Time-varying distributed H systems with parallel computations: the problem is solved*

Maurice Margenstern[1], Yurii Rogozhin[2], and Sergey Verlan[1]

[1] LITA, Université de Metz, France
{margens,verlan}@sciences.univ-metz.fr
[2] IMI, Academy of Sciences of Moldova
rogozhin@math.md

Abstract. In this article we show that time-varying distributed H systems (TVDH systems) with one component are able to model any type-0 grammar. Thus we completely answered to the question of constructing TVDH systems of smallest degree which generate any RE language using the parallel nature of molecular computations based on splicing operations. Another interesting point is that the proof is based on a simulation of a TVDH system of degree two and not of type-0 grammars as it is usually done in similar proofs.

1 Introduction

Starting from [6], a grounding paper on splicing computations, a lot of studies were devoted to various extensions of H systems originating from [1], in particular, pointing at their possible universality power. One of these models are time-varying distributed H systems which were recently introduced in [7, 8] by G. Păun. This model introduces *components*, later see the formal definition, which cannot all be used at the same time but one after another, periodically.

In [7], it is proved that 7 different *components* are enough in order to generate any recursively enumerable language. In [3], the first two present authors proved that one component is enough in order to generate any recursively enumerable language. The proof was made by a sequential modelling of Turing machines.

In the meanwhile, A. Păun published a paper [5] where he proved that time-varying distributed H systems with 4 components generate all recursively enumerable languages. But his solution is a parallel one: its proof consists in modelling any type-0 grammar. A. Păun's result was improved in 2000 by the first two authors of the present paper by reducing the number of components of TVDH systems which model type-0 formal grammars down to 3 [2]. In 2002 the authors proved that it is possible to model type-0 grammars only with two components [4].

Now we improve the last result by reducing the number of components of TVDH systems which model type-0 formal grammars down to one. Thus we completely solve the problem of constructing TVDH systems of smallest degree

* Work supported by French Ministry of Education, *NATO* project PST.CLG.976912 and project IST-2001-32008 **MolCoNet**

which generate any RE language using the parallel nature of molecular computations based on splicing operations.

At last, but not the least, we notice that this proof is very different from previous ones of the first two authors. Indeed, in the mentioned previous proofs, the authors simulated a type-0 grammar. In this proof, we simulate a TVDH system of degree 2, D_G^2, from [4]. Moreover, the simulation is very close to the execution of D_G^2: most applications of a rule of D_G^2 is simulated in one step of the TVDH system of degree 1 which is constructed in the paper. In this regard, this proof is more direct than the previous ones.

Within the limits of this paper it is not possible to give a thorough checking of the computations. We only outline the main arguments of the proof. In [9] the reader can find a similar method in action which is thoroughly explained.

2 Basic definitions

We recall some notions. 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: $\dfrac{u_1 | u_2}{u_1' | u_2'}$.

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 an *splicing scheme* or an H *scheme*.

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

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

A time-varying distributed H *system* [7] (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*, $T \subseteq V$, the *terminal alphabet*, $A \subseteq V^*$, 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

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

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 which cannot enter a splicing are removed.

The language generated by D is:

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

3 TVDH systems of degree 1

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

The theorem follows immediately from Lemmas 1 and 2.

Lemma 1. *For any type-0 formal grammar* $G = (N, T, P, S)$ *there is a TVDH system* $D_G^2 = (V, T, A, R_1, R_2)$ *of degree 2 which simulates* G *and* $L(D_G) = L(G)$.

The proof of this lemma is given in [4].

Lemma 2. *For any TVDH system* D_G^2 *constructed in Lemma 1 there is a TVDH system* $D_G = (V, T, A, R_1)$ *of degree 1 which simulates* D_G^2 *and* $L(D_G) = L(D_G^2)$.

In order to prove the lemma we slightly transform, from technical considerations, the TVDH2 system D_G^2 from [4] into a form which is more suitable for a simulation. Some letters are numbered and some rules have a bigger context, but these transformations do not alter neither the functioning of the system nor the generated language. We will not distinguish between this transformed system and the system from [4] and we will refer further to it as D_G^2. In fact, this system contains the rules 1–18 (see below) distributed over two components as well as some additional rules. The system has the following properties:

- It has two components
- The system has two independent sets of axioms and they persist during the computations. One of these sets is used for splicing in the first component only, the other one is used for splicing in the second component only
- The rules are made in a such way that molecules that code different stages of the simulation of the grammar can be spliced only with axioms from the two above sets.

The method we use to perform the simulation of the system D_G^2 is the following. In our system D_G we have two versions of each axiom for each set: a normal (active) and a prime (passive) version. On an odd step of computation we have active versions of molecules from the first set and passive versions of molecules from the second set. On an even step we have passive versions of the molecules from the first set and active version of the molecules from the second one. The prime versions of molecules cannot enter any rule except the unpriming rule. The normal versions of molecules may enter either a splicing rule which corresponds to a rule of D_G^2 or they may enter a priming rule which tags these molecules with a prime. Thus we create the active molecules only at the moment we really need them. In that way we simulate the behaviour of the first component during an odd step and we simulate the behaviour of the second component during an even step. And so, we perform a correct simulation. The rules from both components of D_G^2 are simply merged into one component. We start with a normal version of the first set and with a prime version of the second set. The transformation of TVDH2 system into TVDH1 system is shown in Fig. 1.

For example: we have molecule Z_2Y_i on a odd step. It may be used for splicing using rule 2 thus we performed the simulation of rule 1.2 of the TVDH2 system or it may be tagged with a prime using rule $z2$: $(Z_2Y_i, Z_2'R_1) \vdash_{z2} (Z_2'Y_i, Z_2R_1)$. On the next step (even step) this molecule (Z_2Y_i) is not accessible so rule 2 cannot be applied. On the other hand we can unprime molecule $Z_2'Y_i$ using rule $z2'$ (molecule Z_2R_1 was obtained on the previous step) and thus we obtain again molecule Z_2Y_i and we are on a odd step.

A similar lot happens to all molecules having Z (with indices).

Rules 1.8, 1.10, 2.7 and 2.9 are used in D_G^2 to obtain the resulting terminal string. In D_G, due to technical details, they are simulated in a special way by rules 7, 16, x.1, x.2, and r.1–r.16.

TVDH2 system D_G^2

Component I	Component II
Rule 1.2: $\begin{array}{c\|c} \mathbf{a} & a_iY \\ \hline Z_2 & Y_i \end{array}$	Rule 2.2: $\begin{array}{c\|c} X_l & \mathbf{ab} \\ \hline X_{l-1}' & Z_{13} \end{array}$
Axiom: Z_2Y_i	Axiom: $X_j'Z_{13}$
Rule for axiom: (present in both components) $\begin{array}{c\|c} Z_2Y_i & \varepsilon \\ \hline Z_2Y_i & \varepsilon \end{array}$	Rule for axiom: (present in both components) $\begin{array}{c\|c} X_j'Z_{13} & \varepsilon \\ \hline X_j'Z_{13} & \varepsilon \end{array}$

TVDH1 system D_G (1 component)

Rule 2: $\begin{array}{c\|c} \mathbf{a} & a_iY \\ \hline Z_2 & Y_i \end{array}$	Rule 13: $\begin{array}{c\|c} X_l & \mathbf{ab} \\ \hline X_{l-1}' & Z_{13} \end{array}$
Axioms: $Z_2Y_i,\ Z_2'R_1$	Axioms: $X_j'Z_{13},\ R_2Z_{13}$
Rules for axioms: $z2$: $\begin{array}{c\|c} Z_2 & Y_i \\ \hline Z_2' & R_1 \end{array}$ $z2'$: $\begin{array}{c\|c} Z_2' & Y_i \\ \hline Z_2 & R_1 \end{array}$	Rules for axioms: $z13$: $\begin{array}{c\|c} X_j' & Z_{13} \\ \hline R_2 & Z_{13}' \end{array}$ $z13'$: $\begin{array}{c\|c} X_j' & Z_{13}' \\ \hline R_2 & Z_{13} \end{array}$

Fig. 1. Transformation of TVDH2 system D_G^2 into a TVDH1 system.

Now we will give a more formal definition of the system.

We define $D_G = (V, T, A, R_1)$ 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:

$1 \leq i \leq n,\ 1 \leq j \leq n-1,\ 2 \leq k \leq n,\ s \in \{0, 2\},\ \mathbf{a}, \mathbf{b} \in N \cup T \cup \{B\}$.

Alphabet: $V = \{V^2 \setminus \{C_1, C_2, D_1, D_2\}\} \cup \{R_1, R_2, Z_m'\ {\scriptstyle (1 \leq m \leq 18)}\} \cup \{C_1^s, C_2^s, R, Z_7^p, Z_{16}^p, Z_D^p, Z_E^p\ {\scriptstyle (1 \leq p \leq 3)}, D_1^s, E_2^s, Z_D, Z_E\}$.

The terminal alphabet is $T = T^2$ (the same as for G).

Axioms: $A = \{XSBY\} \cup A_1 \cup A_2 \cup A_R$, where

$A_1 = \{Z_1vY_i : \exists u \to va_i \in P\} \cup \{Z_1Y_i : \exists u \to \epsilon \in P\} \cup \{Z_2Y_i, Z_3Y_j, Z_4Y_j', Z_5Y_j'', Z_6Y, X'Z_8, X''Z_9\} \cup \{Z_m'R_1 (1 \leq m \leq 7), R_1Z_8', R_1Z_9'\}$.

$A_2 = \{X_ia_iZ_{11}', X_jZ_{12}', X_j'Z_{13}', X_j''Z_{14}', XZ_{15}', Z_{17}'Y', Z_{18}'Y''\} \cup$

$\{Z_{17}R_2, Z_{18}R_2, R_2Z_{m(11\leq m\leq 16)}\}$.
$A_R = \{C_1^0Z_{16}, C_1^2Z_{16}^2, RZ_{16}^1, RZ_{16}^3, Z_7^1C_2^0\} \cup$
$\{Z_7^3C_2^2, Z_7^2R, Z_7R, D_1^0Z_D^1, D_1^2Z_D^3, RZ_D^2, RZ_D, Z_EE_2^0, Z_E^2E_2^2, Z_E^1R, Z_E^3R\}$.

The axioms from A_1, A_2 are used to simulate the behaviour of the first and respectively second component. The axioms from A_R are used in order to obtain the result.

Simulation of rules of the first component of the original system (the original number is given in parenthesis):

$$1(1.1) : \frac{\mathbf{a} \mid uY}{Z_1 \mid vY_i}\ {\scriptstyle (\exists u\to va_i\in P)};\quad 1'(1.1') : \frac{\mathbf{a} \mid a_iuY}{Z_1 \mid Y_i}\ {\scriptstyle (\exists u\to \epsilon\in P)};\quad 2(1.2) : \frac{\mathbf{a} \mid a_iY}{Z_2 \mid Y_i};$$

$$3(1.9) : \frac{\mathbf{a} \mid Y_j''}{Z_3 \mid Y_j};\quad 4(1.3) : \frac{\mathbf{ab} \mid Y_k}{Z_4 \mid Y_{k-1}'};\quad 5(1.4) : \frac{\mathbf{a} \mid Y_j'}{Z_5 \mid Y_j''};$$

$$6(1.7) : \frac{\mathbf{a} \mid Y''}{Z_6 \mid Y};\quad 7(1.8) : \frac{\mathbf{a} \mid BY''}{Z_7 \mid C_2^s};\quad 8(1.5) : \frac{X_1 \mid \mathbf{ab}}{X' \mid Z_8};\quad 9(1.6) : \frac{X' \mid \mathbf{a}}{X'' \mid Z_9};$$

Simulation of rules of the second component of the original system (the original number is given in parenthesis):

$$11(2.1) : \frac{X \mid \mathbf{a}}{X_ia_i \mid Z_{11}};\quad 12(2.8) : \frac{X_j'' \mid \mathbf{a}}{X_j \mid Z_{12}};\quad 13(2.2) : \frac{X_k \mid \mathbf{ab}}{X_{k-1}' \mid Z_{13}};$$

$$14(2.3) : \frac{X_j' \mid \mathbf{a}}{X_j'' \mid Z_{14}};\quad 15(2.6) : \frac{X'' \mid \mathbf{a}}{X \mid Z_{15}};\quad 16(2.7) : \frac{X'' \mid \mathbf{a}}{C_1^s \mid Z_{16}};$$

$$17(2.4) : \frac{\mathbf{ab} \mid Y_1}{Z_{17} \mid Y'};\quad 18(2.5) : \frac{\mathbf{a} \mid Y'}{Z_{18} \mid Y''};$$

Creation of axioms from the first set (used in the first component):
Priming rules:

$$z1 : \frac{Z_1 \mid vY_i}{Z_1' \mid R_1};\quad z2 : \frac{Z_2 \mid Y_i}{Z_2' \mid R_1};\quad z3 : \frac{Z_3 \mid Y_j}{Z_3' \mid R_1};\quad z4 : \frac{Z_4 \mid Y_j'}{Z_4' \mid R_1};$$

$$z5 : \frac{Z_5 \mid Y_j''}{Z_5' \mid R_1};\quad z6 : \frac{Z_6 \mid Y}{Z_6' \mid R_1};\quad z8 : \frac{X' \mid Z_8}{R_1 \mid Z_8'};\quad z9 : \frac{X'' \mid Z_9}{R_1 \mid Z_9'};$$

Unpriming rules:

$$z1' : \frac{Z_1' \mid vY_i}{Z_1 \mid R_1};\quad z2' : \frac{Z_2' \mid Y_i}{Z_2 \mid R_1};\quad z3' : \frac{Z_3' \mid Y_j}{Z_3 \mid R_1};\quad z4' : \frac{Z_4' \mid Y_j'}{Z_4 \mid R_1};$$

$$z5' : \frac{Z_5' \mid Y_j''}{Z_5 \mid R_1};\quad z6' : \frac{Z_6' \mid Y}{Z_6 \mid R_1};\quad z8' : \frac{X' \mid Z_8'}{R_1 \mid Z_8};\quad z9' : \frac{X'' \mid Z_9'}{R_1 \mid Z_9};$$

Creation of axioms from the second set (used in the second component):
Unpriming rules:

$$z11' : \frac{X_ia_i \mid Z_{11}'}{R_2 \mid Z_{11}};\quad z12' : \frac{X_j \mid Z_{12}'}{R_2 \mid Z_{12}};\quad z13' : \frac{X_j' \mid Z_{13}'}{R_2 \mid Z_{13}};\quad z14' : \frac{X_j'' \mid Z_{14}'}{R_2 \mid Z_{14}};$$

$$z15' : \frac{X \mid Z_{15}'}{R_2 \mid Z_{15}};\quad z17' : \frac{Z_{17}' \mid Y'}{Z_{17} \mid R_2};\quad z18' : \frac{Z_{18}' \mid Y''}{Z_{18} \mid R_2};$$

Priming rules:

$$z11: \begin{array}{c|c} X_i a_i & Z_{11} \\ \hline R_2 & Z'_{11} \end{array}; \quad z12: \begin{array}{c|c} X_j & Z_{12} \\ \hline R_2 & Z'_{12} \end{array}; \quad z13: \begin{array}{c|c} X'_j & Z_{13} \\ \hline R_2 & Z'_{13} \end{array}; \quad z14: \begin{array}{c|c} X''_j & Z_{14} \\ \hline R_2 & Z'_{14} \end{array};$$

$$z15: \begin{array}{c|c} X & Z_{15} \\ \hline R_2 & Z'_{15} \end{array}; \quad z17: \begin{array}{c|c} Z_{17} & Y' \\ \hline Z'_{17} & R_2 \end{array}; \quad z18: \begin{array}{c|c} Z_{18} & Y'' \\ \hline Z'_{18} & R_2 \end{array};$$

Rules for result:

$$x.1: \begin{array}{c|c} \mathbf{a} & C_2^s \\ \hline Z_E E_2^s & \varepsilon \end{array}; \quad x.2: \begin{array}{c|c} C_1^s & \mathbf{a} \\ \hline \varepsilon & D_1^s Z_D \end{array};$$

$$r.1: \begin{array}{c|c} C_1^s & Z_{16} \\ \hline R & Z_{16}^1 \end{array}; \quad r.2: \begin{array}{c|c} C_1^s & Z_{16}^1 \\ \hline R & Z_{16}^2 \end{array}; \quad r.3: \begin{array}{c|c} C_1^s & Z_{16}^2 \\ \hline R & Z_{16}^3 \end{array}; \quad r.4: \begin{array}{c|c} C_1^s & Z_{16}^3 \\ \hline R & Z_{16} \end{array};$$

$$r.5: \begin{array}{c|c} D_1^s & Z_D \\ \hline R & Z_D^1 \end{array}; \quad r.6: \begin{array}{c|c} D_1^s & Z_D^1 \\ \hline R & Z_D^2 \end{array}; \quad r.7: \begin{array}{c|c} D_1^s & Z_D^2 \\ \hline R & Z_D^3 \end{array}; \quad r.8: \begin{array}{c|c} D_1^s & Z_D^3 \\ \hline R & Z_D \end{array};$$

$$r.9: \begin{array}{c|c} Z_7 & C_2^s \\ \hline Z_7^1 & R \end{array}; \quad r.10: \begin{array}{c|c} Z_7^1 & C_2^s \\ \hline Z_7^2 & R \end{array}; \quad r.11: \begin{array}{c|c} Z_7^2 & C_2^s \\ \hline Z_7^3 & R \end{array}; \quad r.12: \begin{array}{c|c} Z_7^3 & C_2^s \\ \hline Z_7 & R \end{array};$$

$$r.13: \begin{array}{c|c} Z_E & E_2^s \\ \hline Z_E^1 & R \end{array}; \quad r.14: \begin{array}{c|c} Z_E^1 & E_2^s \\ \hline Z_E^2 & R \end{array}; \quad r.15: \begin{array}{c|c} Z_E^2 & E_2^s \\ \hline Z_E^3 & R \end{array}; \quad r.16: \begin{array}{c|c} Z_E^3 & E_2^s \\ \hline Z_E & R \end{array};$$

Program check

The obtained system was checked for errors using `TVDHsim`: time varying distributed H systems simulator, a program developed by the third author at the University of Metz. It can be found at `http://lita.sciences.univ-metz.fr/~verlan/`. The same program was also widely used during the construction of the system.

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. Margenstern M., Rogozhin Yu.: About Time-Varying Distributed H Systems. *Lecture Notes in Computer Science*, Springer, vol. **2054**, 2001, p.53-62.
3. Margenstern, M., Rogozhin, Yu.: Time-varying distributed H systems of degree 1 generate all recursively enumerable languages, in Words, Semigroups and Transductions (M. Ito, Gh. Paun, S. Yu, eds), World Sci.,Singapore, 2001, p. 329-340.
4. Margenstern, M., Rogozhin, Yu., Verlan, S.: Time-Varying Distributed H Systems of Degree 2 Can Carry Out Parallel Computations. *Lecture Notes in Computer Science*, Springer, vol. **2568**, 2003, p.326-336.
5. Păun, A.: On Time-Varying H Systems. *Bulletin of EATCS*. **67** (1999) 157–164.
6. Păun, G., Rozenberg, G., Salomaa, A.: Computing by splicing. *TCS*. **168**, no.2 (1996) 321–336.
7. Păun, G.: DNA Computing Based on Splicing: Universality Results. *TCS*. **231**, no.2 (2000) 275–296.
8. Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing: New Computing Paradigms. Springer (1998).
9. Verlan, S.: A frontier result on enhanced time-varying distributed H systems with parallel computations, Preproceedings of DCFS'03, Descriptional Complexity of Formal Systems, Budapest, Hungary, July 12-14, 2003, p. 221-232.