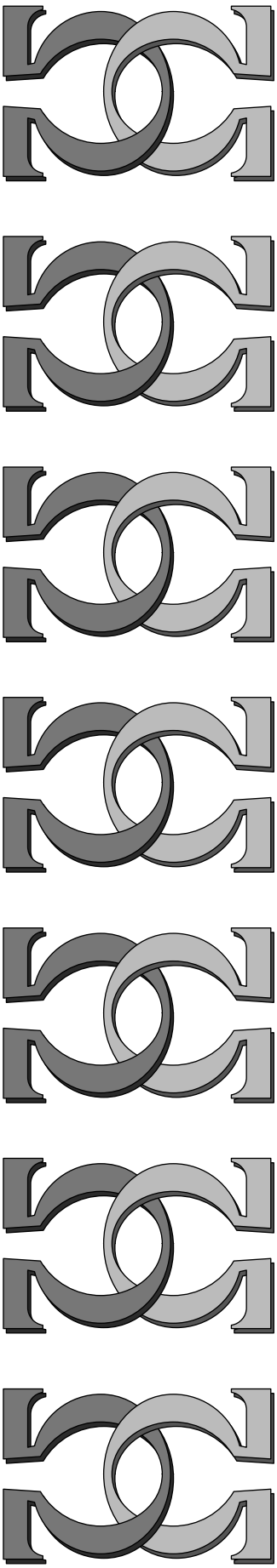

CDMTCS
Research
Report
Series

DNA Computing Based on Splicing: Universality Results

Gheorghe Păun
Institute of Mathematics of the Romanian Academy

CDMTCS-072
December 1997

Centre for Discrete Mathematics and
Theoretical Computer Science

DNA Computing Based on Splicing: Universality Results

Gheorghe PĂUN
Institute of Mathematics of the Romanian Academy
PO Box 1 – 764, 70700 Bucureşti, Romania
Email: gpaun@imar.ro

Abstract. First, we recall some characterizations of recursively enumerable languages by means of finite H systems with certain regulations on the splicing operation. Then, we consider a variant of the splicing operation where the splicing proceeds always in couples of steps: the two strings obtained after a splicing enter immediately a second splicing (the rules used in the two steps are not prescribed). Somewhat surprising if we take into account the loose control on the performed operations, extended H systems with finite sets of axioms and of splicing rules, using this double splicing operation, can again characterize the recursively enumerable languages. Finally, we consider two types of distributed H systems: communicating distributed H systems and time-varying distributed H systems. For the first type of devices, we give a new proof of the recent result of [24] that (in the extended case) such systems with three components characterize the recursively enumerable languages. In what concerns the second mentioned distributed model, we prove that time-varying H systems with seven components can characterize the recursively enumerable languages. The optimality of these two last mentioned results is open.

1. Introduction

An *extended H system* is a language generating mechanism introduced in [20], based on the splicing operation of [10]. This operation is a formal model of the recombinant behavior of DNA molecules under the influence of restriction enzymes and ligases. Informally speaking, two DNA sequences are cut by two restriction enzymes and the fragments are recombined (by ligation, provided that the ends produced by the enzymes match) such that possibly new sequences are produced. The sites where the enzymes can cut are encoded as pairs $(u_1, u_2), (u_3, u_4)$, and the fact that they produce matching ends is represented by the quadruple $((u_1, u_2), (u_3, u_4))$. We say that this is a *splicing rule*. In an *H system*, a set of axioms (initial strings) and a set of splicing rules are given. By an iterated application of these rules, starting from the axioms, we get a language. If also a terminal alphabet is provided and only strings on that alphabet are accepted, then we get the notion of an *extended H system.*

If only a finite set of rules are used, then, even starting from a regular set of axioms, we can generate only regular languages (see, [5], [23]). When using extended H systems, we obtain a characterization of regular languages, [20].

If the set of splicing rules is a regular language (each rule $((u_1, u_2), (u_3, u_4))$ is written as a string $u_1\#u_2\$u_3\#u_4$, hence the set of rules can be considered a language), then extended H systems (with finite sets of axioms) characterize the recursively enumerable languages, that is, they reach the full power of Turing machines/Chomsky type-0 grammars. This has been proved in [14].

However, working with infinite sets of rules, even regular, is not of much practical interest. Finite sets of rules give only regular languages, hence they stop at the level of finite automata/regular Chomsky grammars. There is no natural definition of universality for finite automata such that a universal finite automaton exists (hence we cannot obtain "universal DNA computers" at this level). It is therefore necessary to supplement the model with a feature able to increase its power. Many suggestions about how this can be done come both from the regulated rewriting area in formal language theory, see, e.g., [6], and from the very proof in [14]. Several types of extended H systems with finite sets of axioms and of splicing rules were considered, with the application of splicing rules controlled in specific ways. We mention the control by *permitting* contexts (a rule is applied only to strings containing certain symbols associated to the rule), *forbidding* contexts (a rule is applied only to strings not containing certain symbols associated to the rule; the permitting contexts are a model of *promoters*, the forbidding contexts correspond to *inhibitors* known in biochemistry), [3], [8]; *target* languages (we accept the splicing only when the obtained strings belong to a given regular language), *fitness* mappings, as in the genetic algorithms area, [16]; working with *multisets* (keeping track of the number of copies of each string, starting with the axioms), [3], [8]; *programmed* H systems (the splicing rule to be used at any step depends on the rule used at the previous step), or *evolving* H systems (the splicing rules themselves are modified from a step to the next one, by means of local mutations, that is insertion and deletion operations of single symbols), [21]; H systems with a *priority* relation among splicing rules (at each step one uses a rule which is maximal among the rules which can be applied to the chosen strings). In all these cases, *computational completeness* is obtained, that is characterizations of recursively enumerable languages. Moreover, *universal H systems* of the mentioned types are obtained, in the usual sense: with all components fixed and able to simulate any given H system as soon as a code of it is introduced as an additional axiom in the universal system. Full details about results of this type can be found in the forthcoming monograph [22].

The fact that finite H systems with uncontrolled splicing can generate only regular languages, but apparently weak controls directly lead to characterizations of recursively enumerable languages is worth emphasizing. Roughly spleaking, in order to equal the power of type-0 Chomsky grammars, hence to characterize the recursively enumerable languages, we need two basic ingredients: context-sensitivity and (unbounded) erasing. Moreover, context-sensitivity means not only context-dependency of the operations performed, but also the possibility "to send messages" at arbitrary distances in the processed strings. By its definition, the splicing operation has both context-sensitivity and erasing. However, we still need to improve on the context-sensitivity by means of the mentioned controls. The explanation is that by using only the splicing we cannot "send mesages" along the strings: when a string is cut in parts, we cannot enforce the meeting of the two parts in a further

splicing operation. At a close examination, exactly this is ensured by all control mechanisms mentioned above (and it will be quite visible in the new control we introduce here, the double splicing). On the other hand, the splicing is a "natural" operation, whereas the controls mentioned above are, all, inspired by formal language theory (hence unrealistic for the present day lab techniques). In order to obtain computational completeness we have to pay this price of the control. Because all proofs in this area are constructive (one starts from type-0 grammars and one produces equivalent H systems of the various types mentioned above), they directly imply the existence of universal H systems of these types. The proofs can be slightly modified in such a way to start also from Turing machines. However, this indirect way of producing universal H systems leads to rather complex outputs, by no means accessible to the available biochemical technology. It is a research topic to find a small universal H system of a given type.

We continue here this direction of investigation of imposing restrictions on the splicing operation, by considering *H systems with double splicing*: we ask that the splicing operations take place in double steps consisting of two usual splicings, that is, the two strings obtained by a splicing enter immediately a new splicing, as the two terms of it. The rules used in a step are not prescribed or linked in any prescribed way; however, the intermediate strings, those obtained after the first splicing of such a double step, are not "visible", they are immediately consumed by the second splicing.

The way of splicing in double steps can be seen as a counterpart of the matrix restriction in Chomsky grammars. However, we do not have here matrices specified in advance. All pairs of rules can constitute matrices. Note that in the case of context-free grammars such a restriction on the derivation does not increase the generative power: it simply implies that any derivation has a length which is a multiple of two. Obviously, this does not modify the power of context-free grammars (for instance, introduce new rules $S \to w$, where w is obtained either in one or in two steps in a given grammar, in order to work only with derivations of an even length). In the splicing case, the effect of this restriction is maximal: we jump from the regular languages to recursively enumerable languages (Theorem 3 below).

All these models based on controlled splicing have a common drawback (plus other specific shortcomings) when looking for implementing them: they use a large number of splicing rules, which means a large number of restriction enzymes. In general, several restriction enzymes cannot work together, because each enzyme requires specific conditions, temperature, salinity, etc. (Discusions on this topic can be found, for instance, in [11].) A possible idea to diminish this drawback is to use distributed architectures, as in grammar systems area, [2], [7]. A variant of "distributed test tube systems" was introduced in [4]. Several H systems work independently using their splicing rules and communicate by sending to each other strings; these "messages" are accepted only if they pass certain *filters* (if they are composed of symbols in given subalphabets); the language generated by a designated component of the system is the language generated by the system. Again, a characterization of recursively enumerable languages is obtained. The proof of this result from [4] does not give a bound on the number of components, but in [29] it is shown that distributed systems as in [4] with at most nine components can characterize the recursively enumerable languages. The same authors have then improved by one the result, whereas in [17] it is shown that six components suffice. Recently, it was proven that systems with only three components characterize the recursively enumerable languages, [24]. We give here a new proof of this important result, also bounding the radius of the splicing rules used (the maximal length of a

string u_i in the splicing rules $u_1\#u_2\$u_3\#u_4$). Whether or not two components suffice is still an open problem (we *conjecture* that such systems generate only context-free languages).

A related machinery are the time-varying distributed H systems, introduced in [17]. Again we have several usual H systems, but at any moment only one is enabled; the order of enabling the system components is periodic in time; the components pass from each other the result of the splicing and all terminal strings generated in this way form the language generated by the system. In [17] one characterizes the recursively enumerable languages by time-varying H systems with three splicing rules in each component, but without bounding the number of components. We prove here that seven components are enough (this time the size of components is no longer bounded). It is an open problem whether or not seven components are enough.

2. Splicing Systems

Let us consider an alphabet V and two special symbols, $\#, \$$, not in V. By V^* we denote the set of all strings over V, including the empty one, denoted by λ. By $|x|$ we denote the length of $x \in V^*$. By *FIN, REG, LIN, CF, CS, RE* we denote the families of finite, regular, linear, context-free, context-sensitive, recursively enumerable languages, respectively. For further elements of formal language theory we refer to [26].

A *splicing rule* over V is a string $u_1\#u_2\$u_3\#u_4$, where $u_1, u_2, u_3, u_4 \in V^*$. The maximum of $|u_i|, 1 \leq i \leq 4$, is called the *radius* of this splicing rule. For a splicing rule $r = u_1\#u_2\$u_3\#u_4$ and four strings $x, y, w, z \in V^*$ we write

$$\begin{aligned}(x, y) \vdash_r (w, z) \quad \text{iff} \quad & x = x_1u_1u_2x_2,\ y = y_1u_3u_4y_2,\\ & w = x_1u_1u_4y_2,\ z = y_1u_3u_2x_2,\\ & \text{for some } x_1, x_2, y_1, y_2 \in V^*.\end{aligned}$$

We say that we *splice* the strings x, y at the *sites* u_1u_2, u_3u_4, respectively.

A pair $\sigma = (V, R)$, where V is an alphabet and R is a set of splicing rules over V is called an *H scheme*. With respect to a splicing scheme $\sigma = (V, R)$ and a language $L \subseteq V^*$ we define

$$\begin{aligned}&\sigma(L) = \{w \in V^* \mid (x, y) \vdash_r (w, z) \text{ or } (x, y) \vdash_r (z, w), \text{ for some } x, y \in L, r \in R\},\\ &\sigma^0(L) = L,\\ &\sigma^{i+1}(L) = \sigma^i(L) \cup \sigma(\sigma^i(L)),\ i \geq 0,\\ &\sigma^*(L) = \bigcup_{i \geq 0} \sigma^i(L).\end{aligned}$$

An *extended H system* is a construct

$$\gamma = (V, T, A, R),$$

where V is an alphabet, $T \subseteq V, A \subseteq V^*$, and $R \subseteq V^*\#V^*\$V^*\#V^*$. ($T$ is the *terminal* alphabet, A is the set of *axioms*, and R is the set of *splicing rules*.) When $T = V$, the system is said to be *non-extended.* The pair $\sigma = (V, R)$ is the *underlying H scheme* of γ.

The language generated by γ is defined by

$$L(\gamma) = \sigma^*(A) \cap T^*.$$

(We iterate the splicing operation according to rules in R, starting from strings in A, and we keep only the strings composed of terminal symbols.)

We denote by $EH(F_1, F_2)$ the family of languages generated by extended H systems $\gamma = (V, T, A, R)$, with $A \in F_1, R \in F_2$, where F_1, F_2 are two given families of languages. (Note that R is a language, hence the definition makes sense.)

Two basic results concerning the power of extended H systems are the following ones.

Theorem 1. $EH(FIN, FIN) = EH(REG, FIN) = REG$.

Theorem 2. $EH(FIN, REG) = RE$.

The inclusion $EH(REG, FIN) \subseteq REG$ follows from the results in [5], [23], the inclusion $REG \subseteq EH(FIN, FIN)$ is proved in [20]. Theorem 2 is proved in [14].

3. The Operation of Double Splicing

Consider a splicing scheme $\sigma = (V, R)$, four strings x, y, w, z in V^*, and two rules r_1, r_2 in R. We write

$$(x, y) \vdash_{r_1, r_2} (w, z) \text{ iff } (x, y) \vdash_{r_1} (u, v) \text{ and } (u, v) \vdash_{r_2} (w, z), \text{ for some } u, v \in V^*.$$

Then, for a language $L \subseteq V^*$ we define

$$\begin{aligned}
&\sigma_d(L) = \{w \mid (x, y) \vdash_{r_1, r_2} (w, z) \text{ or } (x, y) \vdash_{r_1, r_2} (z, w), \text{ for } x, y \in L, r_1, r_2 \in R\},\\
&\sigma_d^*(L) = \bigcup_{i \geq 0} \sigma_d^i(L), \text{ where}\\
&\sigma_d^0(L) = L,\\
&\sigma_d^{i+1}(L) = \sigma_d^i(L) \cup \sigma_d(\sigma_d^i(L)), \ i \geq 0.
\end{aligned}$$

Let $\gamma = (V, T, A, R)$ be an extended H system and $\sigma = (V, R)$ its underlying splicing scheme. We associate with γ the language

$$L_d(\gamma) = \sigma_d^*(A) \cap T^*.$$

For two families of languages F_1, F_2 we denote by $EH_d(F_1, F_2)$ the family of languages $L_d(\gamma)$ generated as above by extended H systems $\gamma = (V, T, A, R)$ with $A \in F_1$ and $R \in F_2$.

Let us examine an **example**: consider the extended H system

$$\gamma = (\{a, b, c, d, e\}, \{a, b, c, d\}, \{cabd, caebd\}, R),$$

with R containing the splicing rules

$$r_1 = c\#a\$ca\#ebd, \ r_2 = ce\#bd\$b\#d.$$

Take a string of the form $ca^nb^nd, n \geq 1$; one of the axioms is of this form, with $n = 1$. The only possible splicing involving this string is

$$(c|a^nb^nd, ca|ebd) \vdash_{r_1} (cebd, ca^{n+1}b^nd).$$

In the sense of the double splicing operation, we have to continue; the only possibility is

$$(ce|bd, ca^{n+1}b^n|d) \vdash_{r_2} (ced, ca^{n+1}b^{n+1}d).$$

Consequently, we have

$$(ca^nb^nd, caebd) \vdash_{r_1,r_2} (ced, ca^{n+1}b^{n+1}d).$$

The operation can be iterated.

Another possibility is to start with two copies of the axiom $caebd$:

$$(c|aebd, ca|ebd) \vdash_{r_1} (ce|bd, caaeb|d) \vdash_{r_2} (ced, caaebbd).$$

We can continue, but the symbol e will be present in all obtained strings; these strings cannot enter splicings with strings of the form ca^nb^nd, hence they do not lead to terminal strings.

In conclusion, we obtain

$$L_d(\gamma) = \{ca^nb^nd \mid n \geq 1\},$$

which is not a regular language. Contrast this with Theorem 1: the double splicing is strictly more powerful than the simple one. This assertion will be stressed in the next section in the strongest possible way: extended H systems using the double splicing operation are equal in power to type-0 grammars.

4. A Characterization of Recursively Enumerable Languages

It is known that every recursively enumerable language can be obtained from a linear language (even generated by a grammar using only one nonterminal) by using reduction rules of the form $u \to \lambda$. Results of this type can be found in [9], [28], etc. We will use here the following variant, from [28]:

Lemma 1. *Each language $L \in RE, L \subseteq T^*$, can be generated by a grammar of the form $G = (\{S, B_1, B_2, B_3, B_4\}, T, S, P \cup \{B_1B_2 \to \lambda, B_3B_4 \to \lambda\})$, where P contains rules of the forms $S \to uSv, S \to x$, with $u, v, x \in (T \cup \{B_1, B_2, B_3, B_4\})^+$.*

Theorem 3. $RE = EH_d(FIN, FIN)$.

Proof. We prove only the inclusion $\subseteq$. The reverse inclusion can be proved by a straightforward construction of a type-0 grammar simulating an extended H system based on the double splicing operation (or we can invoke the Church-Turing thesis).

Consider a grammar $G = (\{S, B_1, B_2, B_3, B_4\}, T, S, P \cup \{B_1B_2 \to \lambda, B_3B_4 \to \lambda\})$ as in Lemma 1. We construct the extended H system $\gamma = (V, T, A, R)$ with:

$$\begin{aligned}
V &= T \cup \{S, B_1, B_2, B_3, B_4, X, Y, Z, Z'\},\\
A &= \{SxS \mid S \to x \in P, x \in (T \cup \{B_1, B_2, B_3, B_4\})^*\}\\
&\quad \cup \{SuZvS \mid S \to uSv \in P\}\\
&\quad \cup \{Z', XY\},\\
R &= \{S\#\$Su\#ZvS,\ SZ\#vS\$\#S \mid S \to uSv \in P\}\\
&\quad \cup \{S\#\$\#Z',\ SZ'\#\$\#S\}\\
&\quad \cup \{B_1\#B_2\$X\#Y,\ \#B_1Y\$XB_2\#\}\\
&\quad \cup \{B_3\#B_4\$X\#Y,\ \#B_3Y\$XB_4\#\}.
\end{aligned}$$

The idea of this construction is the following. The splicing rules of the forms $S\#\$Su\#ZvS$, $SZ\#vS\$\#S$ simulate the context-free rules $S \to uSv$ in P, while the splicing rules $B_1\#B_2\$X\#Y$, $\#B_1Y\$XB_2\#$, $B_3\#B_4\$X\#Y$, $\#B_3Y\$XB_4\#$ simulate the rules $B_1B_2 \to \lambda, B_3B_4 \to \lambda$, respectively; the terminal rules of G are simulated by the axioms SxS in A. The context-free derivations in G are simulated in γ in the reverse order, starting from the center of the produced string (from the substring introduced by a rule $S \to x$) towards the ends.

For instance, assume that we have a string of the form SwS with $w \in (T \cup \{B_1, B_2, B_3, B_4\})^*$; the axioms SxS are of this type. If we apply a splicing rule $r_1 = S\#\$Su\#ZvS$, associated with some rule $S \to uSv \in P$, then we get

$$(S|wS, Su|ZvS) \vdash_{r_1} (SZvS, SuwS).$$

We have to continue; because no symbol X, Y, Z' is present, the only possibility is to use the rule $r_2 = SZ\#vS\$\#S$ associated with the same rule $S \to uSv \in P$:

$$(SZ|vS, Suw|S) \vdash_{r_2} (SZS, SuwvS).$$

The double splicing

$$(SwS, SuZvS) \vdash_{r_1, r_2} (SZS, SuwvS)$$

has simulated the use of the rule $S \to uSv$ in the reverse order.

(The reader might check that starting with a double splicing $(SwS|, Su|ZvS) \vdash_{r_1} (SwSZ|vS, |Su) \vdash_{r_1} (SwSZZu, vS)$ does not lead to terminal strings.)

If to a string SwS we apply the rule $r_1 = S\#\$\#Z'$, then we have to continue with the rule $r_2 = SZ'\#\$\#S$:

$$(S|wS, |Z') \vdash_{r_1} (SZ'|, w|S) \vdash_{r_2} (SZ'S, w).$$

The occurrences of S from the ends of the string are removed (this means that from now on no further rule of the form $S \to uSv \in P$ can be simulated in γ starting from the string w).

If to a string w, bounded or not by occurrences of S, we apply the splicing rule $r_1 = B_1\#B_2\$X\#Y$ (providing that a substring B_1B_2 appears in w, that is $w = xB_1B_2y$), then we have to continue with the rule $r_2 = \#B_1Y\$XB_2\#$ (no other rule is applicable to the intermediate strings), hence we get:

$$(xB_1|B_2y, X|Y) \vdash_{r_1} (x|B_1Y, XB_2|y) \vdash_{r_2} (xy, XB_2B_1Y).$$

The occurrence of B_1B_2 specified above is removed from the input string.

The same assertions are true if we apply first the rule $B_3\#B_4\$X\#Y$; an occurrence of the substring B_3B_4 is removed.

The strings $SZS, SZ'S$ cannot enter splicings leading to terminal strings and this can be easily seen. If a string XB_2B_1Y, XB_4B_3Y enters new splicings, they produce nothing new. For instance, for $r = \#B_1Y\$XB_2\#$ we get:

$$(XB_2|B_1Y, XB_2|B_1Y) \vdash_r (XB_2|B_1Y, XB_2|B_1Y) \vdash_r (XB_2B_1Y, XB_2B_1Y).$$

No double splicing of a type different from those discussed above can lead to terminal strings. Consequently, the double splicing operations in γ correspond to using context-free rules in P, to removing two occcurrences of S from the ends of a string, or to using

the erasing rules $B_1B_2 \to \lambda, B_3B_4 \to \lambda$. The order of using these rules is irrelevant. Consequently, $L(G) = L_d(\gamma)$. □

For a splicing rule $r = u_1\#u_2\$u_3\#u_4$ we denote $rad(r) = \max\{|u_i| \mid 1 \leq i \leq 4\}$; this is the *radius* of the rule r. Then, if $\gamma = (V, T, A, R)$ is a splicing system, we define $rad(\gamma) = \max\{rad(r) \mid r \in R\}$. The family of languages $L_d(\gamma)$ generated by extended H systems of radius at most k and with axioms in a family F is denoted by $EH_d(F, [k])$.

In the previous proof we can modify the "linear" rules $S \to uSv$ of P, replacing them by rules of the forms $D \to \alpha E\beta$, where $\alpha, \beta \in T \cup \{B_1, B_2, B_3, B_4\}$ and $|\alpha\beta| = 1$, in such a way that we obtain a grammar which is equivalent with G, but contains only rules with the right hand side of length two; moreover, we may assume that all rules $D \to \alpha E\beta$ have $D \neq E$; the nonterminal alphabet is now bigger, new symbols are used.

However, a linear grammar with several nonterminal symbols can be simulated by an extended H system using double splicing operations in a way similar to the way we have simulated the context-free rules of the grammar G in the previous proof.

Specifically, consider a linear grammar $G = (N, T, S, P)$ and construct the extended H system $\gamma = (V, T, A, R)$ with

$$\begin{aligned}
V &= N \cup T \cup \{Z, Z'\},\\
A &= \{DxD \mid D \to x \in P, x \in T^*\}\\
&\quad \cup \{D\alpha Z\beta D \mid D \to \alpha E\beta \in P, \text{ where } D, E \in N, \alpha, \beta \in T \cup \{\lambda\}\}\\
&\quad \cup \{Z'\},\\
R &= \{E\#\$D\alpha\#Z\beta,\ EZ\#\beta D\$\#E \mid D \to \alpha E\beta \in P, D, E \in N, \alpha, \beta \in T \cup \{\lambda\}\}\\
&\quad \cup \{S\#\$\#Z',\ SZ'\#\$\#S\}.
\end{aligned}$$

The reader can easily check that the derivations in G are simulated in γ in the reverse order, starting from strings DxD associated to terminal rules $D \to x$ and going back to a string of the form SzS, when the symbols S can be eliminated. Therefore, $L(G) = L_d(\gamma)$. Clearly, $rad(\gamma) = 2$.

Combining this idea with the construction in the proof of Theorem 3 (with the way of simulating erasing rules of the form $B_iB_j \to \lambda$; note that the splicing rules associated with these rules are of radius two), we get an extended H system of radius two. Therefore, we can strenghten the previous theorem by stating it in the following way:

Theorem 4. $RE = EH_d(FIN, [2])$.

This result can probably be replaced by a more precise one by considering the *width* of a splicing rule, as in [13] ($width(u_1\#u_2\$u_3\#u_4) = (|u_1|, |u_2|, |u_3|, |u_4|)$; the width of an H system γ is the smallest vector (n_1, n_2, n_3, n_4) which is componentwise larger than or equal to the width of any splicing rule in γ). We do not insist here in this direction, but we conclude by stressing again the unexpected power of the double splicing.

5. Communicating Distributed H Systems

The model we consider in this section is the splicing counterpart of the parallel communicating grammar systems with communication by command: the components work by splicing and communicate by sending to each other strings which pass certain filters specified in advance.

A *communicating distributed H system* (of degree $n, n \geq 1$) is a construct

$$\Gamma = (V, T, (A_1, R_1, V_1), \ldots, (A_n, R_n, V_n)),$$

where V is an alphabet, $T \subseteq V$, A_i are finite languages over V, R_i are finite sets of splicing rules over V, and $V_i \subseteq V$, $1 \leq i \leq n$.

Each triple $(A_i, R_i, V_i), 1 \leq i \leq n$, is called a *component* of Γ; A_i, R_i, V_i are the set of *axioms*, the set of *splicing rules*, and the *selector* (or *filter*) of the component i, respectively; T is the terminal alphabet of the system. (Note that we consider here the extended form of communicating distributed H systems; in [4] and in the subsequent papers mentioned in the Introduction non-extended systems are considered and the first component has only the role of selecting the terminal strings by means of its filter $V_1 = T$.)

We denote

$$B = V^* - \bigcup_{i=1}^{n} V_i^*.$$

The pair $\sigma_i = (V, R_i)$ is the underlying H scheme associated to the component i of the system.

An n-tuple $(L_1, \ldots, L_n), L_i \subseteq V^*, 1 \leq i \leq n$, is called a *configuration* of the system; L_i is also called the *contents* of the ith component, understanding the components as test tubes where the splicing operations are carried out.

For two configurations $(L_1, \ldots, L_n), (L_1', \ldots, L_n')$, we define

$$\begin{aligned}(L_1, \ldots, L_n) \Longrightarrow (L_1', \ldots, L_n') \text{ iff} \\ L_i' = \bigcup_{j=1}^{n} (\sigma_j^*(L_j) \cap V_i^*) \cup (\sigma_i^*(L_i) \cap B), \\ \text{for each } i, 1 \leq i \leq n.\end{aligned}$$

In words, the contents of each component is spliced according to the associated set of rules (we pass from L_i to $\sigma_i^*(L_i), 1 \leq i \leq n$), and the result is redistributed among the n components according to the selectors $V_1, \ldots, V_n$; the part which cannot be redistributed (does not belong to some $V_i^*, 1 \leq i \leq n$) remains in the component. Because we have imposed no restriction over the alphabets V_i, for example, we did not suppose that they are pairwise disjoint, when a string in $\sigma_j^*(L_j)$ belongs to several languages V_i^*, then copies of this string will be distributed to all components i with this property.

The language generated by Γ is defined by

$$\begin{aligned}L(\Gamma) = \{w \in T^* \mid w \in L_1 \text{ for some } L_1, \ldots, L_n \subseteq V^*, \text{ such} \\ \text{that } (A_1, \ldots, A_n) \Longrightarrow^* (L_1, \ldots, L_n)\}.\end{aligned}$$

That is, the first component of the system is designated as its *master* and the language of Γ is the set of all terminal strings generated (or collected by communications) by the master.

We denote by CDH_n the family of languages generated by communicating distributed H systems of degree at most $n, n \geq 1$. When n is not specified, we replace the subscript n with $*$.

Another possibility is to consider as the language generated by Γ the union of all languages generated by its components, but we do not follow this suggestion here.

Communicating distributed H systems characterize RE. Before proving this result, let us examine an **example**:

Consider the system

$$\Gamma = (V, \{a, b, c\}, (A_1, R_1, V_1), (A_2, R_2, V_2)),$$
$$V = \{a, b, c, X, X', Y, Y', Z\},$$

$$A_1 = \{XY, X'aZ, ZbY', cZ, Zc\},$$
$$R_1 = \{X\#\$X'a\#Z,\ \#Y\$Z\#bY',\ X\#\$c\#Z,\ \#Y\$Z\#c,\ \#Y\$X\#\},$$
$$V_1 = \{a, b, X, Y\},$$

$$A_2 = \{XZ, ZY\},$$
$$R_2 = \{X'\#\$X\#Z,\ \#Y'\$Z\#Y\},$$
$$V_2 = \{a, b, X', Y'\}.$$

Starting from a string XwY with $w \in \{a, b\}^*$ (for the axiom XY we have $w = \lambda$), in the first component we can perform

$$(X|wY, X'a|Z)) \vdash (XZ, X'awY),$$
$$(X'aw|Y, Z|bY') \vdash (X'awbY', ZY).$$

The two operations can be also performed in the reverse order. One further occurrence of a and one further occurrence of b can be added in this way to the string w and at the same time, X, Y are replaced by X', Y', respectvely. Moreover, two strings Xw_1Y, Xw_2Y with $w_1, w_2 \in \{a, b\}^*$, can be spliced by

$$(Xw_1|Y, X|w_2Y) \vdash (Xw_1w_2Y, XY).$$

The obtained string can enter splicings of the first type. A string bounded by X', Y' cannot enter further splicings in the first component, but it can be communicated to the second one. Here we can perform splicings of the forms:

$$(X'|xY', X|Z) \vdash (X'Z, XxY'),$$
$$(Xx|Y', Z|Y) \vdash (XxY, ZY').$$

The string XxY can be communicated to the first component, hence the process can be iterated.

That is, pairs of symbols a, b can be added at the ends of a string or any two strings can be concatenated in the sense described above.

At any moment, in the first component we can also replace X, Y with c. If only one of X, Y is replaced by c (and the other one is replaced by its primed version), then the string is "lost": the remaining marker X', Y' which was not replaced by c cannot be removed, because the string cannot be communicated. Thus, both X and Y must be replaced by c at the same time and this ends the derivation.

Consequently, we obtain

$$L(\Gamma) = \{cwc \mid w \in D_{a,b}\},$$

where $D_{a,b}$ is the Dyck language over $\{a, b\}$. This is not a linear language (it is a context-free language of infinite index).

Thus, $CDH_2 - LIN \neq \emptyset$.

The following result was first proved in [24]. We give here a new proof, also bounding the radius of the obtained system.

Theorem 5. *$CDH_n = CDH_* = RE$, for all $n \geq 3$.*

Proof. The inclusions $CDH_n \subseteq CDH_{n+1} \subseteq CDH_* \subseteq RE, n \geq 1$, are obvious. We have only to prove the inclusion $RE \subseteq CDH_3$.

Consider a type-0 grammar $G = (N, T, S, P)$, take a new symbol, B, and denote, for an easy reference,

$$N \cup T \cup \{B\} = \{D_1, D_2, \ldots, D_n\}.$$

Because $N \neq \emptyset, T \neq \emptyset$, we have $n \geq 3$. We construct the communicating distributed H system

$$\Gamma = (V, T, (A_1, R_1, V_1), (A_2, R_2, V_2), (A_3, R_3, V_3)),$$

with

$$\begin{aligned}
V &= N \cup T \cup \{X, Y, X', Y', Z, B\} \\
&\quad \cup \{X_i, Y_i \mid 0 \leq i \leq 2n\},
\end{aligned}$$

$$\begin{aligned}
A_1 &= \{XBSY\} \cup \{ZvY \mid u \to v \in P\} \\
&\quad \cup \{X_{2i}D_iZ \mid 1 \leq i \leq n\} \\
&\quad \cup \{X_{2i}Z \mid 0 \leq i \leq n-1\} \\
&\quad \cup \{ZY_{2i} \mid 0 \leq i \leq n\}, \\
R_1 &= \{\#uY\$Z\#vY \mid u \to v \in P\} \\
&\quad \cup \{\#D_iY\$Z\#Y_{2i},\ X\#\$X_{2i}D_i\#Z \mid 1 \leq i \leq n\} \\
&\quad \cup \{\#Y_{2i+1}\$Z\#Y_{2i},\ X_{2i+1}\#\$X_{2i}\#Z \mid 0 \leq i \leq n-1\}, \\
V_1 &= N \cup T \cup \{X, Y, B\} \cup \{X_{2i+1}, Y_{2i+1} \mid 0 \leq i \leq n-1\},
\end{aligned}$$

$$\begin{aligned}
A_2 &= \{ZY_{2i-1}, X_{2i-1}Z \mid 1 \leq i \leq n\} \cup \{ZZ\}, \\
R_2 &= \{\#Y_{2i}\$Z\#Y_{2i-1},\ X_{2i}\#\$X_{2i-1}\#Z \mid 1 \leq i \leq n\} \\
&\quad \cup \{X'B\#\$\#ZZ,\ \#Y'\$ZZ\#\}, \\
V_2 &= N \cup T \cup \{B, X', Y'\} \cup \{X_{2i}, Y_{2i} \mid 1 \leq i \leq n\},
\end{aligned}$$

$$\begin{aligned}
A_3 &= \{ZY, XZ, ZY', ZX'\}, \\
R_3 &= \{\#Y_0\$Z\#Y,\ X_0\#\$X\#Z,\ \#Y_0\$Z\#Y',\ X_0\#\$X'\#Z\}, \\
V_3 &= N \cup T \cup \{B, X_0, Y_0\}.
\end{aligned}$$

Let us examine the work of Γ. The underlying idea is rotate-and-simulate, used first in [14] and then in several subsequent papers. Starting from strings of the form XwY (the axiom $XBSY$ is of this form), the first component can simulate the rules of P in a suffix of

w, by using splicing rules $\#uY\$Z\#vY$, for $u \to v \in P$, or can start rotating the string. In the first case, the string obtained is again bounded by the markers X, Y, hence the process can be iterated. When removing a symbol D_i from the right hand end of w one replaces Y with Y_{2i}:

$$(Xw_1|D_iY, Z|Y_{2i}) \vdash (Xw_1Y_{2i}, ZD_iY),$$

providing that $w = w_1D_i, 1 \leq i \leq n$ (observe that B can be removed like any symbol in $N \cup T$).

No string containing an occurrence of Z can be moved from a component to another one. If such strings obtained by splicings enter new splicing operations, then no terminal string can be obtained using the resulting strings: both of them contain the symbol Z and by splicing them no new string is obtained. Consider, for instance, the string ZD_iY. Using again the rule $\#D_i\$Z\#Y_{2i}$ we obtain the strings ZY_{2i}, ZD_iY. A similar result will be obtained in all cases below.

The string Xw_1Y_{2i} cannot be communicated, but a further splicing is possible in the first component:

$$(X|w_1Y_{2i}, X_{2j}D_j|Z) \vdash (XZ, X_{2j}D_jw_1Y_{2i}),$$

for some $1 \leq j \leq n$. The two operations can be performed in the reverse order and the result is the same.

Strings bounded by markers X_r, Y_s with even r, s cannot enter new splicings in the first component and can be communicated to the second component. Two splicings are possible here, decreasing by one the subscripts of X and Y. If only one splicing is performed, then the string cannot be communicated. Thus, we get:

$$(X_{2j}|D_jw_1Y_{2i}, X_{2j-1}|Z) \vdash (X_{2j}Z, X_{2j-1}D_jw_1Y_{2i}),$$
$$(X_{2j-1}D_jw_1|Y_{2i}, Z|Y_{2i-1}) \vdash (X_{2j-1}D_jw_1Y_{2i-1}, ZY_{2i}).$$

Again, the order of the two operations is not important.

A string with odd subscripts of the end markers can be communicated to the first component. These operations can be iterated and they must be continued, otherwise there is no way to remove the nonterminal symbols. When in the first component we obtain X_0 or Y_0, the string can no longer be communicated to the second component. If one of the end markers X, Y has the subscript 0 and the other subscript is strictly larger, then the string is "lost", it cannot be communicated and it cannot enter new splicings. If both markers have the subscript 0, then the string can be communicated to the third component.

In the third component, a string of the form X_0wY_0 can be transformed to XwY (and this string is passed to the first component, thus making possible the iteration of the whole process, of simulation of rules in P or of rotation), or to $X'wY'$, or to a string with mixed forms of the markers X, Y, with and without a prime. In the last case, the string is again "lost", it cannot be further processed.

A string of the form $X'wY'$ can be communicated only to the second component, where only two splicings are possible:

$$(X'B|w_1Y', |ZZ) \vdash (X'BZZ, w_1Y'),$$
$$(w_1|Y', ZZ|) \vdash (w_1, ZZY'),$$

providing that $w = Bw_1$ (which ensures that the string has the same permutation as the corresponding string produced by G). A string without end markers cannot enter new

splicings. If it is a terminal one, then it can be communicated to the first component, hence it is an element of $L(\Gamma)$; otherwise it is "lost".

Therefore, the subscripts of the two markers must reach at the same time the value 0. This is possible only when they have started from the same value. In the case above, we must have $i = j$. This means that exactly the symbol D_i which has been erased from the right end of w has been simultaneously introduced in the left end of w. In this way, the rotation phase is correctly implemented, hence all circular permutations of the string can pe obtained. Consequently, all derivations in G can be simulated in Γ and, conversely, only strings in $L(G)$ can be sent as terminal strings to the first component of Γ. Thus, $L(G) = L(\Gamma)$. □

If we start the previous construction from a grammar in Kuroda normal form, then the radius of the obtained system is 3 (reached in simulating rules in R_1). Using the same idea as in [15], one can easily prove that, in fact, rules of radius two suffice.

Communicating distributed H systems of degree 1 do not use communication, hence they are extended finite H systems. In view of Theorem 1, we can write

$$CDH_1 = REG \subset CDH_2$$

(the properness of the second inclusion is proved by the example considered before Theorem 5).

It is an *open problem* whether or not the inclusion $CDH_2 \subseteq CDH_3$ is proper, hence whether or not the result in Theorem 5 can be strengthened, to $n = 2$. We expect a negative answer. (We *conjecture* that $CDH_2 \subset CF$.)

This problem is mainly interesting from a mathematical point of view, not too much for DNA computing: the motivation of considering distributed H systems is to decrease the number of splicing rules used in each component; a small number of components intuitively means components of large size, which is against our goal.

Consider now the very problem which has motivated the definition of distributed H systems – limiting the number of splicing rules working together. For a communicating distributed H system $\Gamma = (V, T, (A_1, R_1, V_1), \ldots, (A_n, R_n, V_n))$ we denote by $tubes(\Gamma)$ the degree of Γ (the number n, of components), by $rad(\Gamma)$ the maximum radius of rules in Γ, and

$$size(\Gamma) = \max\{card(R_i) \mid 1 \leq i \leq n\}.$$

One can characterize the family RE by communicating distributed H systems of minimal size (of course, this is obtained at the expense of leaving the number of components unbounded). Proofs of the following two theorems can be found in [19].

Theorem 6. *For each type-0 grammar $G = (N, T, S, P)$ we can construct a communicating distributed H system Γ such that $L(G) = L(\Gamma)$ and*

$$tubes(\Gamma) = 2(card(N \cup T) + 1) + card(P) + 9,$$
$$size(\Gamma) = 1,$$
$$rad(\Gamma) = card(N \cup T) + 2.$$

At the price of increasing the number of components, we can also bound the radius of the obtained system.

Theorem 7. *For each type-0 grammar $G = (N, T, S, P)$ we can construct a communicating distributed H system Γ such that $L(G) = L(\Gamma)$ and*

$$tubes(\Gamma) \leq 3(card(N \cup T) + 1) + 2 \cdot card(P) + 5,$$
$$size(\Gamma) = 1,$$
$$rad(\Gamma) = 2.$$

6. Time-Varying Distributed H Systems

The distributed architecture we consider in this section can be viewed as a sequential counterpart of the previous systems: at different moments we use different sets of splicing rules. The passing from a set of rules to another one is now specified in a cycle. Thus, the new model corresponds both to periodically time-varying grammars in regulated rewriting area and to controlled tabled Lindenmayer systems. We can also also interpret these systems as counterparts of cooperating distributed grammar systems with the order of enabling the components controlled by a graph having the shape of a ring.

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

A *time-varying distributed H system* (of degree $n, n \geq 1$), [17], is a construct

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

where V is an alphabet, $T \subseteq V$ (terminal alphabet), A is a finite subset of V^* (axioms), and R_i are finite sets of splicing rules over $V, 1 \leq i \leq n$. The sets $R_i, 1 \leq i \leq n$, are called the components of the system.

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

$$L_1 = A,$$
$$L_{k+1} = \sigma_i(L_k), \text{ for } i \equiv k(mod\ n), k \geq 1,$$

where $\sigma_i = (V, R_i), 1 \leq i \leq n$.

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 Γ is defined by

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

We denote 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.

The way of working of a time-varying H systems is surprisingly strong. (The explanation lies in the fact that from a step to another step one passes only the result of splicing operations done at the previous step; strings produced at different "generations" cannot be spliced together.)

For **example**, let us consider the system (of degree 1)

$$\Gamma = (\{a,b,c\}, \{a,b,c\}, \{cab\}, \{a\#b\$c\#a\}).$$

We obtain

$$\begin{aligned}
&L_1 = \{cab\},\\
&L_2 = \{caab, cb\}, \text{ by } (ca|b, c|ab) \vdash (caab, cb),\\
&L_3 = \{ca^4b, cb\}, \text{ by } (caa|b, c|aab) \vdash (ca^4b, cb),\\
&\ldots\ldots\ldots\ldots\\
&L_k = \{ca^{2^{k-1}}b, cb\},\ k \geq 1.
\end{aligned}$$

Therefore,

$$L(\Gamma) = \{ca^{2^n}b \mid n \geq 0\} \cup \{cb\},$$

which is a non-context-free language.

Because each regular language can be generated by a time-varying H system of degree 1 (follow the same construction as in [20], adding splicing rules which pass the axioms from a step to the next one; because the axioms are of a well specified form, this can be easily achieved), we have

Lemma 2. $REG = EH(FIN, FIN) \subset VDH_1 \subseteq VDH_2 \subseteq \ldots \subseteq VDH_* \subseteq RE$.

This hierarchy collapses (at most) at level 7 (we do not know whether or not this result is optimal):

Theorem 8. $VDH_n = VDH_* = RE,\ n \geq 7$.

Proof. Consider a type-0 grammar $G = (N, T, S, P)$ with $N \cup T = \{\alpha_1, \ldots, \alpha_{n-1}\}, n \geq 3$, and $P = \{u_i \to v_i \mid 1 \leq i \leq m\}$. Let $\alpha_n = B$ be a new symbol. We construct the time-varying distributed H system

$$\Gamma = (V, T, A, R_1, \ldots, R_7),$$

with

$$\begin{aligned}
V &= N \cup T \cup \{X, Y, Y', Z, B\}\\
&\quad \cup \{Y_i, Y_i', X_i \mid 0 \leq i \leq n\},\\
A &= \{XBSY, ZY, ZY', ZZ\}\\
&\quad \cup \{Zv_iY \mid 1 \leq i \leq m\}\\
&\quad \cup \{ZY_j, ZY_j', X_j\alpha_jZ, X_jZ \mid 1 \leq j \leq n\},
\end{aligned}$$

and the following sets of splicing rules:

$$R_1 = \{\#u_iY\$Z\#v_iY \mid 1 \leq i \leq m\}$$

$$\cup \{\#Y\$Z\#Y,\ Z\#\$Z\#\}$$
$$\cup \{\#Y_j\$Z\#Y_j \mid 1 \leq j \leq n\},$$
$$R_2 = \{\#\alpha_j Y\$Z\#Y_j \mid 1 \leq j \leq n\}$$
$$\cup \{\#Y\$Z\#Y',\ Z\#\$Z\#\}$$
$$\cup \{\#Y_j\$Z\#Y_j' \mid 1 \leq j \leq n\},$$
$$R_3 = \{X\#\$X_j\alpha_j\#Z \mid 1 \leq j \leq n\}$$
$$\cup \{\#Y'\$Z\#Y,\ Z\#\$Z\#\}$$
$$\cup \{\#Y_j'\$Z\#Y_j \mid 1 \leq j \leq n\},$$
$$R_4 = \{\#Y_j\$Z\#Y_{j-1} \mid 1 \leq j \leq n\}$$
$$\cup \{\#Y\$Z\#Y,\ Z\#\$Z\#\},$$
$$R_5 = \{X_j\#\$X_{j-1}\#Z \mid 1 \leq j \leq n\}$$
$$\cup \{\#Y\$Z\#Y,\ Z\#\$Z\#\},$$
$$R_6 = \{\#Y_0\$Z\#Y,\ \#Y_0\$ZZ\#,\ \#Y\$Z\#Y',\ Z\#\$Z\#\}$$
$$\cup \{\#Y_j\$Z\#Y_j' \mid 1 \leq j \leq n\},$$
$$R_7 = \{X_0\#\$X\#Z,\ X_0B\#\$\#ZZ,\ \#Y'\$Z\#Y,\ Z\#\$Z\#\}$$
$$\cup \{\#Y_j'\$Z\#Y_j \mid 1 \leq j \leq n\}.$$

This system works as follows.

Consider a string of the form $XwY, w \in (N \cup T \cup \{B\})^*$; for the axiom $XBSY$ we have $w = BS$.

If $w = w'u_i, 1 \leq i \leq m$, then the first component can simulate the rule $u_i \to v_i \in P$ for a suffix of w. A string XwY can also be passed to R_2 unmodified, by using the rule $\#Y\$Z\#Y$. Similarly, by using the rule $Z\#\$Z\#$, any axiom (in general, any string containing an occurrence of Z) can be passed from R_1 to R_2 – and the same assertion is true for all consecutive components).

A string XwY can enter in R_2 two splicings:

$$(Xw'|\alpha_j Y, Z|Y_j) \vdash (Xw'Y_j, Z\alpha_j Y), \text{ for } w = w'\alpha_j, 1 \leq j \leq n,$$
$$(Xw|Y, Z|Y') \vdash (XwY', ZY).$$

The string $Xw'Y_j$ can enter only one splicing in R_3:

$$(X|w'Y_j, X_i\alpha_i|Z) \vdash (XZ, X_i\alpha_i w'Y_j) \qquad (*)$$

for some $i, 1 \leq i \leq n$.

A string of the form $X_i x Y_j, 1 \leq i, j \leq n$, will enter splicings in R_4, R_5 which will decrease by one each of i and j, thus producing $X_{i-1}xY_{j-1}$.

A string $X_i x Y_j, 1 \leq i, j \leq n$, will be transformed in R_6 into $X_i x Y_j'$ and this one will be transformed in R_7 into $X_i x Y_j$. R_1 will pass such a string unmodified to R_2 which will again replace Y_j by Y_j'; R_3 will return to $X_i x Y_j$. The components R_4, R_5 will again decrease by one the subscripts of X and Y. Eventually, one of X, Y will get the subscript 0. We have three possibilities:

1) R_6 receives a string $X_0 x Y_j$ with $j \geq 1$. The only applicable rule is $\#Y_j\$Z\#Y_j'$; the string $X_0 x Y_j'$ is passed to R_7 which returns to $X_0 x Y_j$; again Y_j is replaced by Y_j', then R_3

returns to X_0xY_j which reaches R_4. R_4 produces X_0xY_{j-1}. No splicing can be done in R_5 on such a string, hence no terminal string is obtained in this way.

2) R_6 receives a string X_ixY_0 with $j \geq i$. If Y_0 is replaced by Y, then the string X_ixY cannot be spliced in R_7. The same assertion is true if Y_0 is deleted. No terminal string can be produced in this way.

3) R_6 receives a string X_0xY_0. (This means that the string $X_i\alpha_iw'Y_j$ obtained after the splicing $(*)$ has $i = j$, hence the same symbol α_j which was deleted from the right hand end of the string has been introduced in the left hand end.) If R_6 replaces Y_0 by Y, then the only continuation in R_7 is to replace X_0 by X, hence the whole process can be iterated. If R_6 removes Y_0 and R_7 replaces X_0 by X, then the obtained string cannot pass over R_1, hence it is lost. If R_6 removes Y_0 and R_7 removes X_0B, then we get a string without markers, which cannot enter further splicings. If it is terminal, then it belongs to $L(\Gamma)$, otherwise it is lost.

Consequently, every derivation in G can be simulated in Γ by a standard simulate-and-rotate procedure, that is, $L(G) \subseteq L(\Gamma)$.

Assume now that R_2 has produced the string XwY'. If R_3 replaces Y' by Y, then the string XwY will pass unchanged through R_4, R_5, then R_6 will produce XwY' and R_7 will return to XwY, and we arrive back to R_1 with XwY.

If XwY' is spliced in R_3 by a rule $X\#\$X_j\alpha_j\#Z$, $1 \leq j \leq n$, then we get the string $X_j\alpha_jwY'$. Such a string is blocked by R_4, where it cannot be spliced any more.

The strings obtained by the splicings mentioned above and containing occurrences of Z can pass from a component to another one due to the rules $Z\#\$Z\#$ (and also to rules using symbols Y, Y', etc). If such strings enter further splicings, this will happen only together with other strings containing occurrences of Z, either axioms or by-products of other splicings. Thus, both the resulting strings will contain occurrences of Z, hence no terminal string can be produced in this way.

For instance, after a splicing in R_2 using a rule $\#\alpha_jY\$Z\#Y_j$, $1 \leq j \leq n$, we get the string $Z\alpha_jY$. It can pass unmodified through $R_3 - R_7$, but in R_1 we can perform

$$(Z|\alpha_jY, Z|v_iY) \vdash (Zv_iY, Z\alpha_jY),$$

if $\alpha_j \to v_i$ is the ith rule of P. The input strings are reproduced.

The reader can trace the development of other strings of the type of $Z\alpha_jY$ above, and the result will be similar: no terminal string which is not in $L(G)$ can be produced. In conclusion, $L(G) = L(\Gamma)$. □

The constant 7 in the equality $RE = VDH_7$ can probably be replaced by a smaller integer. We do not insist into this direction, because of the motivation we have started with: diminishing the *size* of the components. This is possible also for time-varying distributed H systems; a proof of the following result can be found in [17].

Theorem 9. *Each recursively enumerable language can be generated by a time-varying distributed H system whose components contain at most three splicing rules.*

7. Concluding Remarks

By using the previous proofs, which are based on effective constructions, *universal* H systems of the mentioned types are obtained: just start from a universal type-0 grammar

and follow the above constructions. (An explicit universal type-0 grammar can be found in [1] or can be obtained starting from a universal Turing machine – in particular, from *small* Turing machines, as those in [25] – and constructing its associated type-0 grammar, for instance, as in [27].) This result theoretically proves that "universal programmable DNA computers based on splicing" can be designed in the form of an H system of the types considered in the previous sections.

In [17] and in [18] one also discusses another class of distributed H systems, called *two-level distributed H systems*: the components of the system have their own splicing rules, but there also exists a set of splicing rules at the level of the system; each component has two types of strings, "active strings" and "not so active strings"; the system splicing rules are applied with priority to the active strings of the components and only when no such splicing is possible, a local splicing is performed, in a component. Details can be found in [17], [18]. Again a characterization of recursively enumerable languages is obtained, by two-level H systems with three components (while the case of two components is open, too).

References

[1] C. Calude, Gh. Păun, Global syntax and semantics for recursively enumerable languages, *Fundamenta Informaticae*, 4, 2 (1981), 245 – 254.

[2] E. Csuhaj-Varju, J. Dassow, J. Kelemen, Gh. Păun, *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.

[3] E. Csuhaj-Varju, L. Freund, L. Kari, Gh. Păun, DNA computing based on splicing: universality results, *First Annual Pacific Symp. on Biocomputing*, Hawaii, Jan. 1996 (L. Hunter, T. E. Klein, eds.), World Sci. Publ., Singapore, 1996.

[4] E. Csuhaj-Varju, L. Kari, Gh. Păun, Test tube distributed systems based on splicing, *Computers and AI*, 15, 2-3 (1996), 211 – 231.

[5] K. Culik II, T. Harju, Splicing semigroups of dominoes and DNA, *Discrete Appl. Math.*, 31 (1991), 261 – 277.

[6] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, Heidelberg, 1989.

[7] J. Dassow, Gh. Păun, G. Rozenberg, Grammars systems, chapter 4 in vol. 2 of [26], 155 – 213.

[8] R. Freund, L. Kari, Gh. Păun, DNA computing based on splicing: The existence of universal computers, *Technical Report 185-2/FR-2/95*, TU Wien, 1995, and *Theories of Computer Science*, in press.

[9] V. Geffert, Normal forms for phrase-structure grammars, *RAIRO. Th. Inform. and Appl.*, 25 (1991), 473 – 496.

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

[11] T. Head, Wet splicing systems, *Proc. of Workshop on Molecular Computing*, Mangalia, 1997 (Gh. Păun, ed.), in preparation.

[12] T. Head, Gh. Păun, D. Pixton, Language theory and molecular genetics. Generative mechanisms suggested by DNA recombination, chapter 7 in vol. 2 of [26], 295 – 360.

[13] A. Păun, Extended H systems with permitting contexts of small radius, *Fundamenta Informaticae*, 31 (1997).

[14] Gh. Păun, Regular extended H systems are computationally universal, *J. Automata, Languages, Combinatorics*, 1, 1 (1996), 27 – 36.

[15] Gh. Păun, Computing by splicing: How simple rules ?, *Bulletin of the EATCS*, 60 (1996), 144 – 150.

[16] Gh. Păun, Splicing systems with targets are computationally complete, *Inform. Processing Letters*, 59 (1996), 129 – 133.

[17] Gh. Păun, DNA computing: Distributed splicing systems, in vol. *Structures in Logic and Computer Science. A Selection of Essays in Honor of A. Ehrenfeucht* (J. Mycielski, G. Rozenberg, A. Salomaa, eds.), *Lecture Notes in Computer Science* 1261, Springer-Verlag, 1997, 353 – 370.

[18] Gh. Păun, Two-level distributed H systems, *Proc. of Developments in Language Theory Conf.*, Thessaloniki, 1997.

[19] Gh. Păun, Distributed architectures in DNA computing based on splicing: Limiting the size of components, *Proc. of Conf. on Unconventional Methods of Computing*, Auckland, 1998 (C. Calude, ed.), World Sci. Publ., Singapore, 1998.

[20] Gh. Păun, G. Rozenberg, A. Salomaa, Computing by splicing, *Theoretical Computer Sci.*, 168, 2 (1996), 321 – 336.

[21] Gh. Păun, G. Rozenberg, A. Salomaa, Computing by splicing. Programmed and evolving splicing systems,*1997 IEEE Intern. Conf. on Evolutionary Computing,* Indianapolis, 1997, 273 – 277.

[22] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing: New Computing Paradigms*, Springer-Verlag, Heidelberg, 1998.

[23] D. Pixton, Regularity of splicing languages, *Discrete Appl. Math.*, 69 (1996), 101 – 124.

[24] L. Priese, Y. Rogozhin, M. Margenstern, Finite H systems with 3 tubes are not predictable, *Technical Report 24/97*, Univ. Koblenz-Landau, Institut für Informatik, 1997.

[25] Y. Rogozhin, Small universal Turing machines, *Theoretical Computer Sci.*, 168 (1996), 215 – 240.

[26] G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages*, 3 volumes, Springer-Verlag, Heidelberg, 1997.

[27] A. Salomaa, *Formal Languages*, Academic Press, New York, London, 1973.

[28] P. Turakainen, A unified approach to characterizations of recursively enumerable languages, *Bulletin of the EATCS*, 45 (1991), 223 – 228.

[29] C. Zandron, C. Ferretti, G. Mauri, A reduced distributed splicing system for RE languages, in vol. *New Trends in Formal Languages: Control, Cooperation, Combinatorics.* (Gh. Păun, A. Salomaa, eds.), *Lecture Notes in Computer Science* 1218, Springer-Verlag, 1997, 319 – 329.