Computer Science Journal of Moldova, vol.**6**, no.2(17), 1998

Arithmetically Controlled H Systems

V. Manca Gh. Păun

Abstract

We consider two classes of restricted H systems, both dealing with numbers associated to the terms of splicing operations. In one of them, these numbers indicate the *age* of the strings (the *generation* when the strings are produced), in the second one the numbers can be interpreted as *valences* of the strings. Restricting the splicing to strings of "a similar age", or accepting as complete splicing processes only those processes which produce strings with a null valence increase the generative power of H systems (with finite sets of rules).

1 Introduction

The present paper is a contribution to the recently emerging and highly promising area of DNA Computing; we deal here with the possibility of *computing by splicing*.

The splicing operation was introduced in [4], as a formal counterpart of the recombination of DNA molecules, which are cut by restriction enzymes and, when their sticky ends match, the obtained fragments are ligated in order to produce new molecules. For motivations and for discussions about the abstraction steps made when passing from the biochemical operation to the formal language one, we refer to [4], [5]. Based on this operation, the so-called extended H systems were introduced in [8]. They are generative mechanisms which start from a given set of strings (axioms) and produce a language by iterated splicing according to a given set of splicing rules.

Because H systems with finite sets of axioms and of rules generate only regular languages, [1], [10], many restrictions in the use of splicing

rules were considered, in the aim to increase the generative power. In all cases investigated so far, characterizations of recursively enumerable languages are obtained, see [9]. This is important from the point of view of DNA computing: controlled H systems can be viewed as theoretical models of universal programmable DNA based computers.

In usual H systems, strings produced at various "generations" can be spliced together; for instance, we can splice x, y producing w and then we can splice w with any of x and y. Restricting the splicing to strings of "a similar age" is both natural and a possible new control on the work of H systems. As it is expected, such a restriction increases the power of H systems with finite sets of axioms and of rules, but we do not know whether or not this leads again to a characterization of recursively enumerable languages.

A similar result (and open problem) holds for a variant of such an "arithmetical" control on the splicing operation, based on *valences* associated with the strings: one starts with given valences associated with the axioms and one accepts a string only if it is "neutral" (its valence is zero – in the case when the valences are integers and they are added during the splicing process).

2 H Systems

Consider an alphabet V and two symbols $\#, \$$ not in V. A *splicing rule* over V is a string $r = u_1\#u_2\$u_3\#u_4$, where $u_1, u_2, u_3, u_4 \in V^*$ (V^* is the free monoid generated by V under the operation of concatenation; the empty string is denoted by λ and the length of $x \in V^*$ is denoted by $|x|$). For such a rule r and for $x, y, w \in V^*$ we define

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

(One cuts the strings x, y in between u_1, u_2 and u_3, u_4, respectively, and one concatenates the "first half" of x with the "second half" of y.) We say that we *splice* the strings x, y at the *sites* u_1u_2, u_3u_4, respectively. When r is understood, we write $\vdash$ instead of $\vdash_r$. For clarity, we usually

indicate by a vertical bar the place of splicing: $(x_1u_1|u_2x_2, y_1u_3|u_4y_2) \vdash x_1u_1u_4y_2$.

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

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

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 $A \in F_1, R \in F_2$ (we use to say that γ is *of type* (F_1, F_2)), where F_1, F_2 are two given families of languages. (Note that R is a language, hence the definition makes sense.)

By *FIN, REG, CF, CS, RE* we denote the families of finite, regular, context-free, context-sensitive, and of recursively enumerable languages, respectively; for further elements of formal language theory we refer to [11].

In what follows, all families we consider are supposed to belong to the hierarchy *FIN, REG, CF, CS, RE*.

Two basic results concerning the power of extended H systems are the following (see details and references in [9]).

Theorem 1. (i) $EH(FIN, FIN) = EH(REG, FIN) = REG$. (ii) $EH(CF, FIN) = CF$.

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

From the DNA computing point of view, we need finite systems with a generative power as large as possible (if possibly, characterizing the power of Turing machines) and also having universality properties (containing universal systems, that is systems able to simulate any given system if a code of the particular system is introduced in the universal one). To this aim, several classes of H systems with a controlled splicing operation were considered, in general inspired from the regulated rewriting area and from grammar systems area. Details and complete references can be found in [9]. Two further restrictions, of a new type, are introduced here.

3 H Systems with Age Restrictions

In the splicing operation as defined in the previous section, at any step one can splice strings of different "ages", for instance, a string with a string obtained from the same string after an arbitrarily long sequence of splicings. It is, however, natural to restrict this freedom in choosing the terms of a splicing operation, by bounding the "age difference" by a given constant. This leads to the following definition.

Let $\gamma = (V, T, A, R)$ be an extended H system. For a rule $r \in R$ and for (x, i_1), $(y, i_2), (w, i_3) \in V^* \times \mathbf{N}$ ($\mathbf{N}$ is the set of natural numbers) we define

$$[(x, i_1), (y, i_2)] \vdash_r (w, i_3) \text{ iff } (x, y) \vdash_r w, \text{ and } i_3 = \max(i_1, i_2) + 1.$$

Let us associate the integer 0 to each string $x \in A$, thus considering the set $\{(x,0) \mid x \in A\}$. By iteratively applying this kind of splicing, we obtain pairs $(w,i) \in V^* \times \mathbf{N}$. We say that w in such a pair (w,i) has been *generated* in γ. A splicing $[(x,i_1),(y,i_2)] \vdash_r (w,i_3)$ with $|i_1 - i_2| \leq k$, for some $k \in \mathbf{N}$, is said to be *k-restricted*.

For $k \in \mathbf{N}$, the *k-restricted language* generated by γ, denoted by $L_k(\gamma)$, consists of all strings in $L(\gamma)$ which are generated by using only k-restricted splicing steps.

Obviously, we have $L_0(\gamma) \subseteq L_1(\gamma) \subseteq \ldots \subseteq L(\gamma)$. It is also easy to see that $L(\gamma) = \bigcup_{i \geq 0} L_i(\gamma)$.

We denote by $EH_k(F_1,F_2), k \geq 0$, the family of languages $L_k(\gamma)$, for γ an extended H system of type (F_1,F_2).

Lemma 1. *For all families F_1, F_2 and for all $k \geq 0$, we have $EH(F_1,F_2) \subseteq EH_k(F_1,F_2)$.*

Proof. Let $\gamma = (V,T,A,R)$ be an extended H system. Consider two new symbols Z_1, Z_2 and construct the H system

$$\gamma' = (V \cup \{Z_1,Z_2\}, T, A', R'),$$

where

$$A' = \{Z_1x, xZ_2 \mid x \in A\},$$
$$R' = R \cup \{Z_1\#\$Z_1\#,\ \#Z_1\$Z_1\#,\ \#Z_2\$\#Z_2,\ \#Z_2\$Z_2\#\}.$$

The axioms of γ' are of the forms Z_1x, xZ_2, while the strings obtained by using the splicing rules in R are of the forms Z_1x, xZ_2, Z_1xZ_2. Any string of this form can be carried unchanged from a generation to another one by using the rules $Z_1\#\$Z_1\#, \#Z_2\$\#Z_2$. In this way, any splicing operation with respect to the system γ, irrespective which are the ages of the involved strings, can be reproduced in γ' as a splicing among strings of ages as close as necessary (equal, if $k = 0$). From a string of the form Z_1wZ_2 (maybe, one of Z_1, Z_2 is missing) we can pass to w by using the rules $\#Z_1\$Z_1\#, \#Z_2\$Z_2\#$ (for instance,

$[(|Z_1wZ_2, i), (Z_1|wZ_2, i)] \vdash (wZ_2, i+1)$, $[(w|Z_2, i+1), (w|Z_2, i+1)] \vdash (w, i+2)$). Therefore, $L(\gamma) \subseteq L_k(\gamma')$, for all $k \geq 0$. The converse inclusions are obvious (due to the use of different symbols Z_1, Z_2, the splicing rules in $R' - R$ do not lead to strings in V^* which are not in $L(\gamma)$. □

Lemma 2. $EH_0(FIN, FIN) - CF \neq \emptyset$.

Proof. Consider the (non-extended) H system

$$\gamma = (\{a, b\}, \{a, b\}, \{bab\}, \{a\#b\$b\#a\}).$$

We start with a splicing $[(ba|b, 0), (b|ab, 0)] \vdash (baab, 1)$; we have to continue by splicing $baab$ with itself (and we get $(baaaab, 2)$). At any step, we have only one string of that age. By splicing it with itself, in a unique possible way, the number of a occurrences is doubled. Therefore,

$$L_0(\gamma) = \{ba^{2^n}b \mid n \geq 0\}.$$

This language is not context-free. □

Note that the language $L_0(\gamma)$ is not only non-context-free, but it is not even in the family MAT^λ, of languages generated by matrix grammars with arbitrary context-free rules (but without appearance checking): use the result in [3] (the one-letter languages in MAT^λ are regular) and the closure of MAT^λ under arbitrary morphisms, [2].

We were not able to obtain a similarly strong result for $k \geq 1$, but still we have an increase in power with respect to the non-restricted H systems:

Lemma 3. *For all $k \geq 1$ we have $EH_k(FIN, FIN) - REG \neq \emptyset$.*

Proof. Consider the extended H system

$$\gamma = (\{a, b, c, d\}, \{a, b, c\}, \{cad, dbc\}, \{a\#d\$c\#a,\ b\#c\$d\#b,\ a\#d\$d\#b\}).$$

Let us examine the k-restricted splicings with respect to γ. The strings cad and dbc can be spliced separately with themselves, by

the rules $a\#d\$c\#a$, $b\#c\$d\#b$, respectively, and we get the strings $caad, dbbc$. So, we can continue by splicings which use the same rules. Let us consider only the case of strings of the form ca^nd. Because at each step we can use as terms of a splicing the strings of the last generation together with strings of a generation which is with at most k steps older, we have several possibilities. The shortest string which can be obtained at a given generation is that produced by splicing the shortest available strings, fulfilling these conditions. This means that we have to splice the shortest string of the current generation with the shortest string not older than this one with more than k steps. In turn, the longest string which can be obtained at a given generation is produced by splicing the longest string available (at the current generation) with itself. Let us denote by $\varphi(i)$ the number n such that ca^nd is the shortest string produced at the i-th generation, and by $\mu(i)$ the number n such that ca^nd is the longest string produced at the i-th generation.

The next table indicates the initial values of these parameters:

i	$\varphi(i)$	$\mu(i)$
0	1	1
1	2	2
2	3	4
3	4	8
...	...	...
k	$k+1$	2^k
$k+1$	$(k+1)+2$	2^{k+1}
$k+2$	$((k+1)+2)+3$	2^{k+2}
...	...	...

So, we have:

$$\varphi(i) = i+1, \text{ for } i = 0, 1, 2, \ldots, k,$$
$$\varphi(k+i) = \varphi(k+i-1) + \varphi(i-1), \text{ for } i \geq 1,$$
$$\mu(i) = 2^i, \text{ for } i \geq 0.$$

(Note that for $k = 1$ the values of $\varphi(i)$ are the elements in a Fibonacci sequence.)

Identical results are obtained for the shortest and longest strings of the form db^mc.

Now, consider the use of the rule $a\#d\$d\#b$. We can perform

$$[(ca^n|d, i), (d|b^mc, j)] \vdash (ca^nb^mc, \max(i, j) + 1)$$

only if $|i - j| \leq k$. This imposes a restriction on the possible values of n and m. Specifically, for a given n, denote

$$i_1 = \min\{i \mid n \leq \mu(i)\},$$
$$i_2 = \max\{i \mid n \geq \varphi(i)\}.$$

This means that we have

$$\varphi(i_1) \leq n \leq \mu(i_2).$$

(Note that we can have $\varphi(i) \leq n \leq \mu(i)$ for several values of i, hence we can have $i_1 < i_2$.) In other terms, ca^nd can be produced at any generation between i_1 and i_2.

Because the string db^mc must be produced at most k generations before or after producing the string ca^nd, we must have

$$\varphi(i_1 - k) \leq m \leq \mu(i_2 + k).$$

Consequently, $L(\gamma)$ is an infinite set of strings of the form ca^nb^mc, with n, m subject to the restriction discussed above. Such a language is not a regular one (for example, pump a substring of b^m; we obtain strings of the form ca^nb^tc with arbitrarily large t, hence not depending on n; such strings are not in $L(\gamma)$, a contradiction). □

Theorem 3. (i) *For all $k \geq 0$ and all families F_1, F_2 such that $(F_1, F_2) \notin \{(FIN, FIN), (REG, FIN), (CF, FIN)\}$, we have $EH(F_1, F_2) = EH_k(F_1, F_2) = RE$.* (ii) *For all $k \geq 0$ and for $F_1 \in \{FIN, REG\}$, we have $REG = EH(F_1, FIN) \subset EH_k(F_1, FIN)$; moreover, $CF = EH(CF, FIN) \subset EH_0(CF, FIN)$.*

Proof. Combine the equalities in Theorems 1 and 2 with the results in Lemmas 1, 3, and 3. □

It remains as an *open problem* to find a precise characterization of the size of the families $EH_k(F_1, FIN), F_1 \in \{FIN, REG, CF\}$, for $k \geq 0$. For instance, all controlled extended H systems with finite sets of axioms and of splicing rules considered so far characterize the family of recursively enumerable languages, [9]; is this the case also for k-restricted H systems ?

In the systems discussed above, the restriction on the strings age is the same for all splicing rules and it is formulated in terms of the inequation $|i - j| \leq k$. We can consider a more general variant, in the form of an *extended H system with age selection by means of predicates.*

Such a system is a construct $\gamma = (V, T, A, R)$, where V and T are as in a usual H system, A is a finite subset of $V^* \times \{0\}$, and R is a finite set of pairs (r, π_r), where r is a splicing rule over V and π_r is a predicate on $\mathbf{N} \times \mathbf{N}$. For $(x, i_1), (y, i_2), (w, i_3) \in V^* \times \mathbf{N}$ and (r, π_r) in R we write

$$[(x, i_1), (y, i_2)] \vdash_r (w, i_3) \quad \text{iff} \quad \begin{array}{l}(x, y) \vdash_r w, i_3 = \max(i_1, i_2) + 1, \\ \text{and } \pi_r(i_1, i_2) = 1.\end{array}$$

(We compute the ages as above, but we apply a splicing rule only to strings whose ages satisfy the predicate associated with the rule.)

The language generated by γ is defined in the usual way. By $EH_{ls}(F_1, F_2)$ we denote the family of languages generated by H systems with age selection by means of predicates and of type (F_1, F_2).

If all $(r, \pi_r) \in R$ have the same predicate (there is π such that $\pi_r = \pi$ for all $(r, \pi_r) \in R$), then we say that γ has a *global* age selection. The corresponding families of languages are denoted by $EH_{gs}(F_1, F_2)$ (thus, ls stands for "local selection" and gs for "global selection").

Directly from the definitions we obtain:

Lemma 4. *For all families F_1, F_2 and for all $k \geq 0$ we have $EH_k(F_1, F_2) \subseteq EH_{gs}(F_1, F_2) \subseteq EH_{ls}(F_1, F_2)$.*

Consequently, the assertions in Theorem 3 hold true also for extended H systems with (local or global) age selection by means of predicates.

The control by predicates, especially in the local mode, seems to be rather powerful; still, we do not know whether or not a characterization of RE can be found by using such H systems with finite sets of axioms and of rules.

4 H Systems with Valences

A natural variant of the previous restriction in the work of H systems is to consider integer numbers associated with strings and to compute the integers associated with the results of splicing steps in a specified way (in the style of valence grammars introduced in [6]).

Let $(M, \diamond, e)$ be a group (with the operation $\diamond$ and the identity e). An extended *valence H system* over M is a construct $\gamma = (V, T, A, R)$, where V, T, R are as in a usual extended H system and A is a finite subset of $V^* \times M$.

For $(x, v_1), (y, v_2), (w, v_3) \in V^* \times M$ and $r \in R$ we write $[(x, v_1), (y, v_2)] \vdash_r (w, v_3)$ if and only if $(x, y) \vdash_r w$ and $v_3 = v_1 \diamond v_2$. (For a pair $(x, v) \in V^* \times M$ we say that v is the *valence* of x.) Then,

$$L(\gamma) = \{x \in T^* \mid (x, e) \in \sigma^*(A)\}.$$

(We accept all terminal strings whose valence is equal to the identity of the group.)

We work here with the groups of integers and of positive rational numbers, $(\mathbf{Z}, +, 0), (\mathbf{Q}_+, \cdot, 1)$. Accordingly, we call the corresponding H systems *additive* and *multiplicative* H systems, respectively. We denote by $EH_a(F_1, F_2), EH_m(F_1, F_2)$ the families of languages generated by such systems of type (F_1, F_2), respectively (a stands for "additive", m stands for "multiplicative").

Remark 1. Let us note here the similarities and the differences between H systems with age restrictions and those with valences: In both cases we associate numbers with strings and we compute the numbers associated with new produced strings depending on the numbers associated with the strings entering the splicing (in general, we may

consider a mapping $f(n,m)$, which is $\max(n,m)+1$ in the case of H systems with age restriction, $n \diamond m$ in the case of valence H systems, or might be a more complex one). However, the age restriction acts as a selection criterion on the terms of the splicing (the input strings), while the valence restriction is used in choosing the strings which are accepted in the generated language (they must be "neutral" from a "chemical" point of view, which fits very much the biochemical nature of DNA).

Example 1. Consider the system

$$\gamma = (\{a,b,c,d\},\{a,b,c\},\{(cad,-1),(dbc,1)\},$$

$$\{a\#d\$c\#ad,\ db\#c\$d\#b,\ a\#d\$d\#b\}).$$

We obtain

$$L(\gamma) = \{ca^nb^nc \mid n \geq 1\}.$$

Indeed, the only way to obtain a string with a null valence is by performing a splicing $[(ca^n|d,-n),(d|b^nc,n)] \vdash (ca^nb^nc,0)$, by using the rule $a\#d\$d\#b$; strings of the forms $(ca^nd,-n),(db^mc,m),n,m \geq 1$, can be obtained by using the other two rules in R. No splicing can involve a string of the form ca^nb^nc.

The language $L(\gamma)$ is not regular. Therefore, as in the case of H systems with age restrictions, the use of valences increases the power of systems with finite components. □

By associating to axioms valences which are equal to the identity of the group, each usual H system can be considered a valence H system; thus, we obtain:

Lemma 5. *For all families F_1, F_2 we have $EH(F_1,F_2) \subseteq EH_\alpha(F_1,F_2), \alpha \in \{a,m\}$.*

As in the case of valence grammars, [6], the multiplicative valences are at least as powerful as the additive ones.

Lemma 6. $EH_a(F_1,F_2) \subseteq EH_m(F_1,F_2)$, *for all families F_1, F_2.*

Proof. Consider an extended H system $\gamma = (V, T, A, R)$ with additive valences and construct the H system $\gamma' = (V, T, A', R)$ with multiplicative valences, where

$$A' = \{(x, 2^i) \mid (x, i) \in A\}.$$

If, in order to generate a string (w, k) in γ, we use the strings $(x_1, i_1), \ldots, (x_n, i_n)$ as terms of a sequence of splicings (that is, $\sum_{j=1}^{n} i_j = 0$), then in γ' we can perform the same splicings, for the strings $(x_1, 2^{i_1}), \ldots, (x_n, 2^{i_n})$. Therefore, $\Pi_{j=1}^{n} 2^{i_j} = 2^{\sum_{j=1}^{n} i_j} = 2^0 = 1$.

Conversely, a sequence of splicing steps in γ' which leads to a string with the valence equal to 1 corresponds to a sequence of splicing steps in γ producing the same string with the valence 0. That is, $L(\gamma) = L(\gamma')$. □

Combining these results and the relations in Theorems 1, 2, we obtain:

Theorem 4. (i) *For all families F_1, F_2 such that $(F_1, F_2) \notin \{(FIN, FIN), (REG, FIN), (CF, FIN)\}$, we have $EH(F_1, F_2) = EH_a(F_1, F_2) = EH_m(F_1, F_2) = RE$.* (ii) *$REG = EH(F_1, FIN) \subset EH_a(F_1, FIN) \subseteq EH_m(F_1, FIN), F_1 \in \{FIN, REG\}$.*

For the case (CF, FIN) we only have

$$CF = EH(CF, FIN) \subseteq EH_a(CF, FIN) \subseteq EH_m(CF, FIN),$$

but we do not know whether or not these inclusions are proper. However, at least one of them is proper (and probably also the inclusions

$$EH_a(F_1, FIN) \subseteq EH_m(F_1, FIN), F_1 \in \{FIN, REG\},$$

are proper); this follows from the results below, which relate valence grammars and valence H systems.

A (context-free) *valence grammar* is a construct $G = (N, T, S, P)$, where N is the nonterminal alphabet, T is the terminal alphabet, $S \in$

N (the axiom), and P is a finite set of pairs $(A \to x, i)$, with $A \to x$ a (context-free) rule over $N \cup T$ and i the valence of this rule; in the case of additive valence grammars we have $i \in \mathbf{Z}$, in the multiplicative case we have $i \in \mathbf{Q}_+$. A derivation step in such a grammar is of the form $(u, i_1) \Longrightarrow (v, i_2)$, such that $u = u_1 A u_2, v = u_1 x u_2$ for $(A \to x, i) \in P$, and $i_2 = i_1 \diamond i$ ($\diamond$ is the operation of the considered group). We start from $(S, 0)$ in the additive case an from $(S, 1)$ in the multiplicative case; a string is included in the generated language only if its valence is 0 in the additive case and 1 in the multiplicative case.

We denote by $VREG_a, VREG_m$ the families of the languages obtained in this way, using regular grammars with additive and multiplicative valences, respectively. (We do not consider here also context-free valence grammars; they are much more powerful than the regular ones and not useful for our study.)

Here are some results about these families, [6]:

$$VREG_a - REG \neq \emptyset,\ VREG_a \subset CF,$$
$$VREG_a \subset VREG_m,$$
$$VREG_m \text{ is incomparable with } CF.$$

For instance,

$$\{a^n b^n \mid n \geq 1\} \in VREG_a - REG,$$
$$\{a^n b^n a^n \mid n \geq 1\} \in VREG_m - CF,$$
$$\{a^n b^n \mid n \geq 1\}^2 \in (VREG_m \cap CF) - VREG_a.$$

Lemma 7.
$VREG_a \subseteq EH_a(FIN, FIN),\ VREG_m \subseteq EH_m(FIN, FIN).$

Proof. Consider a regular valence grammar $G = (N, T, S, P)$ (so, the rules in the pairs of P are of the forms $r : X \to aY, r : X \to a, r : X \to \lambda$, for $X, Y \in N, a \in T$; we assume the rules labeled in a one-to-one manner), with valences over a group $(M, \diamond, e)$. We construct the valence H system

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

where

$$\begin{aligned} V &= N \cup T \cup \{Z_r \mid r : X \to x \text{ is a rule in } P\}, \\ A &= \{(S, e)\} \\ &\quad \cup \{(Z_r x, i_r) \mid (r : X \to x, i_r) \in P\}, \\ R &= \{\#X\$Z_r\#aY \mid r : X \to aY \text{ is a rule in } P\} \\ &\quad \cup \{\#X\$Z_r\#a \mid r : X \to a \text{ is a rule in } P\} \\ &\quad \cup \{\#X\$Z_r\# \mid r : X \to \lambda \text{ is a rule in } P\}. \end{aligned}$$

It is easy to see that any derivation in G can be simulated by a sequence of splicings in γ, using the axioms associated to the rules in G, and, conversely, each splicing step in γ corresponds to a derivation step in G. A terminal derivation in G starts from S and ends by using a terminal rule; a sequence of splicings in γ leads to a terminal string in the same conditions: we have to start by using S at the first step and we stop by using a splicing corresponding to a terminal rule at the last step. Moreover, because the symbols Z_r precisely identify the rules of G (hence also their valences), the valences of rules of G precisely correspond to the valences of the associated axioms of γ. This ensures that the total valence of a derivation in G (in the sense of the operation $\diamond$) is equal to the total valence of the corresponding sequence of splicings in γ. In conclusion, $L(G) = L(\gamma)$. □

This lemma transfers to valence H systems the power of valence regular grammars. In particular, we obtain:

Theorem 5. $EH_m(FIN, FIN) - CF \neq \emptyset$.

5 Final Remarks

We emphasize the fact that both types of "arithmetical" controls considered above, the age restriction and the valence restriction, increase the generative power of extended H systems with finite sets of axioms and of splicing rules, but, most probably, we do not obtain in this way characterizations of recursively enumerable languages. An observation

supporting this conjecture concerns the fact that our restrictions do not improve the context-sensitivity of H systems, but they induce additional properties to the generated strings, somewhat supplementary to the properties inherently induced by the splicing rules themselves.

Note. The work of the second author has been supported by CNR, Gruppo Nazionale per l'Informatica Matematica, Italy.

References

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

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

[3] D. Hauschild, M. Jantzen, Petri nets algorithms in the theory of matrix grammars, *Acta Informatica*, 31 (1994), pp.719–728.

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

[5] T. Head, Gh. Păun, D. Pixton, Language theory and molecular genetics. Generative mechanisms suggested by DNA recombination, chapter 7 in vol.2 of [11], pp.295–360.

[6] Gh. Păun, A new type of generative devices: valence grammars, *Rev. Roum. Math. Pures Appl.*, 25, 6 (1980), pp.911–924.

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

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

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

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

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

Vincenzo Manca, Gheorghe Păun,

Received February 26, 1998

Vincenzo Manca
Università degli Studi di Pisa,
Dipartimento di Informatica
Corso Italia 40, 56125 Pisa,
Italy

Gheorghe Păun
Institute of Mathematics
of the Romanian Academy
PO Box 1-764, 70700 Bucureşti,
Romania