\}; \quad \forall\, \mathbf{a}, x, y, \mathbf{b} \in \mathcal{I}_R;$$

$$2.2 : \begin{array}{r|l} sy\mathbf{b}\,z & R \\ \hline L\mathbf{a}\,x\,z & t \end{array} \qquad \forall\, s \in \{0,1,X\}; \quad \forall\, z \in \{0,1\}; \quad \forall\, t \in \{0,1,Y\}; \quad \forall\, \mathbf{a}, x, y, \mathbf{b} \in \mathcal{I}_R;$$

Block "Left shift"

$$3.1 : \begin{array}{r|l} s & z\mathbf{a}\,xpw \\ \hline L & \mathbf{b}\,zyR \end{array} \qquad \forall\, s \in \{0,1,X\}; \quad \forall\, p, z \in \{0,1\}; \quad \forall\, w \in \{0,1,Y\}; \quad \forall\, \mathbf{a}, x, y, \mathbf{b} \in \mathcal{I}_L;$$

$$3.1' : \begin{array}{r|l} s & z\mathbf{a}\,xY \\ \hline L & \mathbf{b}\,zyR \end{array} \qquad \forall\, s \in \{0,1,X\}; \quad \forall\, z \in \{0,1\}; \quad \forall\, \mathbf{a}, x, y, \mathbf{b} \in \mathcal{I}_L;$$

$$3.2 : \begin{array}{r|l} s\mathbf{b}zy & R \\ \hline Lz\mathbf{a}x & p \end{array} \qquad \forall\, s \in \{0,1,X\}; \quad \forall\, p \in \{0,1,Y\}; \quad \forall\, z \in \{0,1\}; \quad \forall\, \mathbf{a}, x, y, \mathbf{b} \in \mathcal{I}_L;$$

Block "Right signal"

$$4.1 : \begin{array}{r|l} 1 & q_0 01 \\ \hline R & Q \rightarrow t_{q0} \end{array}$$

$$4.2 : \begin{array}{r|l} 1Q \rightarrow & t_{q0} \\ \hline Rq_0 0 & \end{array}$$

$$4.3 : \begin{array}{r|l} s & \rightarrow 1 \\ \hline t_{r1} & 1 \rightarrow t_{r2} \end{array} \qquad \forall s \in \{1,Q\};$$

$$4.4 : \begin{array}{r|l} s1 \rightarrow & t_{r2} \\ \hline t_{r1} \rightarrow 1 & \end{array} \qquad \forall s \in \{1,Q\};$$

Block "Left signal"

$$5.1 : \begin{array}{r|l} s & 1 \rightarrow p \\ \hline Z & \leftarrow 1 \end{array} \qquad \forall s \in \{1,Q\}; \forall p \in \{0,Y\};$$

$$5.2 : \begin{array}{r|l} 1 \leftarrow & p \\ \hline t_{l1} \leftarrow 1 & t_{l2} \end{array} \qquad \forall\, p \in \{1,Y\};$$

$$5.3 : \begin{array}{r|l} & 1 \leftarrow t_{l2} \\ \hline t_{l1} & \leftarrow 1p \end{array} \qquad \forall\, p \in \{1,Y\};$$

Block "Result"

$$6.1:\ \begin{array}{r|l} Q\leftarrow & 1 \\ \hline & Y' \end{array}$$

$$6.2:\ \begin{array}{r|l} 1 & Q\leftarrow Y' \\ \hline & \leftarrow Y \end{array}$$

$$6.3:\ \begin{array}{r|l} X0\leftarrow & 1 \\ \hline Xq_1 0 & Z \end{array}$$

The following rules are also contained:

$\begin{array}{r|l} \alpha & \\ \hline \alpha & \end{array}$ for each axiom $\alpha \in A$, except $Xq_1 01Y$. As it can be seen from the computations, these axioms must always be present for making it possible to apply most of the rules.

5 Checking the simulation

5.1 Checking the simulation of one step in a Turing computation

The first thing we have now to check is that starting from a word representing the current configuration of machine $T'_{\mathcal{L}}$, the application of the rules of Γ leads to a word representing the next configuration. In other words, given any instruction of the program of $T'_{\mathcal{L}}$, we have to show that there are rules in Γ and axioms allowing to mimic the application of that instruction to a suitable configuration.

There are several cases according to the move of the instruction and of the position of the scanned cell in the configuration: according to whether the move is to right or to left, according to whether the scanned cell is an interior one or a cell being at the right end of the configuration.

Case of an interior cell

Let ω_ℓ, ω_r be finite words of alphabet $\{0,1\}$ and $\omega_\ell\,\mathbf{a}\,x\,\omega_r$ be the current configuration of machine $T'_{\mathcal{L}}$ with $\omega_\ell = \nu_\ell\, sz$ and $\omega_r = p\,\nu_r$. Assume that Γ has obtained word $X\,\nu_\ell\, sz\,\mathbf{a}\,x\,p\,\nu_r\,Y$ at a time, say t. Assume that instruction $\boxed{\mathbf{a}\,x\,L\,y\,\mathbf{b}}$ applies to the current configuration. Possibly after several steps, Γ should produce word $X\,\nu_\ell\, s\mathbf{b}\,z\,y\;p\nu_r\,Y$. The reader can get convinced that in two steps Γ produces the expected word and no undesirable one: indeed, rule 3.1 (3.1') applies as far as the axiom $L\,\mathbf{b}\,z\,y\,R$ is present, leading to words $X\,\nu_\ell\,s\,\mathbf{b}\,zyR$ and $L\,z\mathbf{a}\,x\,p\nu_r\,Y$ that can enter in rule 3.2 and, therefore, are kept for the further processing. Now, rule 3.2 straightforward applies to these words and provides us with $X\,\nu_\ell\,s\mathbf{b}\,z\,y\,p\,\nu_r\,Y$ and $L\,z\,\mathbf{a}\,xR$. Observe now that $L\,z\,\mathbf{a}\,xR$ is an axiom. On another hand, word $X\,\nu_\ell\,s\mathbf{b}\,z\,y\,p\,\nu_r\,Y$ is the representation of the next configuration and we see that the previous conditions are again observed, which proves the simulation of one step of Turing computation by Γ, at least when an instruction with a move to left is applied.

The reader can now check that in the case of an instruction $\mathbf{a}\,x\,R\,y\,\mathbf{b}$ with a move to right, rules 2.1 (2.1') and 2.2 successively apply, providing us with the expected next configuration, namely $X\,\omega_\ell\,y\,\mathbf{b}\,\omega_r\,Y$ (where ω_r may be empty).

Case of the right end of the tape

We have now to check what happens when the head has to go to the right of the right end of the configuration. As in our representation of a configuration the ends are realised by special markers, X for the left end, Y for the right one, such cases are easily characterised: $X\,\mathbf{a}\,x\,\nu\,Y$ for the left end and $X\,\omega\,\mathbf{a}\,xY$ for the right one.

In the case of the left end, the case is dealt with by the already seen rules 2.1 (2.1'), 2.2 if the instruction determined by $\mathbf{a}\,x$ has a move to right.

In the case of the right end there is a uniform way for solving that case: it is enough to transform representation $X\,\omega\,\mathbf{a}\,Y$ into the following one: $X\,\omega\,\mathbf{a}\,0\,Y$, which can be dealt with as the case of an interior cell. It is easy to check that rule 1.1 allows to obtain the just described transformation.

5.2 Checking how to resume the computation on $\mathbf{01}^{x+2}$

We now arrive at an important point of our simulation.

From what we already seen, we now prove that Γ transforms $X\,q_1\,0\,1^{x+1}\,Y$ into $X\,0\,1^{x+2}\,q_0\,0\,1^{f(x)+1}\,0\,\ldots\,0\,Y$. Let us write $\omega_m = 1^{f(m)+1}$, where f enumerates $\mathcal{L}$, and inductively define $\mathcal{L}_m$ by

$$\begin{aligned}\mathcal{L}_0 &= \{\omega_0\}\\ \mathcal{L}_{k+1} &= \mathcal{L}_k \cup \{\omega_{k+1}\}.\end{aligned}$$

It will be enough to check that if at some time t_k we have obtained $\mathcal{L}_k$ and $X\,q_1\,0\,1^{k+2}\,Y$, then at a later time t_{k+1} we shall obtain $\mathcal{L}_{k+1}$ and $X\,q_1\,0\,1^{k+3}\,Y$. And so, starting from $X\,0\,1^{k+3}\,q_0\,0\,1^{f(k+1)+1}\,0\,\ldots\,0\,Y$, we leave a marker, Q at the place of $q_0 0$ and, at the same time, send a signal searching the first 0 following the 0 scanned by the head at that time. When the wanted 0 is found, it cuts off the superfluous 0's and Y and the signal goes back to the marker. When it reaches it, it splits the word in two parts: the right one is the expected ω_{k+1}; in the left one, the marker is changed into Y and the signal goes on to left until it meets 0. At that moment it transforms what is on the left of that $X0$ into $Xq_1 0$, and the new expected starting configuration is reached. So, we explain that in details.

Starting from word $\boxed{X\,0\,1^{k+3}\,q_0\,0\,1^{f(k+1)+1}\,0\,\ldots\,0\,Y}$, the application of rules 4.1 and 4.2 allows to put marker Q on the place of q_0, to erase $q_0 0$ and to insert in front of $1^{f(k+1)+1}$ signal $\rightarrow$: indeed, the application of rule 4.1 yields $X\,0\,1^{k+3}\,Q \rightarrow t_{q0}$ and $Rq_0\,0\,1^{f(k+1)+1}\,0\,\ldots\,0\,Y$. Both words are kept as far as they enter rule 4.2 which in its turn yields word

$\boxed{X\,0\,1^{k+3}\,Q \rightarrow 1^{f(k+1)+1}\,0\,\ldots\,0\,Y}$, as well as $R\,q_0\,0t_{q0}$ which enter no rule of Γ and, consequently, is ruled out. Then, rules 4.3 and 4.4 allow to cross over the block of 1's constituting the expected result ω_{k+1}. Indeed, assuming that the computation starts from word $X\,0\,1^{k+3}\,Q\,1^u \rightarrow 11^v\,0\,\ldots\,0\,Y$, rule 4.3 yields $X\,0\,1^{k+3}\,Q\,1^u 1 \rightarrow t_{r2}$ and $t_{r1} \rightarrow 1\,1^v\,0\,\ldots\,0\,Y$ which enter rule 4.4 yielding $X\,0\,1^{k+3}\,Q\,1^u 1 \rightarrow\ 1^v\,0\,\ldots\,0\,Y$ and $t_{r1} \rightarrow 1\ t_{r2}$ which is ruled out, as far as it enters no rule of Γ. And so, by induction on the number of 1's to be crossed over, we see that the appropriate iteration of rules 4.3 and 4.4 leads to word $\boxed{X\,0\,1^{k+3}\,Q\,1^{f(k+1)+1} \rightarrow 0\,\ldots\,0\,Y}$. But now, rule 5.1 allows to throw away the superfluous substring $0\,\ldots\,0\,Y$ and to reverse the signal, leading to word $\boxed{X\,0\,1^{k+3}\,Q\,1^{f(k+1)} \leftarrow 1}$. Now repeated applications of rule 5.2 and 5.3 allow to arrive to word $\boxed{X\,0\,1^{k+3}\,Q\ \leftarrow\ 1^{f(k+1)+1}}$. Indeed, starting from $X\,0\,1^{k+3}\,Q\,1^u\,1 \leftarrow\ 1^v$, rule 5.2 yields $X\,0\,1^{k+3}\,Q\,1^u\,1 \leftarrow t_{l2}$ and $t_{l1} \leftarrow 1\,1^v$, both word entering rule 5.3 yielding $X\,0\,1^{k+3}\,Q\,1^u \leftarrow 1\,1^v$ and $t_{l1}1 \leftarrow t_{l2}$, the latter being ruled out as far as it enters no rule of Γ.

Now, whatever is the value of $f(k{+}1)$, rule 6.1 and 6.2 apply and allow to separate the result as a word in the terminal alphabet, namely ω_{k+1} from the other part, transformed into word $\boxed{X\,0\,1^{k+3}\ \leftarrow\ Y}$. Indeed, applied to word $X\,0\,1^{k+3}\,Q\ \leftarrow\ 1^{f(k+1)+1}$ rule 6.1 yields $X\,0\,1^{k+3}\,Q \leftarrow Y'$ and $\boxed{1^{f(k+1)+1}}$. The latter word is the expected ω_{k+1} and the computation goes on with the former word, and it enters rule 6.2 allowing to obtain $\boxed{X\,0\,1^{k+3}\ \leftarrow\ Y}$ and also $Q \leftarrow Y'$ which is ruled out, as entering no rule of Γ. We are now reaching the last stage of the computation leading to $\boxed{X\,q_1\,0\,1^{k+3}\,Y}$.

Starting from word $X\,0\,1^{k+3}\ \leftarrow\ Y$, rules 5.2 and 5.3 again apply, allowing after an appropriate iteration to reach word $X\,0\ \leftarrow\ 1^{k+3}\,Y$, to which rule 5.3 applies, yielding the needed configuration $\boxed{X\,q_1\,0\,1^{k+3}\,Y}$ together with $X\,0 \leftarrow Z$ which is ruled out as entering no rule of Γ.

And so we proved our claim about turning from $\mathcal{L}_k$ to $\mathcal{L}_{k+1}$.

Acknowledgement. The authors acknowledge the very helpful contribution of *INTAS project* 97-1259 and *NATO project* PST.CLG.976912 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), INRIA Lorraine and the French Ministry of Education and Research.

References

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

3. Kleene, S.: Introduction to Metamathematics. Van Nostrand Comp. Inc., New-York, 1952
4. Margenstern, M.: Non-erasing Turing machines: a new frontier between a decidable halting problem and universality. Lecture Notes in Computer Science **911**, in *Proceedings of LATIN'95* (1995) 386–397
5. Margenstern, M.: Frontier between decidability and undecidability: a survey. TCS **231-2** (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, Technical Report in *Publications du G.I.F.M.*, Institut Universitaire de Technologie de Metz, ISBN 2-9511539-5-3, 1999
9. Margenstern, M., Rogozhin, Yu.: About Time-Varying Distributed H systems. In Proceedings of DNA Based Computers Conference, 6, Leiden, The Netherlands (2000) 55–64
10. Margenstern, M., Rogozhin, Yu.: About Time-Varying Distributed H systems. DNA Computing, 6th International Workshop on DNA-Based Computers, DNA 2000, Leiden, The Netherlands, June 13-17, 2000, LNCS **2054** (2001) 53–62
11. Margenstern, M., Rogozhin,Yu.: Time-Varying Distributed H-systems of Degree 2 Generate All Recursively Enumerable Languages. Where Mathematics, Computer Science, Linguistics and Biology Meet. Kluwer Academic Publishers (2001) 399–407
12. Margenstern, M., Rogozhin, Yu.: A universal time-varying distribyted H system of degree 1. Accepted for DNA7 conference.
13. Cocke, J., Minsky, M.: Universality of tag systems with P=2. J. Assoc. Comput. Mach., **11** (1964) 15–20
14. Minsky, M.L.: Computations: Finite and Infinite Machines. Prentice Hall, Englewood Cliffts, NJ, 1967
15. Priese, L.: Towards a precise characterization of the complexity of universal and nonuniversal Turing machines. SIAM Journal of Computation, **8**, 4 (1979) 508–523
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., Singapore (1998) 545–556
17. Păun, A.: On Time-Varying H Systems. *Bulletin of EATCS*, N°67, (February 1999) 157–164
18. 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
19. Păun, G.: DNA Computing Based on Splicing: Universality Results. TCS **231-2** (2000) 275–296
20. Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing: New Computing Paradigms. Springer, 1998.
21. Rogozhin, Yu.: Seven universal Turing machines. Mathematical Studies, Kishinev, Academy of Sciences, **69** (1982) 76–90 (in Russian)

22. Rogozhin, Yu.: Small universal Turing machines. TCS **168-2** (1996) 215–240
23. Rogozhin, Yu.: A Universal Turing Machine with 22 States and 2 symbols. Romanian Journal of Information Science and Technology, **1**, 3 (1998) 259–265
24. Verlan, S.: On Extended Time-Varying Distributed H Systems. In Proceedings of DNA Based Computers Conference, 6, Leiden, The Netherlands (2000) 281