A universal time-varying distributed H system of degree 1

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 systems (TVDH systems shortly) of degree n are 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 2. Now we prove that there is a universal TVDH system of degree 1. It is a surprising result because we did not thought that these systems are so powerful.
Recently both authors proved that TVDH systems of degree 1 can generate any recursively enumerable languages. We present here the short description of the main idea of the proof that result.

1 Introduction

Time-varying distributed H systems are a well known theoretical model of biocomputing based on splicing operations [17–19]. We refer the reader to [6–11, 15, 23] 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.

As a biochemical motivation, that model starts from the assumption that the splicing rules are based on enzymes whose work essentially depends on the environment conditions. Hence, at any moment, only a subset of the set of all available rules is active. If the environment changes periodically, then the active enzymes also change periodically.

Consider a model M of discrete computations. It consists in a family of systems $\mathcal{S}_M$ that follow a certain paradigm: the one that is defined by the notion of computation that is associated to M. Remember that a system U of M is called *universal* when it is able to *simulate* any element of $\mathcal{S}_M$. Usually, the proof of the simulation consists in the following: starting from an encoding $<G>$ of a formal description of a member G of $\mathcal{S}_M$ and of an encoding $<\Delta>$ of some data Δ for G, to each step of computation s of G on Δ, there is a finite sequence $t_1, \ldots, t_k$ of computations of U on $(<G>,<\Delta>)$, such that the result of the application of $t_1, \ldots, t_k$ is an encoding of the result of the result of the application of G on Δ at step s.

It is known, under the name of *Church's thesis*, that any model of this kind can be simulated by a Turing machine. Indeed, this can be formally proved for each model. As a result, in order to prove that a system U in M is universal, it is enough to prove that it is able to simulate any Turing machine or any system of computation that is known to be able to simulate any Turing machine.

In particular, one way to prove the universality property consists in proving that the system is able to generate any recursively enumerable language. Classical theorems of computer science prove that this is equivalent to the property of being able to simulate any Turing machine.

In [18] it is proved that 7 different *components* are enough in order to generate any recursively enumerable language.

In [16], the number of components was reduced down to 4. 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, which is 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 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 prove that TVDH systems of degree one can carry out universal computations, i.e. that there is a TVDH system of degree one that is capable of simulating the computation of any Turing machine (for instance any universal Turing machine from [20–22]).

The question of whether or not a single component is enough for generating any recursively enumerable language is solved now, and we present here a short description of the main idea of the proof that TVDH systems of degree 1 can generate any recursively enumerable languages.

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: $\begin{array}{c|c} u_1 & u_2 \\ \hline u_1' & u_2' \end{array}$.

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:

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

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:

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

The language generated by 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:

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

3 Main Result

Theorem 1. *There is a universal TVDH system of degree 1.*

In our proof, we use tag-systems which were introduced by E.Post. Cocke and Minsky proved that tag-systems are able to define universal computations [12, 13]. Tag-systems were used in the sixties for devising the smallest universal Turing machines, see a survey [14] and, more recently [5], leading to the smallest one known up to now constructed by the second author, see [20–22]. Tag-systems were also used for constructing small non-erasing Turing machines, see [4] and [5].

Let us now define tag-systems. For a given positive integer m and a given alphabet $\Omega = \{a_1, \ldots, a_n, a_{n+1}\}$, an m-tag-system on Ω transforms the word w on Ω as follows: we delete the first m letters of w and we append to the right of the result a word that depends on the first letter of w. This process is iterated until m letters cannot be deleted or the first letter is a_{n+1}, and then stops. Formally, we have the following definitions.

A *tag-system* is a three-tuple $\mathcal{T} = (m, \Omega, P)$, where m is a positive integer, $\Omega = \{a_1, \ldots, a_n,\ a_{n+1}\}$ is a finite alphabet, and P maps $\{a_1, \ldots, a_n\}$ into the set Ω^* of finite words on the alphabet Ω and a_{n+1} to $STOP$.

A tag-system $\mathcal{T} = (m, \Omega, P)$ is called m-tag-system when it is needed to stress on number m. Words $P_i = P(a_i) \in \Omega^*$ are called the *productions* of the tag-system $\mathcal{T}$. The letter a_{n+1} is the *halting symbol.* Productions are often displayed as follows:

$$\begin{aligned} a_i &\to P_i, \quad i = 1, \ldots, n, \\ a_{n+1} &\to STOP. \end{aligned}$$

A *computation* of the tag-system $\mathcal{T} = (m, \Omega, P)$ on word $w \in \Omega^*$ is a sequence $w = v_0, v_1, \ldots,$ of words on Ω such that, for all nonnegative integer k, *a current word* v_k is transformed into v_{k+1} by deleting the first m letters of v_k and appending the word P_i to the result if the first letter of v_k is a_i. The computation stops in k steps if the length of v_k is less than m or the first letter of v_k is a_{n+1}. In that latter case, the result is v_k.

Cocke-Minsky theorem more precisely states that there are universal 2-tag-systems ([13]), and therefore, we will deal only with 2-tag-systems. Due to the proof of Coke-Minsky theorem, we can restrict our attention to tag-systems which also have the following properties:

1. The computation of a tag-system stops only on a word beginning with the halting symbol a_{n+1}.

2. The productions $P_i,\ i = 1, \ldots, n$, are not empty.

3. The current word $v_k,\ k \geq 0$ contains at least 4 letters (it is always possible to require that for tag-systems that simulate two-register machines, see the details of Minsky's proof in his book [13]).

Henceforth, the tag-systems will be 2-tag-systems satisfying the above conditions.

Lemma 1. *For every tag-system $\mathcal{T}$ there is a time-varying distributed H system $\mathcal{T}'$ of degree 1 which simulates $\mathcal{T}$.*

From the lemma, the proof of our theorem is straightforward.

Proof. Consider tag-system $\mathcal{T}$: $\begin{array}{c} a_i \to P_i \quad (i = 1, \ldots, n), \\ a_{n+1} \to STOP. \end{array}$

Let be $\Omega = \{a_j\},\ j = 1, \ldots, n+1$. Also denote the symbol a_{n+1} by h.

We construct the time-varying distributed H system $\mathcal{T}'$ as follows:

The terminal alphabet of $\mathcal{T}'$ is $T = \Omega$.

Let us introduce variables for $a_1, \ldots, a_{n+1}$, that we shall also denote by $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$.

The alphabet of $\mathcal{T}'$ is $V = \Omega \cup \{X, X', Y, Z, L, R, \xrightarrow{j}, \leftarrow\}$, $j = 1, \ldots, n+1$.

We denote $\xrightarrow{n+1}$ also by $\xrightarrow{h}$.

Axioms are given by $A = \{X'\mathbf{b} \xrightarrow{i} Z,\ L\mathbf{a} \xrightarrow{j} \mathbf{c}R,\ Z \leftarrow \mathbf{a}P_iY,\ L\mathbf{a} \leftarrow \mathbf{b}R,\ X\mathbf{a}Z,\ h \xrightarrow{h} Z,\ Z\mathbf{a}, XwY\}$, $\forall \mathbf{a}, \mathbf{b}, \mathbf{c}$ and $i = 1, \ldots, n$, $j = 1, \ldots, n+1$.

Block "START":

$$1.1: \frac{Xa_i\mathbf{ab} \mid \mathbf{c}}{X'\mathbf{b} \xrightarrow{i} \mid Z}, \quad \forall \mathbf{a}, \mathbf{b}, \mathbf{c},\ \forall i = 1, \ldots, n.$$

Block "RIGHT":

$$2.1: \frac{\mathbf{b} \mid \xrightarrow{j} \mathbf{ac}}{L \mid \mathbf{a} \xrightarrow{j} \mathbf{c}R}, \quad \forall \mathbf{a}, \mathbf{b}, \mathbf{c},\ \forall j = 1, \ldots, n+1;$$

$$2.2: \frac{\mathbf{b} \mid \xrightarrow{i} \mathbf{a}Y}{Z \mid \leftarrow \mathbf{a}P_iY}, \quad \forall \mathbf{a}, \mathbf{b},\ \forall i = 1, \ldots, n;$$

$$2.3: \frac{\mathbf{ba} \xrightarrow{j} \mathbf{c} \mid R}{L \xrightarrow{j} \mathbf{ac} \mid \mathbf{d}}, \quad \forall \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d},\ \forall j = 1, \ldots, n+1;$$

$$2.4: \frac{\mathbf{ba} \xrightarrow{j} \mathbf{c} \mid R}{L \xrightarrow{j} \mathbf{ac} \mid Y}, \quad \forall \mathbf{a}, \mathbf{b}, \mathbf{c},\ \forall j = 1, \ldots, n+1.$$

Block "LEFT":

$3.1: \dfrac{\mathbf{ab} \leftarrow | \mathbf{c}}{L\mathbf{a} \leftarrow \mathbf{b} | R}, \quad \forall \mathbf{a}, \mathbf{b}, \mathbf{c};$

$3.2: \dfrac{X'\mathbf{a} \leftarrow | \mathbf{b}}{X\mathbf{a} | Z}, \quad \forall \mathbf{a}, \mathbf{b};$

$3.3: \dfrac{\mathbf{da} | \mathbf{b} \leftarrow R}{L\mathbf{a} | \leftarrow \mathbf{bc}}, \quad \forall \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d};$

$3.4: \dfrac{X'\mathbf{a} | \mathbf{b} \leftarrow R}{L\mathbf{a} | \leftarrow \mathbf{bc}}, \quad \forall \mathbf{a}, \mathbf{b}, \mathbf{c}.$

Block "STOP":

$4.1: \dfrac{Xh | \mathbf{a}}{h \xrightarrow{h} | Z}, \quad \forall \mathbf{a};$

$4.2: \dfrac{\mathbf{b} | \xrightarrow{h} \mathbf{a}Y}{Z | \mathbf{a}}, \quad \forall \mathbf{a}, \mathbf{b}.$

Note. For each axiom $\alpha \in A$ there are rules $\dfrac{\alpha | \varepsilon}{\alpha | \varepsilon}$.

So, all the axioms $\alpha \in A$ are available at each step of operation under the TVDH system T'.

4 Details

The system $\mathcal{T}'$ works as follows.

Let the tag-system $\mathcal{T}$ start with a word $w = v_0 \in \Omega^*$. Then the system $\mathcal{T}'$ starts with a word XwY. That word, XwY, is added to the axioms of the system $\mathcal{T}'$ without adding a new rule $\dfrac{XwY | \varepsilon}{XwY | \varepsilon}$, as mentioned in Note above. So, the axiom XwY is available only at the first step of operation under the TVDH system T'.

At any time a current word $v_k \in \Omega^*,\ k \geq 0$ in the tag-system $\mathcal{T}$ is presented as a word $Xv_kY,\ X, Y \notin \Omega$ in the TVDH system $\mathcal{T}'$.

We shall see that if the tag-system $\mathcal{T}$ stops on a word $hv' = v_k \in \Omega^*$, then the system $\mathcal{T}'$ produces the same word hv' and does not produce any other word over Ω^*.

Recall that $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ are the variables for $a_1, \ldots, a_{n+1}$.

Let be $Xv_kY = Xa_i\mathbf{abc}w'Y$, $i \in \{1, 2 \ldots, n\}$.

At first the TVDH system $\mathcal{T}'$ erases the first two letters of the current word $a_i\mathbf{abc}w'$ ($Xa_i\mathbf{abc}w'Y$). Simultaneously the tag-system $\mathcal{T}'$ sends a signal $\xrightarrow{i}$ to the right end of the current word, attached to the first erased letter a_i (Block START, rule 1.1):

$(Xa_i\mathbf{ab}|\mathbf{c}w'Y,\ X'\mathbf{b} \xrightarrow{i} |Z) \vdash_{1.1} (Xa_i\mathbf{ab}Z \uparrow,\ X'\mathbf{b} \xrightarrow{i} \mathbf{c}w'Y)$, where $w' \in \Omega^*$ and $X'\mathbf{b} \xrightarrow{i} Z$ is an axiom.

So, the word $X'\mathbf{b} \xrightarrow{i} \mathbf{c}w'Y$ passes to the next step of computation and the word $Xa_i\mathbf{ab}Z$ is ruled out, because it cannot enter any splicing rule.

That signal (symbol $\xrightarrow{i}$) crosses over the word (Block RIGHT, rules 2.1, 2.3 and 2.4) until Y is met:

$(X'w'\mathbf{b}| \xrightarrow{i} \mathbf{ac}w''Y,\ L|\mathbf{a} \xrightarrow{i} \mathbf{c}R) \vdash_{2.1} (X'w'\mathbf{ba} \xrightarrow{i} \mathbf{c}R,\ L \xrightarrow{i} \mathbf{ac}w''Y)$,

$(X'w'\mathbf{ba} \xrightarrow{i} \mathbf{c}|R,\ L \xrightarrow{i} \mathbf{ac}|\mathbf{d}w'''Y) \vdash_{2.3} (X'w'\mathbf{ba} \xrightarrow{i} \mathbf{cd}w'''Y,\ L \xrightarrow{i} \mathbf{ac}R\uparrow)$,

where $w', w'', w''' \in \Omega^*$, $w'' = \mathbf{d}w'''$ and $L\mathbf{a} \xrightarrow{i} \mathbf{c}R$ is an axiom.

If $w'' = \varepsilon$ then:

$(X'w'\mathbf{ba} \xrightarrow{i} \mathbf{c}|R,\ L \xrightarrow{i} \mathbf{ac}|Y) \vdash_{2.4} (X'w'\mathbf{ba} \xrightarrow{i} \mathbf{c}Y,\ L \xrightarrow{i} \mathbf{ac}R\uparrow)$.

So, the word $X'w'\mathbf{ba} \xrightarrow{i} \mathbf{cd}w'''Y$ (or $X'w'\mathbf{ba} \xrightarrow{i} \mathbf{c}Y$) passes to the next step of computation and the word $L \xrightarrow{i} \mathbf{ac}R$ is ruled out, because it cannot enter any splicing rule.

The moment when Y is met indicates that the production P_i must be added to the right side of the current word and at that very moment, a resuming signal (a symbol $\leftarrow$) is sent backward to the beginning of the word in order to start again the process of performing one step of tag-system (Block RIGHT, rule 2.2):

$(X'w'\mathbf{b}| \xrightarrow{i} \mathbf{a}Y,\ Z| \leftarrow \mathbf{a}P_iY) \vdash_{2.2} (X'w'\mathbf{b} \leftarrow \mathbf{a}P_iY,\ Z \xrightarrow{i} \mathbf{a}Y\uparrow)$,

where $\uparrow$ indicates that computations starting from that molecule give rise to molecules that eventually are ruled out from the computation without producing a word in the terminal alphabet.

So, the word $X'w'\mathbf{b} \leftarrow \mathbf{a}P_iY$ passes to the next step of computation and the word $Z \xrightarrow{i} \mathbf{a}Y$ is ruled out, because it cannot enter any splicing rule.

After that signal $\leftarrow$ crosses over the word to left (Block LEFT, rules 3.1, 3.3 and 3.4) until X' is met: $(X'w'\mathbf{ab} \leftarrow |\mathbf{c}w''Y,\ L\mathbf{a} \leftarrow \mathbf{b}|R) \vdash_{3.1} (X'w'\mathbf{ab} \leftarrow R,\ L\mathbf{a} \leftarrow \mathbf{bc}w''Y)$,

$(X'w'''\mathbf{da}|\mathbf{b} \leftarrow R,\ L\mathbf{a}| \leftarrow \mathbf{bc}w''Y) \vdash_{3.3} (X'w'''\mathbf{da} \leftarrow \mathbf{bc}w''Y,\ L\mathbf{ab} \leftarrow R\uparrow)$,

where $w', w'', w''' \in \Omega^*$, $w' = w'''\mathbf{d}$ and $L\mathbf{a} \leftarrow \mathbf{b}R$ is an axiom.

If $w' = \varepsilon$ then:

$(X'\mathbf{a}|\mathbf{b} \leftarrow R,\ L\mathbf{a}| \leftarrow \mathbf{bc}w''Y) \vdash_{3.4} (X'\mathbf{a} \leftarrow \mathbf{bc}w''Y,\ L\mathbf{ab} \leftarrow R\uparrow)$.

So, the word $X'w'''\mathbf{da} \leftarrow \mathbf{bc}w''Y$ (or $X'\mathbf{a} \leftarrow \mathbf{bc}w''Y$) passes to the next step of computation and the word $L\mathbf{ab} \leftarrow R$ is ruled out, because it cannot enter any splicing rule.

The moment when X' is met indicates that the circle of modelling is over: the signal $\leftarrow$ vanishes and a new current word v_{j+1} ($Xv_{j+1}Y$) is obtained (Block LEFT, rule 3.2) for a new step of the tag-system computation:

$(X'\mathbf{a} \leftarrow |\mathbf{b}w''Y,\ X\mathbf{a}|Z) \vdash_{3.2} (X'\mathbf{a} \leftarrow Z\uparrow,\ X\mathbf{ab}w''Y)$.

If in the process of computation the first letter of the current word v_k is the halting symbol $h = a_{n+1}$ then the system $\mathcal{T}'$ deletes the symbol X from the beginning of the word Xv_kY and sends a special signal, for instance letter $\xrightarrow{h}$, to the right end of the word (Block STOP, rule 4.1).

$$(Xh|\mathbf{a}w''Y,\ h \xrightarrow{h} |Z) \vdash_{4.1} (XhZ\uparrow,\ h \xrightarrow{h} \mathbf{a}w''Y).$$

After that the signal $\xrightarrow{h}$ crosses over the word to the right (Block RIGHT, rules 3.1, 3.3 and 3.4) until Y is met. That moment indicates that the whole circle of modelling is over: system $\mathcal{T}'$ erases the signal $\xrightarrow{h}$ and the symbol Y, and so a result $v_{k+1} \in \Omega^*$ is obtained (Block STOP, rule 4.2).

$$(w'\mathbf{b}| \xrightarrow{h} \mathbf{a}Y,\ Z|\mathbf{a}) \vdash_{4.2} (w'\mathbf{ba},\ Z \xrightarrow{h} \mathbf{a}Y\uparrow).$$

It is easy to see that no other words than the result v_{k+1} (over the terminal alphabet Ω) can be obtained during computations of system $\mathcal{T}'$.

5 The last new result

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.

Theorem 2. $VDH_1 = VDH_* = RE.$

A short description of the main idea of 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 that 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.* $\mathcal{L} = \{f_{\mathcal{L}}(x)\ ;\ x \in \mathbb{N}\}$. 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 $q_1 01^{x+1}$ into the final configuration $q_0 01^{f_{\mathcal{L}}(x)+1}0\ldots 0$. 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 semi-infinite to the 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 $q_1 01^{x+1}$ for some x, we would obtain an encoding of $f_{\mathcal{L}}(x)$ as $q_0 01^{f_{\mathcal{L}}(x)+1}$. 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 $01^{x+2}q_0 01^{f_{\mathcal{L}}(x)+1}0\ldots 0$, starting from $q_1 01^{x+1}$. It will be possible for us to device a time-varying distributed H system of degree 1

Γ which, arriving to that stage of the computation, will split the obtained word into two parts: $01^{f_{\mathcal{L}}(x)+1}$ as a *generated* word and $q_1 01^{x+2}$ as a new starting configuration. This will allow us to obtain exactly $\mathcal{L}$ as $\mathcal{L}(\Gamma)$.

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. Cocke, J., Minsky, M.: Universality of tag systems with P=2. J. Assoc. Comput. Mach., **11** (1964) 15–20
13. Minsky, M.L.: Computations: Finite and Infinite Machines. Prentice Hall, Englewood Cliffts, NJ, 1967

14. Priese, L.: Towards a precise characterization of the complexity of universal and nonuniversal Turing machines. SIAM Journal of Computation, **8**, 4 (1979) 508–523
15. 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
16. Păun, A.: On Time-Varying H Systems. *Bulletin of EATCS*, N°67, (February 1999) 157–164
17. 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
18. Păun, G.: DNA Computing Based on Splicing: Universality Results. TCS **231-2** (2000) 275–296
19. Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing: New Computing Paradigms. Springer, 1998.
20. Rogozhin, Yu.: Seven universal Turing machines. Mathematical Studies, Kishinev, Academy of Sciences, **69** (1982) 76–90 (in Russian)
21. Rogozhin, Yu.: Small universal Turing machines. TCS **168-2** (1996) 215–240
22. Rogozhin, Yu.: A Universal Turing Machine with 22 States and 2 symbols. Romanian Journal of Information Science and Technology, **1**, 3 (1998) 259–265
23. Verlan, S.: On Extended Time-Varying Distributed H Systems. In Proceedings of DNA Based Computers Conference, 6, Leiden, The Netherlands (2000) 281