On the Power of Networks of Evolutionary Processors

Jürgen Dassow
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
PSF 4120, D-39016 Magdeburg, Germany

and

Bianca Truthe*
Universitat Rovira i Virgili, Facultat de Lletres, GRLMC
Plaça Imperial Tàrraco 1, E-43005 Tarragona, Spain

February 6, 2007

Abstract

We discuss the power of networks of evolutionary processors where only two types of nodes are allowed. We prove that (up to an intersection with a monoid) every recursively enumerable language can be generated by a network with one deletion and two insertion nodes. Networks with an arbitrary number of deletion and substitution nodes only produce finite languages, and for each finite language one deletion node or one substitution node is sufficient. Networks with an arbitrary number of insertion and substitution nodes only generate context-sensitive languages, and (up to an intersection with a monoid) every context-sensitive language can be generated by a network with one substitution node and one insertion node.

1 Introduction

Motivated by some models of massively parallel computer architectures (see [10, 9]) networks of language processors have been introduced in [6] by E. CSUHAJ-VARJÚ and A. SALOMAA. Such a network can be considered as a graph where the nodes are sets of productions and at any moment of time a language is associated with a node. In a derivation step any node derives from its language all possible words as its new language. In a communication step any node sends those words to other nodes where the outgoing words have to satisfy an output condition given as a regular language, and any node takes words sent by the other nodes if the words satisfy

*The research was supported by the Alexander von Humboldt Foundation of the Federal Republic of Germany.

an input condition also given by a regular language. The language generated by a network of language processors consists of all (terminal) words which occur in the languages associated with a given node.

Inspired by biological processes, J. CASTELLANOS, C. MARTIN-VIDE, V. MITRANA and J. SEMPERE introduced in [3] a special type of networks of language processors which are called networks with evolutionary processors because the allowed productions model the point mutation known from biology. The sets of productions have to be substitutions of one letter by another letter or insertions of letters or deletion of letters; the nodes are then called substitution node or insertion node or deletion node, respectively. Results on networks of evolutionary processors can be found e. g. in [3, 4, 2, 1]. In [4] it was shown that networks of evolutionary processors are universal in that sense that they can generate any recursively enumerable language, and that networks with six nodes are sufficient to get all recursively enumerable languages. In [1] the latter result has been improved by showing that networks with three nodes are sufficient. The proof uses one node of each type (and intersection with a monoid).

Therefore it is a natural question to study the power of networks with evolutionary processors where the nodes have only two types, i. e.,

(i) networks with deletion nodes and substitution nodes (but without insertion nodes),
(ii) networks with insertion nodes and substitution nodes (but without deletion nodes), and
(iii) networks with deletion nodes and insertion nodes (but without substitution nodes).

In this paper we investigate the power of such systems and study the number of nodes sufficient to generate all languages which can be obtained by networks of the type under consideration. We prove that networks of type (i) and (iii) produce only finite and context-sensitive languages, respectively. Every finite, context-sensitive or recursively enumerable language can be generated by a network of type (i) with one node, by a network of type (ii) with two nodes or by a network of type (iii) with three nodes, respectively.

2 Definitions

We assume that the reader is familiar with the basic concepts of formal language theory (see e. g. [12]). We here only recall some notations used in the paper.

By V^* we denote the set of all words (strings) over V (including the empty word λ). The length of a word w is denoted by $|w|$.

In the proofs we shall often add new letters of an alphabet U to a given alphabet V. In all these situations we assume that $V \cap U = \emptyset$.

A phrase structure grammar is specified as a quadruple $G = (N,T,P,S)$ where N is a set of nonterminals, T is a set of terminals, P is a finite set of productions which are written as $\alpha \to \beta$ with $\alpha \in (N \cup T)^* \setminus T^*$ and $\beta \in (N \cup T)^*$, and $S \in N$ is the axiom. The grammar G is called monotone, if $|\alpha| \leq |\beta|$ holds for every rule $\alpha \to \beta$ of P.

A phrase structure grammar is in Kuroda normal form if all its productions have one of the following forms:

$$AB \to CD,\ A \to CD,\ A \to x,\ A \to \lambda \text{ where } A,B,C,D \in N,\ x \in N \cup T.$$

A conditional (monotone) grammar is a quadruple $G = (N,T,P',S)$ where N, T, and S are specified as in a phrase structure grammar and P' is a finite set of pairs $p = (\alpha_p \to \beta_p, R_p)$

where $\alpha_p \to \beta_p$ is a monotone production and R_p is a regular set. The direct derivation $w \Longrightarrow_p v$ in a conditional grammar is defined by the following conditions:

$$w = w_1\alpha_p w_2,\ v = w_1\beta_p w_2, \text{ and } w \in R_p,$$

i. e., a rule can only be applied to sentential forms which belong to the regular set associated with the rule. The language generated by a conditional grammar G is defined as the set of all words $z \in T^*$ for which productions $p_1, p_2, \ldots, p_r$ exist such that

$$S \Longrightarrow_{p_1} u_1 \Longrightarrow_{p_2} u_2 \Longrightarrow_{p_3} \ldots \Longrightarrow_{p_r} u_r = z.$$

We call a production $\alpha \to \beta$ a
– substitution if $|\alpha| = |\beta| = 1$,
– deletion if $|\alpha| = 1$ and $\beta = \lambda$.

We introduce insertions as a counterpart of a deletion. We write $\lambda \to a$, where a is a letter. The application of an insertion $\lambda \to a$ derives from a word w any word $w_1 a w_2$ with $w = w_1 w_2$ for some (possibly empty) words w_1 and w_2.

We now introduce the basic concept of this paper, the networks of evolutionary processors.

Definition 2.1

(i) *A network of evolutionary processors (of size n) is a tuple $\mathcal{N} = (V, N_1, N_2, \ldots, N_n, E, j)$ where*

- *V is a finite alphabet,*
- *for $1 \le i \le n$, $N_i = (M_i, A_i, I_i, O_i)$ where*
  - *M_i is a set of evolution rules of a certain type, i. e., $M_i \subseteq \{a \to b \mid a, b \in V\}$ or $M_i \subseteq \{a \to \lambda \mid a \in V\}$ or $M_i \subseteq \{\lambda \to b \mid b \in V\}$,*
  - *A_i is a finite subset of V^*,*
  - *I_i and O_i are regular sets over V,*
- *E is a subset of $\{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$, and*
- *j is a natural number such that $1 \le j \le n$.*

(ii) *A configuration C of $\mathcal{N}$ is an n-tuple $C = (C(1), C(2), \ldots, C(n))$ if $C(i)$ is a subset of V^* for $1 \le i \le n$.*

(iii) *Let $C = (C(1), C(2), \ldots, C(n))$ and $C' = (C'(1), C'(2), \ldots, C'(n))$ be two configurations of $\mathcal{N}$. We say that C derives C' in one*

– *evolution step (written as $C \Longrightarrow C'$) if, for $1 \le i \le n$, $C'(i)$ consists of all words $w \in C(i)$ to which no rule of M_i is applicable and of all words w for which there are a word $v \in C(i)$ and a rule $p \in M_i$ such that $v \Longrightarrow_p w$ holds,*

– *communication step (written as $C \vdash C'$) if, for $1 \le i \le n$,*

$$C'(i) = (C(i) \setminus O_i) \cup \bigcup_{(k,i) \in E} C(k) \cap O(k) \cap I(i).$$

The computation of $\mathcal{N}$ is a sequence of configurations $C_t = (C_t(1), C_t(2), \ldots, C_t(n))$, $t \ge 0$, such that

– $C_0 = (A_1, A_2, \dots, A_n)$,
– *for any* $t \geq 0$, C_{2t} *derives* C_{2t+1} *in one evolution step:* $C_{2t} \Longrightarrow C_{2t+1}$,
– *for any* $t \geq 0$, C_{2t+1} *derives* C_{2t+2} *in one communication step:* $C_{2t+1} \vdash C_{2t+2}$.

(iv) *The language* $L(\mathcal{N})$ *generated by* $\mathcal{N}$ *is defined as*

$$L(\mathcal{N}) = \bigcup_{t \geq 0} C_t(j)$$

where $C_t = (C_t(1), C_t(2), \dots, C_t(n))$, $t \geq 0$ *is the computation of* $\mathcal{N}$.

Intuitively a network with evolutionary processors is a graph consisting of some, say n, nodes $N_1, N_2, \dots, N_n$ (called processors) and the set of edges given by E such that there is a directed edge from N_k to N_i if and only if $(k,i) \in E$. Any processor N_i consists of a set of evolution rules M_i, a set of words A_i, an input filter I_i and an output filter O_i. We say that N_i is a substitution node or a deletion node or an insertion node if $M_i \subseteq \{a \to b \mid a, b \in V\}$ or $M_i \subseteq \{a \to \lambda \mid a \in V\}$ or $M_i \subseteq \{\lambda \to b \mid b \in V\}$, respectively. The input filter I_i and the output filter O_i control the words which are allowed to enter and to leave the node, respectively. With any node N_i and any time moment $t \geq 0$ we associate a set $C_t(i)$ of words (the words contained in the node at time t). Initially, N_i contains the words of A_i. In a derivation step we derive from $C_t(i)$ all words applying rules from the set M_i. In a communication step any processor N_i sends out all words $C_t(i) \cap O_i$ (which pass the output filter) to all processors to which a directed edge exists (only the words from $C_t(i) \setminus O_i$ remain in the set associated with N_i) and, moreover, it receives from any processor N_k such that there is an edge from N_k to N_i all words sent by N_k and passing the input filter I_i of N_i, i. e., the processor N_i gets in addition all words of $(C_t(k) \cap O_k) \cap I_i$. We start with a derivation step and then communication steps and derivation steps are alternately performed. The language consists of all words which are in the node N_j (j is chosen in advance) at some moment t, $t \geq 0$.

3 Networks with only Deletion and Substitution Nodes

In this section we study the power of networks which have only deletion and substitution nodes but no insertion nodes.

Lemma 3.1 *For any network* $\mathcal{N}$ *of evolutionary processors, which has only deletion and substitution nodes,* $L(\mathcal{N})$ *is a finite language.*

Proof. Let $\mathcal{N} = (V, N_1, N_2, \dots, N_n, E, j)$ be a network, which has only deletion and substitution nodes. Obviously, any evolution step and any communication step do not increase the length of a word contained in some $C_t(i)$, $1 \leq i \leq n$, $t \geq 0$. Therefore $L(\mathcal{N})$ contains only words of length at most

$$\max\{|w| \mid w \in A_i,\ 1 \leq i \leq n\}.$$

Hence $L(\mathcal{N})$ is a finite language. □

On the other hand, every finite language can be generated by a network of evolutionary processors without insertion nodes.

Lemma 3.2

(i) *For any finite language L, there is a network $\mathcal{N}$ of evolutionary processors which has exactly one substitution node such that $L(\mathcal{N}) = L$.*

(ii) *For any finite language L, there is a network $\mathcal{N}$ of evolutionary processors which has exactly one deletion node such that $L(\mathcal{N}) = L$.*

Proof. Obviously, the network $\mathcal{N} = (alph(L) \cup \{a,b\}, (\{a \to b\}, L, \emptyset, \emptyset), \emptyset, 1)$ generates L and its only node is a substitution node. Therefore part (i) is shown.
In order to prove part (ii), we change the system by using $a \to \lambda$ instead of $a \to b$. □

Combining the two preceding lemmas we get immediately the following statement.

Corollary 3.3 *The family of languages which can be generated by networks of evolutionary processors which have only deletion and substitution nodes coincides with $\mathcal{L}(FIN)$.* □

4 Networks with only Insertion and Substitution Nodes

In this section we study the power of networks which have only insertion and substitution nodes but no deletion nodes.

Lemma 4.1 *For any network $\mathcal{N}$ of evolutionary processors which has only insertion and substitution nodes, $L(\mathcal{N})$ is a context-sensitive language.*

Proof. Let $\mathcal{N} = (V, N_1, N_2, \ldots, N_n, E, h)$ be a network which has only insertion and substitution nodes. For $1 \le i \le n$, we set

$$V^{(i)} = \{x^{(i)} \mid x \in V\}.$$

If $w = x_1 x_2 \ldots x_m$, then $w^{(i)} = x_1^{(i)} x_2^{(i)} \ldots x_m^{(i)}$. In the grammar given below we shall use $w^{(i)}$ if the word w is in the node N_i. For $1 \le i \le n$, let $N_i = (M_i, A_i, I_i, O_i)$. Without loss of generality we assume that $N_1, N_2, \ldots, N_r$ are substitution nodes and that $N_{r+1}, N_{r+2}, \ldots, N_n$ are the insertion nodes. Moreover, for $1 \le i \le n$, let $\mathcal{A}_i = (V, Z_i, z_{0i}, \delta_i, F_i)$ and $\mathcal{B}_i = (V, Z'_i, z'_{0i}, \delta'_i, F'_i)$ be finite deterministic automata with input set V, state sets Z_i and Z'_i, initial states z_{0i} and z'_{0i}, transition functions δ_i and δ'_i and sets F_i and F'_i of accepting states, respectively, which accept the input filter I_i and the output filter O_i, respectively.

We construct the conditional grammar $G = (N, V \cup \{y\}, P, S)$, where

$$\begin{aligned} N \; = \; & V^{(1)} \cup V^{(2)} \cup \cdots \cup V^{(n)} \cup \{S, Y, A, A', B, B', X\} \\ & \cup \{X_i \mid 1 \le i \le n\} \cup \{X'_i \mid 1 \le i \le n\} \cup \{X_{ik} \mid 1 \le i \le n,\ 1 \le k \le n,\ (i,k) \in E\} \\ & \cup \{(z', z) \mid z' \in Z'_i,\ z \in Z_k,\ 1 \le i \le n,\ 1 \le k \le n,\ (i,k) \in E\} \\ & \cup \{z \mid z \in Z'_i,\ 1 \le i \le n\} \end{aligned}$$

and P is the set of all rules of the following forms

$$(S \to X_i A z^{(i)} X,\ \{S\}) \text{ for } z \in A_i,\ 1 \le i \le n$$

(from the axiom we derive X_iA followed by an indexed version of a word z of the initial set A_i and X; the letter X_i refers to a phase simulating a derivation step according to M_i),

$$(Ax^{(i)} \to x^{(i)}A,\ \{X_i\}(V^{(i)})^*\{A\}(V^{(i)})^*\{X\}) \text{ for } x \in V,\ 1 \le i \le n,$$
$$(Aa^{(i)} \to Bb^{(i)},\ \{X_i\}(V^{(i)})^*\{A\}(V^{(i)})^*\{X\}) \text{ for } a,b \in V,\ a \to b \in M_i,\ 1 \le i \le r,$$
$$(A \to Bb^{(i)},\ \{X_i\}(V^{(i)})^*\{A\}(V^{(i)})^*\{X\}) \text{ for } b \in V,\ \lambda \to b \in M_i,\ r+1 \le i \le n,$$
$$(AX \to YY,\ \{X_i\}(V^{(i)})^*\{AX\}) \text{ for } 1 \le i \le n,$$
$$(x^{(i)}B \to Bx^{(i)},\ \{X_i\}(V^{(i)})^*B(V^{(i)})^*\{X\}) \text{ for } x \in V,\ 1 \le i \le n$$

(we move the letter A to the right until we apply a rule of M_i which introduces B which is moved to the left; if no rule is applied we introduce the trap symbol Y which cannot be rewritten),

$$(X_iB \to X_{ik}(z'_{0i}, z_{0k}),\ \{X_iB\}(V^{(i)})^*\{X\}) \text{ for } 1 \le i \le n,\ 1 \le k \le n,\ (i,k) \in E,$$
$$(X_iB \to X'_iz'_{0i},\ \{X_iB\}(V^{(i)})^*\{X\}) \text{ for } 1 \le i \le n$$

(we change to a phase simulating a communication step where a word from node N_i is sent to node N_k announced by X_{ik} or to a phase simulating that that word does not leave the node N_i during the communication announced by X'_i),

$$((z',z)x^{(i)} \to x^{(k)}(\delta'_i(z',x),\delta_k(z,x)),\ \{X_{ik}\}(V^{(k)})^*\{(z',z) \mid z' \in Z'_i, z \in Z_k\}(V^{(i)})^*\{X\})$$
$$\text{for } z' \in Z'_i,\ z \in Z_k,\ 1 \le i \le n, 1 \le k \le n, (i,k) \in E,$$
$$((z',z)X \to B'X,\ \{X_{ik}\}(V^{(k)})^*\{(z',z) \mid z' \in Z'_i, z \in Z_k\}\{X\})$$
$$\text{for } z' \in F'_i,\ z \in F_k,\ 1 \le i \le n, 1 \le k \le n, (i,k) \in E,$$
$$((z',z)X \to YY,\ \{X_{ik}\}(V^{(k)})^*\{(z',z) \mid z' \in Z'_i, z \in Z_k\}\{X\})$$
$$\text{for } (z',z) \notin F'_i \times F_k,\ 1 \le i \le n, 1 \le k \le n, (i,k) \in E,$$
$$(x^{(k)}B' \to B'x^{(k)},\ \{X_{ik}\}(V^{(k)})^*\{B'\}(V^{(k)})^*\{X\})$$
$$\text{for } x \in V,\ 1 \le i \le n, 1 \le k \le n, (i,k) \in E,$$
$$(X_{ik}B' \to X_kA,\ \{X_{ik}B'\}(V^{(k)})^*\{X\}) \text{ for } 1 \le i \le n, 1 \le k \le n, (i,k) \in E$$

(we move (z',z) from left to right, i. e., we read the word in the node, and simulate the work of $\mathcal{B}_i$ and $\mathcal{A}_k$ in the first and second component, respectively; if accepting states of both automata are reached, the word can pass the output filter O_i and the input filter I_k, and therefore it goes to the node N_k; this corresponds to the change of any letter $x^{(i)}$ to $x^{(k)}$; the letter B' is sent back to the left where a change to a derivation step in N_k is done; if one of the states after reading the word is not accepting, we generate the trap symbol Y),

$$(z'x^{(i)} \to x^{(i)}\delta'_i(z',x),\ \{X'_i\}(V^{(i)})^*\{z' \mid z' \in Z'_i\}(V^{(i)})^*\{X\}) \text{ for } z' \in Z'_i,\ 1 \le i \le n,$$
$$(z'X \to B'X,\ \{X'_i\}(V^{(i)})^*\{z' \mid z' \in Z'_i\}\{X\}) \text{ for } z' \notin F'_i,\ 1 \le i \le n,$$
$$(z'X \to YY,\ \{X'_i\}(V^{(i)})^*\{z' \mid z' \in Z'_i\}\{X\}) \text{ for } z' \in F'_i,\ 1 \le i \le n,$$
$$(x^{(i)}B' \to B'x^{(i)},\ \{X'_i\}(V^{(i)})^*\{B'\}(V^{(i)})^*\{X\}) \text{ for } x \in V,\ 1 \le i \le n,$$
$$(X'_iB' \to X_iA,\ \{X'_iB'\}(V^{(i)})^*\{X\}) \text{ for } 1 \le i \le n$$

(we read the word, again, without a change and go to a derivation step according to M_i if the word cannot pass the output filter O_i, i. e., if the word is not accepted by $\mathcal{B}_i$),

$(X_h B \to yA',\ \{X_h B\}(V^{(h)})^*\{X\})$,
$(X_{ih} B' \to yA',\ \{X_{ih} B'\}(V^{(h)})^*\{X\})$ for $1 \le i \le n$,
$(A'x^{(h)} \to x^{(h)}A',\ \{y\}(V^{(h)})^*\{A'\}(V^{(h)})^*\{X\})$ for $x \in V$,
$(A'X \to yy,\ \{y\}(V^{(h)})^*\{A'X\})$

(if one derivation phase or communication phase is finished we transform the word in node N_h, which collects the elements of the language, to the terminal alphabet).

By the explanations given to the rules it is easy to see that $L(G) = \{y\}L(\mathcal{N})\{yy\}$. Since conditional monotone grammars only generate context-sensitive languages (see [8], page 122), $L(G)$ is context-sensitive. By the closure of the family of context-sensitive languages under derivatives, $L(\mathcal{N})$ is context-sensitive, too. □

Lemma 4.2 *For any context-sensitive language L, there are a set T and a network $\mathcal{N}$ of evolutionary processors with exactly one insertion node and exactly one substitution node such that $L = L(\mathcal{N}) \cap T^*$.*

Proof. Let L be a context-sensitive language and $G = (N,T,P,S)$ be a grammar in Kuroda normal form with $L(G) = L$. Let $R_1, R_2, \ldots, R_7$ be the following sets:

$R_1 = \{\, A \to p_0,\ p_0 \to x \mid p = A \to x \in P,\ A \in N,\ x \in T \,\}$,
$R_2 = \{\, A \to p_1 \mid p = A \to CD \in P$ or $p = AB \to CD \in P,\ A,B,C,D \in N \,\}$,
$R_3 = \{\, B \to p_2 \mid p = AB \to CD \in P,\ A,B,C,D \in N \,\}$,
$R_4 = \{\, p_1 \to p_3 \mid p \in P \,\}$,
$R_5 = \{\, p_2 \to p_4 \mid p \in P \,\}$,
$R_6 = \{\, p_3 \to C \mid p = A \to CD \in P$ or $p = AB \to CD \in P,\ A,B,C,D \in N \,\}$,
$R_7 = \{\, p_4 \to D \mid p = A \to CD \in P$ or $p = AB \to CD \in P,\ A,B,C,D \in N \,\}$.

We construct a network of evolutionary processors

$$\mathcal{N} = (V,(M_1,\{\,S\,\},I_1,O_1),(M_2,\emptyset,I_2,V^*),\{\,(1,2),(2,1)\,\},1)$$

with

$$\begin{aligned} V &= N \cup T \cup \{\, p_0,p_1,p_2,p_3,p_4 \mid p \in P \,\}, \\ M_1 &= R_1 \cup R_2 \cup R_3 \cup R_4 \cup R_5 \cup R_6 \cup R_7, \\ I_1 &= (N \cup T)^* \{\, p_1p_2 \mid p = A \to CD \in P \,\} (N \cup T)^*, \\ O_1 &= V^* \setminus ((N \cup T)^* \bar{O} (N \cup T)^*), \end{aligned}$$

where

$$\begin{aligned} \bar{O} = \{\,\lambda\,\} &\cup \{\, p_1 \mid p = AB \to CD \in P \,\} \\ &\cup \{\, p_1p_2 \mid p = AB \to CD \in P \,\} \\ &\cup \{\, p_3p_2 \mid p = A \to CD \in P \text{ or } p = AB \to CD \in P \,\} \\ &\cup \{\, p_3p_4 \mid p = A \to CD \in P \text{ or } p = AB \to CD \in P \,\} \\ &\cup \{\, Cp_4 \mid p = A \to CD \in P \text{ or } p = AB \to CD \in P \,\}, \end{aligned}$$

and

$$M_2 = \{\, \lambda \to p_2 \mid p = A \to CD \in P \,\},$$
$$I_2 = (N \cup T)^* \{\, p_1 \mid p = A \to CD \in P \,\} (N \cup T)^*.$$

First, we show that any application of a rule of the grammar G can be simulated by the network $\mathcal{N}$. In the sequel, A, B, C, D are non-terminals, x is a terminal and $w_1 w_2 \in (N \cup T)^*$.

Case 1. Application of the rule $p = A \to x \in P$ to a word $w_1 A w_2$.
This is achieved by the rules $A \to p_0 \in R_1$ and then $p_0 \to x \in R_1$. After each of these two evolution steps, the word does not leave the node.

Case 2. Application of the rule $p = AB \to CD \in P$ to a word $w_1 AB w_2$.
The word $w_1 AB w_2$ is changed to $w_1 p_1 B w_2$ (by an appropriate rule of R_2) which cannot pass the output filter, so it remains in the first node. It is then changed to $w_1 p_1 p_2 w_2$ (by R_3), and further, without leaving the first node, changed to $w_1 p_3 p_2 w_2$ (by R_4), to $w_1 p_3 p_4 w_2$ (by R_5), to $w_1 C p_4 w_2$ (by R_6) and finally to $w_1 CD w_2$ (by R_7). This word is not communicated in the next step, since it cannot pass the output filter. Hence the application of the rule $p = AB \to CD \in P$ to a word $w_1 AB w_2$ can be simulated in six evolution steps (the six corresponding communication steps have no effect).

Case 3. Application of the rule $p = A \to CD \in P$ to a word $w_1 A w_2$.
The word $w_1 A w_2$ is changed to $w_1 p_1 w_2$ (by an appropriate rule of R_2). This word passes the output filter of the first node and the input filter of the second one. There, the symbol p_2 is inserted behind p_1 and the obtained word $w_1 p_1 p_2 w_2$ is communicated back to the first node. There, the word is changed to $w_1 p_3 p_2 w_2$ (by R_4) and further as in the Case 2 to the words $w_1 p_3 p_4 w_2$ (by R_5), $w_1 C p_4 w_2$ (by R_6) and $w_1 CD w_2$ (by R_7). This word is not communicated in the next step, since it cannot pass the output filter. Hence the application of the rule $p = A \to CD \in P$ to a word $w_1 A w_2$ can be simulated in six evolution steps and two effective communication steps (the other four have no effect).

Since the start symbol S also belongs to the language of the network, any derivation step in the grammar G can be simulated by evolution and communication steps in the network $\mathcal{N}$. Hence, we have the inclusion $L(G) \subseteq L(\mathcal{N}) \cap T^*$. We show now $L(\mathcal{N}) \cap T^* \subseteq L(G)$.

Let $F(G)$ be the set of all sentential forms generated by the grammar G. We show that $L(\mathcal{N}) \cap (N \cup T)^* \subseteq F(G)$. Then $L(\mathcal{N}) \cap T^* \subseteq L(G)$ follows immediately.

The start symbol S belongs to both sets $L(\mathcal{N}) \cap (N \cup T)^*$ and $F(G)$. We now consider a word $w = w_1 A w_2$ of the set $L(\mathcal{N}) \cap (N \cup T)^*$ with $A \in N$. The word is in the first node and it is not communicated, so we start with an evolution step.

Case 1. Application of a rule $A \to p_0 \in R_1$.
This yields the word $w_1 p_0 w_2$ in the first node. Due to the output filter, it remains there. Thereafter, the rule $p_0 \to x \in R_1$ has to be applied or we loose the word. Hence, these two evolution steps represent the derivation $w_1 A w_2 \Longrightarrow w_1 x w_2$ in G.

Case 2. Application of a rule $A \to p_2 \in R_3$.
This leads to the word $w_1 p_2 w_2$ in the first node, which is then sent out. Since the second node does not accept it, the word is lost.

Case 3. Application of a rule $A \to p_1 \in R_2$.
There are two possibilities for the rule p that belongs to p_1.

Case 3.1. $p = A \to CD$. In this case, the word $w_1p_1w_2$ is sent out and caught by the second node. The second node inserts a q_2. If q_2 is not p_2 or if it is p_2 but not inserted immediately behind p_1, then the obtained word is not $w_1p_1p_2w_2$. It is sent back but not accepted by the first node and therefore lost. If p_2 is inserted at the correct position, then the word $w_1p_1p_2w_2$ enters the first node. We set $w_2' = w_2$ and continue with the word $w_1p_1p_2w_2'$ to the next evolution step.

Case 3.2. $p = AB \to CD$. In this case, the word $w_1p_1w_2$ remains in the first node. If the word after the next evolution step is not $w_1p_1p_2w_2'$ with $w_2 = Bw_2'$, then it is sent out, because applying any other rule of $R_1 \cup R_2 \cup R_3 \cup R_4$ (rules of R_5, R_6 and R_7 are not applicable) yields a word which passes the output filter. Since it cannot pass the input filter of the other node, the word gets lost. So, the word is only kept alive, if it is $w_1p_1p_2w_2'$ with $w_2 = Bw_2'$. This word cannot leave the first node, so we continue with $w_1p_1p_2w_2'$ in the first node to the next evolution step.

In both subcases, the only word that can be obtained in the first node after two evolution steps and two communication steps starting from the word $w = w_1Aw_2$ is $w_1p_1p_2w_2'$. We continue with an evolution step. Applying a rule of R_1, R_2, R_3 or R_5 leads to a word which leaves the first node and disappears. Rules of R_6 and R_7 are not applicable. By the only successful rule $p_1 \to p_3 \in R_4$, we obtain the word $w_1p_3p_2w_2'$ which is kept in the first node. The next evolution step uses the rule $p_2 \to p_4 \in R_5$ because the rules of R_1, R_2, R_3 and R_6 lead to loosing the word and R_4 and R_7 are not applicable. This yields the word $w_1p_3p_4w_2'$ which is also kept in the first node. In the next evolution step, the rules of R_1, R_2, R_3 and R_7 make the word disappear and R_4 and R_5 are not possible. Hence, the only next word is $w_1Cp_4w_2'$ after applying the rule $p_3 \to C \in R_6$. It remains in the first node. Now the rules of R_4, R_5 and R_6 are not applicable; by rules of R_1, R_2 and R_3 the word will be lost. The only possible rule $p_4 \to D \in R_7$ yields the word w_1CDw_2' which is not sent out in the next communication step.

Hence, in this case, the derivation $w_1Aw_2 \Longrightarrow w_1CDw_2'$ (which in G is obtained by the initially chosen rule p) is simulated.

Other rules are not applicable to the word w.

By the case distinction above, we have shown that every word $z \in L(\mathcal{N}) \cap (N \cup T)^*$ that is derived by the network $\mathcal{N}$ from a word $w \in L(\mathcal{N}) \cap (N \cup T)^*$ is also derived by the grammar G from the word w and, hence, belongs to the set $F(G)$.

From the inclusion $L(\mathcal{N}) \cap (N \cup T)^* \subseteq F(G)$, the required inclusion $L(\mathcal{N}) \cap T^* \subseteq L(G)$ follows. Together with the first part of the proof, we have $L(G) = L(\mathcal{N}) \cap T^* = L$. □

Corollary 4.3 *For any context-sensitive language L, there is a network $\mathcal{N}$ of evolutionary processors with three nodes which are insertion nodes and substitution nodes such that $L = L(\mathcal{N})$.*

Proof. Let L be a context-sensitive language. Then we construct as in the proof of Lemma 4.2 a network $\mathcal{N} = (V, N_1, N_2, E, 1)$ with one insertion node and one substitution node such that $L = L(\mathcal{N}) \cap T^*$ and from $\mathcal{N}$ the network

$$\mathcal{N}' = (V, N_1, N_2, N_3, E \cup \{(1,3)\}, 3) \text{ with } N_3 = (\emptyset, \emptyset, T^*, \emptyset).$$

It is obvious from the proof of Theorem 4.2 that N_3 collects exactly the words from $L(\mathcal{N}) \cap T^*$. Thus $L(\mathcal{N}') = L$. □

By Lemma 4.1 and Corollary 4.3 we get immediately the following statement.

Corollary 4.4 *The family of languages which can be generated by networks of evolutionary processors which have only insertion and substitution nodes coincides with* $\mathcal{L}(CS)$. □

5 Networks with only Deletion and Insertion Nodes

In this section we discuss networks which have only insertion and deletion nodes. In the paper [11], the authors have also studied systems where only insertion and deletion are allowed. However, in contrast to our definition it is possible to delete and insert words of arbitrary length (the authors show that words of length at most three are sufficient); we can delete and insert only letters. On the other hand, we can use filters which is not possible in [11]. We shall prove that networks with deletion and insertion nodes can generate any recursively enumerable language. This means that partition of the rules to nodes and the use of regular filters has the same power as deletion and insertion of words of arbitrary length.

Lemma 5.1 *For any recursively enumerable language* L*, there are a set* T *and a network* $\mathcal{N}$ *of evolutionary processors with exactly two insertion nodes and exactly one deletion node such that* $L = L(\mathcal{N}) \cap T^*$.

Proof. Let L be a recursively enumerable language and $G = (N,T,P,S)$ be a grammar in Kuroda normal form with $L(G) = L$. In the sequel, A, B, C, D, X, Y, Z designate nonterminals, x a terminal, p, r rules of P; $q \notin N \cup T$ is a new symbol, and p_1, p_2, p_3, p_4, p_5 are new symbols for every rule $p \in P$ (and only those rules). We define now sets that will be used for defining the filters (to make them more readable). Let

$$\begin{aligned}
\alpha_{p_1q} &= \{\, p_1q \mid \exists A : p = A \to \lambda \,\},\\
\alpha_{p_1x} &= \{\, p_1x \mid \exists A : p = A \to x \,\},\\
\alpha_{p_1CD} &= \{\, p_1CD \mid \exists A : p = A \to CD \text{ or } \exists A,B : p = AB \to CD \,\},\\
\beta_1 &= \alpha_{p_1q} \cup \alpha_{p_1x} \cup \alpha_{p_1CD},
\end{aligned}$$

$$\begin{aligned}
\alpha_{Ap_5} &= \{\, Ap_5 \mid p = A \to \lambda \,\},\\
\alpha_{Ap_4} &= \{\, Ap_4 \mid \exists x : p = A \to x \text{ or } \exists C,D : p = A \to CD \,\},\\
\alpha_{ABp_4} &= \{\, Aq^nBp_4 \mid n \geq 0 \text{ and } \exists C,D : p = AB \to CD \,\},\\
\beta_2 &= \alpha_{Ap_5} \cup \alpha_{Ap_4} \cup \alpha_{ABp_4},
\end{aligned}$$

$$\begin{aligned}
\alpha_{Aqp_5} &= \{\, Aqp_5 \mid p = A \to \lambda \,\},\\
\alpha_{Ap_2p_4} &= \{\, Ap_2p_4 \mid \exists x : p = A \to x \text{ or } \exists C,D : p = A \to CD \,\},\\
\alpha_{ABp_2p_4} &= \{\, Aq^nBp_2p_4 \mid n \geq 0 \text{ and } \exists C,D : p = AB \to CD \,\},\\
\beta_3 &= \alpha_{Aqp_5} \cup \alpha_{Ap_2p_4} \cup \alpha_{ABp_2p_4},
\end{aligned}$$

$$
\begin{aligned}
\alpha_{Aq} &= \{\, Aq \mid A \to \lambda \in P \,\}\,, \\
\alpha_{Ap_2} &= \{\, Ap_2 \mid \exists x : p = A \to x \text{ or } \exists C, D : p = A \to CD \,\}\,, \\
\alpha_{ABp_2} &= \{\, Aq^n Bp_2 \mid n \geq 0 \text{ and } \exists C, D : p = AB \to CD \,\}\,, \\
\alpha_{Ap_2p_3} &= \{\, Ap_2p_3 \mid \exists C, D : p = A \to CD \,\}\,, \\
\beta_4 &= \alpha_{Aq} \cup \alpha_{Ap_2} \cup \alpha_{ABp_2}, \\
\beta_4' &= \beta_4 \cup \alpha_{Ap_2p_3}
\end{aligned}
$$

$$
\begin{aligned}
\alpha_{p_1Aq} &= \{\, p_1Aq \mid p = A \to \lambda \,\}\,, \\
\alpha_{p_1Ap_2} &= \{\, p_1Ap_2 \mid \exists x : p = A \to x \,\}\,, \\
\alpha_{p_1ABp_2} &= \{\, p_1Aq^n Bp_2 \mid n \geq 0 \text{ and } \exists C, D : p = AB \to CD \,\}\,, \\
\alpha_{p_1Ap_2p_3} &= \{\, p_1Ap_2p_3 \mid \exists C, D : p = A \to CD \,\}\,, \\
\beta_5 &= \alpha_{p_1Aq} \cup \alpha_{p_1Ap_2} \cup \alpha_{p_1ABp_2} \cup \alpha_{p_1Ap_2p_3},
\end{aligned}
$$

$$
\begin{aligned}
\alpha_{p_1xp_2} &= \{\, p_1xp_2 \mid \exists A : p = A \to x \,\}\,, \\
\alpha_{p_1CBp_2} &= \{\, p_1Cq^n Bp_2 \mid n \geq 0 \text{ and } \exists A, D : p = AB \to CD \,\}\,, \\
\alpha_{p_1Cp_2p_3} &= \{\, p_1Cp_2p_3 \mid \exists A, D : p = A \to CD \,\}\,, \\
\alpha_{p_1CDp_2} &= \{\, p_1CDq^n p_2 \mid n \geq 0 \text{ and } \exists A, B : p = AB \to CD \\
&\qquad\qquad \text{or } n = 0 \text{ and } \exists A : p = A \to CD \,\}, \\
\beta_6 &= \alpha_{p_1xp_2} \cup \alpha_{p_1CBp_2} \cup \alpha_{p_1Cp_2p_3} \cup \alpha_{p_1CDp_2},
\end{aligned}
$$

$$
\begin{aligned}
\alpha_{Ap_1qBr_4} &= \{\, Aq^n p_1 q q^m Br_4 \mid n, m \geq 0 \text{ and } \exists X : p = X \to \lambda \\
&\qquad\qquad \text{and } \exists C, D : r = AB \to CD \,\}, \\
\alpha_{p_1Cr_4D} &= \{\, p_1Cr_4D \mid (\exists A : p = A \to CD \text{ or } \exists A, B : p = AB \to CD) \\
&\qquad\qquad \text{and } (\exists x : r = C \to x \text{ or } \exists X, Y : r = C \to XY) \,\}, \\
\alpha_{Zp_1Cr_4D} &= \{\, Zq^n p_1Cr_4D \mid n \geq 0 \text{ and } (\exists A : p = A \to CD \text{ or } \exists A, B : p = AB \to CD) \\
&\qquad\qquad \text{and } \exists X, Y : r = ZC \to XY \,\}, \\
\alpha_{p_1CDr_4} &= \{\, p_1CDr_4 \mid (\exists p = A \to CD \text{ or } \exists A, B : p = AB \to CD) \text{ and} \\
&\qquad\qquad (\exists x : r = D \to x \text{ or } \exists X, Y : (r = D \to XY \text{ or } r = CD \to XY)) \,\}, \\
\alpha_{p_1CDXr_4} &= \{\, p_1CDq^n Xr_4 \mid n \geq 0 \text{ and } (\exists A : p = A \to CD \text{ or } \exists A, B : p = AB \to CD) \\
&\qquad\qquad \text{and } \exists X, Y : r = DX \to YZ \,\}, \\
\alpha_{p_1Cr_5D} &= \{\, p_1Cr_5D \mid (\exists A : p = A \to CD \text{ or } \exists A, B : p = AB \to CD) \\
&\qquad\qquad \text{and } r = C \to \lambda \,\}, \\
\alpha_{p_1CDr_5} &= \{\, p_1CDr_5 \mid (\exists A : p = A \to CD \text{ or } \exists A, B : p = AB \to CD) \\
&\qquad\qquad \text{and } r = D \to \lambda \,\}, \\
\beta_7 &= \alpha_{Ap_1qBr_4} \cup \alpha_{p_1Cr_4D} \cup \alpha_{Zp_1Cr_4D} \cup \alpha_{p_1CDr_4} \cup \alpha_{p_1CDXr_4} \cup \alpha_{p_1Cr_5D} \cup \alpha_{p_1CDr_5},
\end{aligned}
$$

$$
\begin{aligned}
\alpha_{p_1p_2} &= \{\, p_1p_2 \mid \exists A, x : p = A \to x \,\}, \\
\alpha_{p_1Bp_2} &= \{\, p_1q^nBp_2 \mid n \geq 0 \text{ and } \exists A, C, D : p = AB \to CD \,\}, \\
\alpha_{p_1Cp_2} &= \{\, p_1Cq^np_2 \mid n \geq 0 \text{ and } \exists A, B, D : p = AB \to CD \\
&\qquad\qquad \text{or } n = 0 \text{ and } \exists A, D : p = A \to CD \,\}, \\
\alpha_{p_1p_2p_3} &= \{\, p_1p_2p_3 \mid \exists A, C, D : p = A \to CD \,\}, \\
\beta_8 &= \alpha_{p_1p_2} \cup \alpha_{p_1Bp_2} \cup \alpha_{p_1Cp_2} \cup \alpha_{p_1p_2p_3},
\end{aligned}
$$

$$
\begin{aligned}
T_q &= (T \cup \{\, q \,\})^*, \\
W &= (N \cup T \cup \{\, q \,\})^*.
\end{aligned}
$$

Now, we construct a network of evolutionary processors

$$\mathcal{N} = (V, (M_1, \{\, S \,\}, I_1, O_1), (M_2, \emptyset, I_2, O_2), (M_3, \emptyset, I_3, O_3), \{\, (1,2), (2,1), (2,3), (3,2) \,\}, 1)$$

with

$$
\begin{aligned}
V = N \cup T \cup \{\, q \,\} &\cup \bigcup_{p = A \to \lambda \in P} \{\, p_1, p_5 \,\} \cup \bigcup_{p = A \to x \in P} \{\, p_1, p_2, p_4 \,\} \\
&\cup \bigcup_{p = A \to CD \in P} \{\, p_1, p_2, p_3, p_4 \,\} \cup \bigcup_{p = AB \to CD \in P} \{\, p_1, p_2, p_4 \,\},
\end{aligned}
$$

$$
\begin{aligned}
M_1 &= \{\, \lambda \to q \,\} \cup \{\, \lambda \to p_i \mid 1 \leq i \leq 5 \text{ and } p_i \in V \,\}, \\
M_2 &= \{\, A \to \lambda \mid A \in N \,\} \cup \{\, p_i \to \lambda \mid 1 \leq i \leq 5 \text{ and } p_i \in V \,\} \cup \{\, q \to \lambda \,\}, \\
M_3 &= \{\, \lambda \to A \mid A \in N \,\} \cup \{\, \lambda \to x \mid x \in T \,\}, \\
O_1 &= V^* \setminus (W\beta_4'W), \\
O_2 &= V^* \setminus (T_q \{\, p_1 \mid p \in P \text{ and } p_1 \in V \,\} T_q), \\
O_3 &= V^*, \\
I_1 &= W(\beta_1 \cup \beta_2 \cup \beta_4)W \cup T^*, \\
I_2 &= W(\beta_3 \cup \beta_5 \cup \beta_6 \cup \beta_7)W \cup W\beta_1W\beta_2W \cup W\beta_2W\beta_1W, \\
I_3 &= W\beta_8W.
\end{aligned}
$$

An element of V is called a non-terminal if it belongs to N, a terminal if it belongs to T and a marker otherwise.

First, we show that any application of a rule of the grammar G can be simulated by the network $\mathcal{N}$. At the beginning, the start symbol S is to be found in the first node. Regarding S, there are three possibilities for a rule $p \in P$.

Case 1. $p = S \to \lambda$.

In the first node, q is inserted behind S, the word does not leave the node, p_1 is inserted before S and then the word is communicated to the second node. There, S is removed. The word is now p_1q and does not leave the node. Still in the second node, first q and then p_1 are deleted. In the next communication step, the empty word λ is transferred to the first node. The derivation $S \Longrightarrow \lambda$ in G has been simulated in the network $\mathcal{N}$.

Case 2. $p = S \to x$.

In the first node, p_2 is inserted behind S, the word does not leave, p_1 is inserted before S and then the word is communicated to the second node. There, S is removed. The word is now p_1p_2 and is communicated to the third node. There, x is inserted between p_1 and p_2. The word p_1xp_2 goes back to the second node, where first p_2 and then p_1 are removed. The obtained word x is sent to the first node. The derivation $S \Longrightarrow x$ in G has been simulated in the network $\mathcal{N}$.

Case 3. $p = S \to CD$.

In the first node, first p_2, then p_3 and finally p_1 are inserted to obtain the word $p_1Sp_2p_3$. The order of the insertions is important; otherwise the word would not remain in the first node. The word $p_1Sp_2p_3$ is communicated to the second node, where S will be deleted. The word $p_1p_2p_3$ is then sent to the third node. There, C is inserted between p_1 and p_2. After transferring the word $p_1Cp_2p_3$ back to the second node, p_3 is deleted. Then, the word p_1Cp_2 is sent to the third node again, where D is inserted behind C. The word p_1CDp_2 is communicated to the second node, where p_2 will be deleted. After that, the word p_1CD moves to the first node. The derivation $S \Longrightarrow CD$ in the grammar G has been simulated in the network $\mathcal{N}$ such that in the end the word p_1CD with $p = S \to CD$ is to be found in the first node and the next derivation step is a rewriting step.

We describe now how the further derivations can be simulated. Let w be the word of the first node containing a non-terminal and a marker r_1 (from the previous simulation), but no other markers.

Case 4. $p = A \to \lambda$ and $w = w_1Aw_2$.

The first node inserts p_5 behind A and the word $w_1Ap_5w_2$ is transferred to the second node. There r_1 is deleted and the word is sent back to the first node. This node inserts q between A and p_5 and sends the word to the second node. This node deletes p_5 and sends the word back. The first node inserts p_1 before A. The word now contains p_1Aq as a subword and enters the second node. This node deletes A. The network has now simulated the application of p. If there is a non-terminal left, the word is send back to the first node. If this A was the last non-terminal, the second node deletes all qs and finally p_1. The terminal word t is sent to the first node. The network has simulated the derivation $S \Longrightarrow^* t$.

Case 5. $p = A \to x$ and $w = w_1Aw_2$.

The first node inserts p_4 behind A and the word $w_1Ap_4w_2$ is transferred to the second node. There r_1 is deleted and the word is sent back to the first node. This node inserts p_2 between A and p_4 and sends the word to the second node. This node deletes p_4 and sends the word back. The first node inserts p_1 before A. The word now contains p_1Ap_2 as a subword and enters the second node, where A is deleted. The word then goes to the third node, where x is inserted between p_1 and p_2. Then, the word moves to the second node, which deletes p_2. The network has now simulated the application of p. If there is a non-terminal left, the word is send back to the first node. Otherwise, the second node deletes all qs and finally p_1. The terminal word t is sent to the first node. The network has simulated the derivation $S \Longrightarrow^* t$.

Case 6. $p = A \to CD$ and $w = w_1Aw_2$.

As in the second case, the first node inserts p_4 behind A. The word $w_1Ap_4w_2$ is transferred to the second node. There, r_1 is deleted and the word is sent back to the first node. This node inserts p_2 between A and p_4 and sends the word to the second node. This node deletes p_4 and sends the word back. The first node inserts p_3 behind p_2 and, because the word does not leave the node, also p_1 before A. The word now contains $p_1Ap_2p_3$ as a subword and enters the second node, where A is deleted. The word then goes to the third node, where C is inserted between p_1 and p_2. Then, the word moves to the second node, which deletes p_3. The word is communicated to the third node, where D is inserted between C and p_2. Thereafter, the word moves to the second node, which deletes p_2 and sends the word to the first node. The network has now simulated the application of p.

Case 7. $p = AB \to CD$ and $w = w_1Aq^n w_3 q^m Bw_2$ with $m,n \geq 0$ and $w_3 \in \{\lambda\} \cup \{r_1 \mid r \in P\}$. As in the second case, the first node inserts p_4 behind B. The word is transferred to the second node. There, r_1 is deleted and the word is sent back to the first node. This node inserts p_2 between B and p_4 and sends the word to the second node. This node deletes p_4 and sends the word back. The first node inserts p_1 before A. The word now contains $p_1Aq^{n+m}p_2$ as a subword and enters the second node, where A is deleted. The word then goes to the third node, where C is inserted behind p_1. Then, the word moves to the second node, which deletes B. The word is communicated to the third node, where D is inserted behind C. Thereafter, the word moves to the second node, which deletes p_2 and sends the word to the first node. The network has now simulated the application of p.

The cases described above are repeated until the obtained word in the first node contains only one non-terminal A and the rule to be applied is of the Case 4 or 5. Then, the simulation of the derivation finishes with a terminal word in the first component.

Hence, any derivation $S \Longrightarrow^* w$ with $w \in T^*$ in the grammar G can be simulated by the network $\mathcal{N}$. Thus, we have the inclusion $L(G) \subseteq L(\mathcal{N}) \cap T^*$.

We now prove the inclusion $L(\mathcal{N}) \cap T^* \subseteq L(G)$.

We start with the word S (axiom) in the first node and trace each of its derivations in the network. The pure word of a word w is the word which is obtained by removing all markers from the word w. Let us consider the following situations for the general case, in which the 'observed' word can be found such that the next step is a rewriting step. The first component gives a 'description' (a set where the word belongs to) and the second component states the number of the node where the word resides:

$$
\begin{array}{ll}
\sigma_1 \;= (\{S\}, 1), & \sigma_2 \;= (W\alpha_{p_1CD}W, 1), \\
\sigma_3 \;= (WNW\alpha_{p_1x}W \cup W\alpha_{p_1x}WNW, 1), & \sigma_4 \;= (WNW\alpha_{p_1q}W \cup W\alpha_{p_1q}WNW, 1), \\
\sigma_5 \;= (W\alpha_{Ap_4}W, 1), & \sigma_6 \;= (W\alpha_{ABp_4}W, 1), \\
\sigma_7 \;= (W\alpha_{Ap_5}W, 1), & \sigma_8 \;= (W\alpha_{Ap_2}W, 1), \\
\sigma_9 \;= (W\alpha_{ABp_2}W, 1), & \sigma_{10} = (W\alpha_{Aq}W, 1), \\
\sigma_{11} = (T_q\{p_1 \mid p \in P\}T_q, 2), & \sigma_{12} = (T^*, 1).
\end{array}
$$

The following graph shows the connections between these situations (Figure 1). A directed edge from a situation σ_i to a situation σ_j means that a word which is in situation σ_i can be transformed by the (general) network into a word which is in situation σ_j, such that during the transformation no situation $\sigma_1, \ldots, \sigma_{12}$ is met.

We investigate each situation σ_i and show that

- a situation σ_j is directly reachable in finitely many rewriting and communication steps if and only if there is an edge from σ_i to σ_j in the graph of Figure 1, and
- the corresponding pure word is a sentential form of the grammar G which contains a non-terminal if $i \leq 10$ and is terminal if $i \geq 11$.

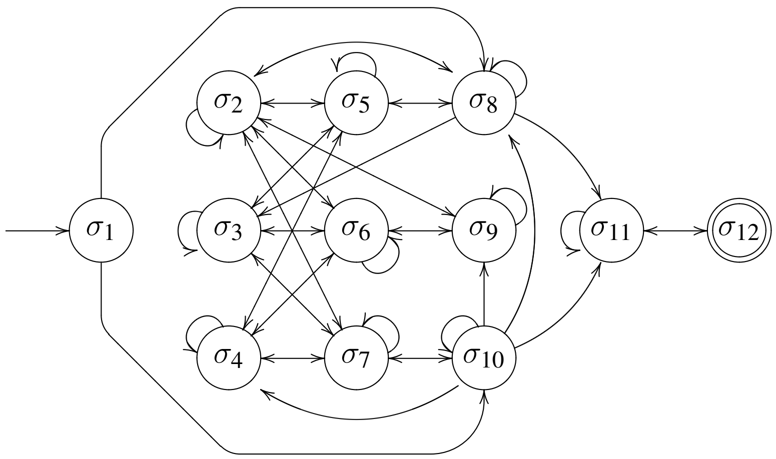

Figure 1: Situation graph

It turns out, that no terminal word occurs in the first node of the network outside the situation σ_{12}. After proving that every terminal word which is generated by the network $\mathcal{N}$ is also a word of the grammar G, we have immediately the desired inclusion $L(\mathcal{N}) \cap T^* \subseteq L(G)$.

Case 1. We start with the axiom S in the first node (situation σ_1). If one of the symbols p_1, p_3, p_4 or p_5 is inserted, then the word leaves the network. This also happens, if q is inserted behind S but without the rule $S \to \lambda$ being in the set P or if q is inserted before S. If a symbol p_2 is inserted behind S but the rule p does not have S on its left hand side or if p_2 is inserted before S, the word disappears as well. The two remaining cases are:

Case 1.1. The symbol p_2 for a rule p that has S on its left hand side is inserted behind S. Then the word does not leave the node, and we have now the situation σ_8.

Case 1.2. The symbol q is inserted behind S and the rule $S \to \lambda$ exists in the rule set P. Then the word does not leave the node, and we have now the situation σ_{10}.

In both subcases, the pure word is still S, hence a sentential form with a non-terminal of the grammar G.

Case 2. The word in the first node has the form $w_1p_1CDw_2$ where $w_1, w_2 \in W$ and $p \in P$ has the word CD on its right hand side. If a symbol q, r_1 or r_3 is inserted, then the word leaves the network. This also happens, if a symbol $r_2 \neq p_2$ is inserted or p_2, p_4 or p_5 but not at a 'feasible' position.

Case 2.1. If p_2 is inserted and the word is accepted by the second node, then there are two possibilities:

Case 2.1.1. $p = A \to CD$. Then the word in the second node contains p_1CDp_2 as a subword. The only possibilities without loosing the word are now

- to delete the previously inserted p_2 – then we reach σ_2 again –, or
- to delete p_1 – if $A = D$, we reach σ_8, otherwise we loose the word –, or
- to delete D – then the word enters the third node, where only D can be inserted again in order not to loose the word and we obtain the same word in the second node as in the beginning of this subcase.

Case 2.1.2. $p = AB \to CD$. Then the word in the second node contains $p_1CDq^np_2$ as a subword. As it will turn out, this word can only be achieved by starting with the word $w_1Aq^nBp_2w_2$ in the first node, hence situation σ_9. The only possibilities without loosing the word are now

- to delete the previously inserted p_2 – then we reach σ_2 again –, or
- to delete D – then the word enters the third node, where D or B can be inserted (with D we have the same situation as before), with B inserted, the word goes to the second node which can delete B again (then we return to the same situation) or if $C = A$ it deletes p_1 and we reach situation σ_9 or it deletes C, then the word is communicated to the third node that inserts C again (same as before) or A and sends the word to the second node, it deletes A again (same as before) or p_1 such we reach the situation σ_9 –, or
- to delete C if $B = D$ (and $n = 0$) – then the word goes to the third node which inserts C again (as before) or A and sends the word to the second node, that deletes A again (as before) or p_1 such that we reach the situation σ_9 –, or
- to delete p_1 if $AB = CD$ and $n = 0$ – then we have the situation σ_9.

If we reach the situation σ_2, the word has not changed; if we reach σ_9, then the word is $w_1Aq^nBp_2w_2$. This word had been already in the network (in σ_9), otherwise the word $w_1p_1CDw_2$ (in the beginning of Case 2) could not have occurred. The word $w_1Aq^nBp_2w_2$ contains a non-terminal and has the same sentential form status is it already had.

Case 2.2. If the first node inserts r_4 and the second node accepts the word, then the second node deletes r_4 again and we return to situation σ_2 or it deletes p_1 and we reach the situation σ_5 or σ_6 or the second node deletes another symbol and the word vanishes. If we reach the situation σ_5 or σ_6, the pure word has not changed and hence is a sentential form with a non-terminal.

Case 2.3. If the first node inserts r_5 and the second node accepts the word, then the second node deletes r_5 again and we return to situation σ_2 or it deletes p_1 and we reach the situation σ_7 with the same pure word or the second node deletes another symbol and the word leaves the network.

Case 3. The word in the first node has the form $w_1p_1xw_2$ where $w_1, w_2 \in W$ containing a non-terminal and $p \in P$ is a rule $A \to x$. If a symbol q, r_1, $r_2 \neq p_2$ or r_3 is inserted or p_2 or a symbol r_4 or r_5 is inserted at a 'wrong' position, then the word leaves the network.

Case 3.1. If p_2 is inserted, then the word is accepted by the second node only, if it is $w_1p_1xp_2w_2$. The second node has to delete p_2 again or the word would disappear. So, we have situation σ_3 with the same word.

Case 3.2. If the first node inserts r_4 and the second node accepts the word, then the second node deletes r_4 again and we return to situation σ_3 or it deletes p_1 and we reach the situation σ_5 or σ_6 or the second node deletes another symbol, but then the word is lost. If we reach the situation σ_5 or σ_6, the pure word has not changed and hence it is a sentential form with a non-terminal.

Case 3.3. If the first node inserts r_5 and the second node accepts the word, then the second node deletes r_5 again and we return to situation σ_3 or it deletes p_1 and we reach the situation σ_7 with the same pure word or the second node deletes another symbol and the word leaves the network.

Case 4. The word in the first node has the form $w_1 p_1 q w_2$ where $w_1, w_2 \in W$ containing a non-terminal and $p \in P$ is a rule $A \to \lambda$. If a symbol q, r_1, r_2 or r_3 is inserted or a symbol r_4 or r_5 is inserted at an unacceptable position, then the word leaves the network.

Case 4.1. If the first node inserts r_4 and the second node accepts the word, then the second node deletes r_4 again and we return to situation σ_4 or it deletes p_1 and we reach the situation σ_5 or σ_6 or the second node deletes another symbol, but then the word is lost. If we reach the situation σ_5 or σ_6, the pure word has not changed and hence it is a sentential form with a non-terminal.

Case 4.2. If the first node inserts r_5 and the second node accepts the word, then the second node deletes r_5 again and we return to situation σ_4 or it deletes p_1 and we reach the situation σ_7 with the same pure word or the second node deletes another symbol and the word leaves the network.

Case 5. The word in the first node has the form $w_1 A p_4 w_2$ with $w_1, w_2 \in W$ and $p \in P$ being a rule $A \to x$ or $A \to CD$. If a symbol q, $r_2 \neq p_2$, r_3, r_4 or r_5 is inserted, then the word leaves the network.

Case 5.1. If r_1 is inserted and the word is accepted by the second node, then the second node must delete r_1 or p_4, otherwise we loose the word. If r_1 is deleted, we have the situation σ_5 again; if p_4 is deleted, we have one of the situations σ_2, σ_3 or σ_4 with the same pure word.

Case 5.2. If p_2 is inserted but not between A and p_4, the word is lost, otherwise it enters the second node. If there p_2 is deleted, we return to the situation σ_5 without having changed the pure word. If the second node deletes p_4, we reach the situation σ_8 with the same pure word. If the second node deletes another symbol, we loose the word.

Case 6. The word in the first node has the form $w_1 A q^n B p_4 w_2$ where $w_1, w_2 \in W$ and $p \in P$ is a rule $AB \to CD$. If a symbol q, $r_2 \neq p_2$, r_3, r_4 or r_5 is inserted, then the word disappears.

Case 6.1. If r_1 is inserted and the word is accepted by the second node, then the second node must delete r_1 or p_4, otherwise we loose the word. If r_1 is deleted, we have the situation σ_6 again; if p_4 is deleted, we have one of the situations σ_2, σ_3 or σ_4 with the same pure word.

Case 6.2. If p_2 is inserted but not between B and p_4, the word is lost, otherwise it enters the second node. If there p_2 is deleted, we return to the situation σ_6 without having changed the pure word. If the second node deletes p_4, we reach the situation σ_9 with the same pure word. If the second node deletes another symbol, we loose the word.

Case 7. The word in the first node has the form $w_1 A p_5 w_2$ with $w_1, w_2 \in W$ and $p = A \to \lambda \in P$. If a symbol r_2, r_3, r_4 or r_5 is inserted, then the word disappears.

Case 7.1. If r_1 is inserted and the word is accepted by the second node, then the second node must delete r_1 or p_5, otherwise the word is lost. If r_1 is deleted, we have the situation σ_7; if p_5 is deleted, we have one of the situations σ_2, σ_3 or σ_4 with the same pure word.

Case 7.2. If q is inserted but not between A and p_5, the word is lost, otherwise it enters the second node. If there the q between A and p_5 is deleted, we return to the situation σ_7 with the same pure word. If the second node deletes p_5, we reach the situation σ_{10} with the unchanged pure word. If the second node deletes another symbol, the word disappears.

Case 8. The word in the first node has the form $w_1 A p_2 w_2$ with $w_1, w_2 \in W$ and $p \in P$ being a rule $A \to x$ or $A \to CD$. If a symbol $r_1 \neq p_1$, r_2, $r_3 \neq p_3$, $r_4 \neq p_4$ or r_5 is inserted, then the word leaves the network.

Case 8.1. If q is inserted between A and p_2, we loose the word, if it is inserted somewhere else, then the word remains in the node and we stay in the situation σ_8 with the same pure word.

Case 8.2. If p_1 is inserted, then there are two possibilities not to loose the word:

Case 8.2.1. $p = A \to CA$ and the word contains p_1CAp_2 as a subword. Then the second node deletes p_1 again and we return to the situation σ_8 or it deletes p_2 and we obtain the situation σ_2 with the same pure word or the second node deletes the A between C and p_2, then the word moves to the third node, where this A has to be inserted again or the word is lost. If the second node deletes another symbol, the word leaves the network.

Case 8.2.2. $p = A \to x$ and the word contains p_1Ap_2 as a subword. Then the second node deletes the A between p_1 and p_2 (otherwise the word disappears). The third node inserts the A again (the we are in the same situation as before) or x at the same position (or the word vanishes). If x is inserted between p_1 and p_2, then the word moves to the second node. If there p_1 or a non-terminal is deleted, the word disappears. So, the second node has to delete p_2. If no non-terminal is left, we have reached the situation σ_{11}, otherwise we have the situation σ_3. Since we assume that the pure word we started with is a sentential form of the grammar G, the pure word now in situation σ_{11} or σ_3 is a sentential form, too, because the transition in this subcase can be regarded as a simulation of the application of the rule $A \to x$ in the grammar G.

Case 8.3. Let the first node insert p_3. Then the rule p has the form $A \to CD$, because for the form $r = A \to x$, there is no symbol r_3 in the alphabet V. If p_3 is inserted behind p_2, the word stays in the first node, otherwise it leaves the network. If now qs are inserted, the word is not lost, as long as the subword Ap_2p_3 is not affected. The only other possibility to insert a symbol and to keep the word is to insert p_1 before A. Then, the word enters the second node. If this node deletes p_1 again, we return to the situation as before. If $A \neq C$, then the only other way not to loose the word is to delete the A between p_1 and p_2. Then the word moves to the third node. This node inserts A again between p_1 and p_2 (same as before) or it inserts C between p_1 and p_2. If another insertion takes place, we loose the word. Then, the second node receives the word containing $p_1Cp_2p_3$ as a subword. If we have $A = C$, then we can skip the last two rewriting and two communication steps (deleting A and inserting C), because $p_1Ap_2p_3$ and $p_1Cp_2p_3$ are not distinguishable. The second node deletes the C of this subword (if $A \neq C$ then we get a situation we already had, otherwise the word moves to the third node which has to insert C at the same position again in order not to loose the word) or it deletes p_3 or the word disappears. If p_3 is deleted, then the word is communicated to the third node. The only possibility not to loose the word is to insert D between C and p_2. Then the word moves to the second node. There this D can be deleted to return to the previous situation. If p_2 is deleted, we reach the situation σ_2. If p_1 is deleted, then the third node does not accept the word and the first node does it only if $D = A$; then we reach the situation σ_8 again. Since we assume that we started with the pure word being a sentential form, the pure word in the new situation is a sentential form, too, because the transition in this subcase can be seen as a simulation of the application of the rule $A \to CD$ in the grammar G.

Case 8.4. If p_4 is inserted but not behind p_2, the word vanishes. If it is inserted behind p_2, the word moves to the second node. This node must delete p_2 or p_4; otherwise we would loose the word. If p_4 is deleted, we are again in the situation σ_8; if p_2 is deleted, we reach the situation σ_5; in both cases, the pure word is not changed.

Case 9. The word in the first node has the form $w_1 A q^n B p_2 w_2$ with $w_1, w_2 \in W$, $n \geq 0$ and $p \in P$ being a rule $AB \to CD$. If a q is inserted between B and p_2 or a symbol $r_1 \neq p_1$, r_2, r_3, $r_4 \neq p_4$ or r_5 is inserted, then the word leaves the network. This also happens, if p_1 is inserted but not before A or p_4 is inserted but not behind p_2.

Case 9.1. As long as qs are inserted but not between B and p_2, the word remains in the node and neither the pure word nor the situation are changed.

Case 9.2. Suppose p_1 is inserted before A. The word enters the second node. If p_1 is deleted there, we return to the situation σ_9 without changing the pure word. If p_2 is deleted, the word is not lost if $p = AB \to AB$ and $n = 0$ where we reach the situation σ_2 with the same pure word. If a q or a non-terminal outside the subword $p_1 A q^n B p_2$ is deleted, the word is lost.

Case 9.2.1. If the A of the subword is deleted, the word enters the third node. If neither A or C is inserted behind p_1, the word disappears. If A is inserted, we return to the situation we just had. If C is inserted, the word moves to the second node. If there this C is deleted, we return to the previous situation. If a q or a non-terminal outside the subword $p_1 C q^n B p_2$ is deleted, the word disappears. If p_1 is deleted, then the word is not accepted by the third node and it is accepted by the first node only in the case that $C = A$; then we have the situation σ_9 again with the same pure word. If p_2 is deleted, the word does not pass the input filter of the third node and it is accepted by the first node only if $D = B$ and $n = 0$; then we have reached the situation σ_2 and have 'applied' the rule $p = AB \to CB$ to the pure word which is assumed to be a sentential form of the grammar G, hence we have obtained another sentential form. If the second node deletes B from the subword $p_1 C q^n B p_2$, the word enters the third node. If there this B is inserted again, we return. Otherwise, the only possibility to keep the word is to insert D behind C. Then the word $w_1 p_1 C D q^n p_2 w_2$ is communicated to the second node. If the D inserted last is deleted, we return. This is also the case, if the node deletes C and $C = D$. If the node deletes C and $C \neq D$ or it deletes a q, p_1 or a non-terminal outside the subword $p_1 C D q^n p_2$, then the word disappears. If p_2 is deleted, the word enters the first node and we reach the situation σ_2. We have 'applied' the rule $p = AB \to CD \in P$ to the pure word in the beginning of the Case 9. Hence, if this word was a sentential form, we end up with another sentential form in the situation σ_2.

Case 9.2.2. If the B of the subword $p_1 A q^n B p_2$ is deleted, the word enters the third node, if $A = C$ or vanishes otherwise. If the third node inserts B again, we return. The only other possibility to keep the word is to insert D behind A. Then, the word $w_1 p_1 A D q^n p_2 w_2$ is communicated to the second node. If the D is deleted or the A is deleted and $A = D$, we return to the previous situation. If the node deletes A and $A \neq D$ or it deletes a q, p_1 or a non-terminal outside the subword $p_1 A D q^n p_2$, then the word disappears. If p_2 is deleted, the word enters the first node and we obtain the situation σ_2, having simulated the application of the rule $p = AB \to AD \in P$ to the pure word in the beginning of the Case 9 which is assumed to be a sentential form. Hence, we have another sentential form in the situation σ_2.

Case 9.3. Suppose p_4 is inserted behind p_2. The word $w_1 A q^n B p_2 p_4 w_2$ enters the second node. There, p_2 or p_4 has to be deleted in order to keep the word in the network. If p_4 is deleted, we return to the situation σ_9; if p_2 is deleted, we reach the situation σ_6; in both cases without changing the pure word.

Case 10. The word in the first node has the form $w_1 A q w_2$ with $w_1, w_2 \in W$ and $A \to \lambda \in P$. If a symbol p_3 or p_4 is inserted, we loose the word. Otherwise, we have the following possibilities:

Case 10.1. If the first node inserts a q, the pure word is not changed and we stay in the situation σ_{10}.

Case 10.2. The first node inserts p_1. If the word afterwards contains a subword $p_1 A q$ where $p = A \to \lambda \in P$, then the word enters the second node (note that the word is not necessarily $w_1 p_1 A q w_2$), if not, then the word disappears. In order to keep the word 'alive', the second node has to delete p_1 or the A next to it on its right hand side. If p_1 is deleted, we have not changed the pure word and return to the situation σ_{10}.

Case 10.2.1. If A is deleted but there is another non-terminal left, then the word moves to the first node. The situation is σ_4 and the pure word is the result of applying the rule $p = A \to \lambda \in P$ to the pure word in the beginning of the Case 10.

Case 10.2.2. If the deleted A was the last non-terminal in the word, then the word remains in the second node and we reached the situation σ_{11}. The pure word has been derived from the pure word in the beginning of the Case 10 by the rule $p = A \to \lambda \in P$. Hence, it is a sentential form of the grammar G, if the pure word before was a sentential form, too.

Case 10.3. If the first node inserts p_2, then either the word leaves the network or it remains in the first node. If it stays there, we have the situation σ_8 or σ_9 with the same pure word.

Case 10.4. Let the first node insert p_5. If the word afterwards contains a subword $A q p_5$ with $p = A \to \lambda \in P$, then the word enters the second node (note that the word is not necessarily $w_1 A q p_5 w_2$), otherwise it disappears. If the second node does not delete the q between A and p_5 or p_5, the word leaves the network. If it deletes p_5, we return to the situation σ_{10} with the same pure word. If it deletes the q between A and p_5, we reach the situation σ_7 with the same pure word.

Case 11. The word in the second node has the form $w_1 p_1 w_2$ with $w_1, w_2 \in T_q$ and $p \in P$. As long as the second node deletes qs, the word stays in the node and the situation does not change (neither does the pure word). If p_1 is deleted while there is still a q left, the word disappears. If there is no q left, when p_1 is deleted, then the word only consists of terminal symbols. This terminal word is send to the first node. Hence, we have reached situation σ_{12}.

Case 12. The word in the first node is terminal. If this node inserts a symbol p_1 for a rule $p = A \to x$ left to a letter x, the word is accepted by the second node, which leads us to the situation σ_{11}. In all other cases, the word leaves the network.

Since we start with a sentential form of the grammar G in the situation σ_1 and we obtain in every situation a sentential form from another sentential form, the terminal word in the situation σ_{12} is a word of the language $L(G)$ generated by G. Other terminal words are not produced.

Since every terminal word which is generated by the network $\mathcal{N}$ is also a sentential form (hence a word) of the grammar G, the inclusion $L(\mathcal{N}) \cap T^* \subseteq L(G)$ holds. Together with the first part, we have proved the claim $L(\mathcal{N}) \cap T^* = L(G)$. □

Corollary 5.2 *There is a network $\mathcal{N}$ of evolutionary processors with two insertion nodes and one deletion node such that $L(\mathcal{N})$ is a non-recursive language.*

Proof. Since the family of recursive languages is closed under intersection with sets T^*, where T is an alphabet, the network constructed in the proof of Lemma 5.1 for a non-recursive language L generates a non-recursive language. □

Corollary 5.3 *For any recursively enumerable language L there is a network $\mathcal{N}$ of evolutionary processors with four nodes which are insertion nodes and deletion nodes such that $L = L(\mathcal{N})$.*

Proof. The proof can be given analogously to that of Corollary 4.3. □

Obviously, any language generated by a network of evolutionary processors with only insertion and deletion nodes is recursively enumerable since arbitrary networks of evolutionary processors only generate recursively enumerable languages. Thus we get the following statement by Lemma 5.1.

Corollary 5.4 *The family of networks of evolutionary processors which have only insertion and deletion nodes coincides with the family of recursively enumerable languages.* □

6 Conclusion

In the paper we have determined the power of networks of evolutionary processors if only two different types of nodes are used in the network. We have shown that

- up to an intersection with a monoid every recursively enumerable language can be generated by a network with one deletion and two insertion nodes,
- networks with an arbitrary number of deletion and substitution nodes only produce finite languages, and for each finite language one deletion node or one substitution node is sufficient, and
- networks with an arbitrary number of insertion and substitution nodes only generate context-sensitive languages, and (up to an intersection with a monoid) every context-sensitive language can be generated by a network with one substitution node and one insertion node.

The latter two results are optimal with respect to the minimal number of necessary nodes, whereas it is an open problem whether or not one deletion and one insertion node are sufficient to generate all recursively enumerable languages.

If one considers networks with all three types of nodes, it is known that it is not necessary to allow all graphs. One can obtain all recursively enumerable languages if one restricts to special graphs e. g. to those which are known as useful structures in technology as grids or rings (see [7], [5]). Obviously, the restriction to complete graphs does not restrict the power in the case of networks with nodes of two types, either, because the graph given in the proof of Lemma 4.2 is complete and we can extend the network of Lemma 5.1 to a language equivalent network with a complete underlying graph (adding the edge $(1,3)$ enforces the output filter of the first processor to be changed to $O_1 = V^* \setminus (W(\beta_4' \cup \beta_8)W)$; for adding the edge $(3,1)$, no changes are necessary). These graphs can be extended further to complete graphs according to those given in the proofs of the Corollaries 4.3 and 5.2. Due to the input and output filters of the new nodes, the new edges have no influence to the language generated.

Moreover, the graphs in the proofs of the Corollaries 4.3 and 5.2 are stars (if we ignore the directions) which proves that the restriction to stars does not decrease the power. The same situation holds with respect to backbones. A general investigation of special graphs remains as a task.

Analogously, one also gets all recursively enumerable languages from networks with all three types of nodes, if one restricts the form of the regular sets e. g. to random context sets, where one requires the presence and/or absence of some letters in the word (see [5]). The languages we used in our proofs are more complicated since they require absence and/or presence of some subwords. We leave as an open problem the power of networks with two types of nodes and random context regular sets.

References

[1] A. ALHAZOV, C. MARTIN-VIDE and YU. ROGOZHIN, On the number of nodes in universal networks of evolutionary processors. *Acta Inf.* **43** (2006) 331–339.

[2] J. CASTELLANOS, P. LEUPOLD and V. MITRANA, On the size complexity of hybrid networks of evolutionary processors. *Theor. Comput. Sci.* **330** (2005) 205–220.

[3] J. CASTELLANOS, C. MARTIN-VIDE, V. MITRANA and J. SEMPERE, Solving NP-complete problems with networks of evolutionary processors. In: *Proc. IWANN*, Lecture Notes in Computer Science **2084**, Springer-Verlag, Berlin, 2001, 621–628.

[4] J. CASTELLANOS, C. MARTIN-VIDE, V. MITRANA and J. SEMPERE, Networks of evolutionary processors. *Acta Informatica* **38** (2003) 517–529.

[5] E. CSUHAJ-VARJÚ, C. MARTIN-VIDE and V. MITRANA, Hybrid networks of evolutionary processors are computationally complete. *Acta Informatica* **41** (2005) 257–272.

[6] E. CSUHAJ-VARJÚ and A. SALOMAA, Networks of parallel language processors. In: *New Trends in formal Language Theory* (Eds. GH. PĂUN and A. SALOMAA), Lecture Notes in Computer Science **1218**, Springer-Verlag, Berlin, 1997, 299–318.

[7] J. DASSOW, On special networks of parallel language processors. *Romanian Journal of Information Science and Technology* **1** (1998) 331–341.

[8] J. DASSOW and GH. PĂUN, *Regulated Rewriting in Formal Language Theory*. Springer-Verlag, Berlin, 1989.

[9] S. E. FAHLMANN, G. E. HINTON and T. J. SEIJNOWSKI, Massively parallel architectures for AI: NETL, THISTLE and Boltzmann machines. In: *Proc. AAAI National Conf. on AI*, William Kaufman, Los Altos, 1983, 109–113.

[10] W. D. HILLIS, *The Connection Machine*. MIT Press, Cambridge, 1985.

[11] M. MARGENSTERN, GH. PĂUN, YU. ROGOZHIN and S. VERLAN, Context-free insertion-deletion systems. *Theor. Comput. Sci.* **330** (2005) 339–348.

[12] G. ROZENBERG and A. SALOMAA, *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.