SYNCHRONIZED EXTENSION SYSTEMS

Ferucio Laurenţiu Ţiplea, Erkki Mäkinen
and Corina Apachite

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES

UNIVERSITY OF TAMPERE

REPORT A-2000-1

UNIVERSITY OF TAMPERE
DEPARTMENT OF COMPUTER AND
INFORMATION SCIENCES
SERIES OF PUBLICATIONS A
A-2000-1, JANUARY 2000
REVISED DECEMBER 2000

SYNCHRONIZED EXTENSION SYSTEMS

Ferucio Laurenţiu Ţiplea, Erkki Mäkinen
and Corina Apachite

University of Tampere
Department of Computer and Information Sciences
P.O.Box 607
FIN-33014 University of Tampere, Finland

ISBN 951-44-4743-3
ISSN 1457-2060

Synchronized Extension Systems [1]

Ferucio Laurenţiu ŢIPLEA [a] **Erkki MÄKINEN** [b]
Corina APACHITE [a]

[a] Faculty of Computer Science
"Al. I. Cuza" University of Iaşi
6600 Iaşi, Romania
E-mail: `fltiplea@infoiasi.ro`

[b] Department of Computer and Information Sciences
P.O. Box 607, FIN-33014 University of Tampere, Finland
E-mail: `em@cs.uta.fi`

Abstract

Synchronized extension systems (SE-systems, for short) are 4-tuples $G = (V, L_1, L_2, S)$, where V is an alphabet and L_1, L_2 and S are languages over V. They generate languages extending L_1 by L_2 to the left or to the right, and synchronizing on words in S. Such systems appear naturally when considering stacks, queues, grammar-like generative devices, splicing systems, zigzag-codes etc.

1 Introduction

There are many cases where we need to extend some words to the right (or to the left), by taking into consideration some of the last letters of the word. For example, the words in a pushdown stack are generated by such a rule: only the last letter is used for synchronization (considering the top of the stack word as its right end).

We generalize this idea by introducing the concept of an SE-system and provide several examples to show that our SE-systems are poweful enough to handle various generative devices. Then we study some of their basic properties and generalize a theorem in [2] regarding the regularity of the stack language of a pushdown automaton. Encoding by a code and encoding by a zigzag-code means generation of a word by a (proper) synchronized extension to the right. This topic is discussed in sections 4 and 5. Finally, we generalize SE-systems in such a way that also splicing systems can be handled by them.

We recall now a very few basic concepts and notations on formal languages (for further details the reader is referred to [13]). For a finite non-empty set

[1]The work of the first author was carried out while visiting Universität Augsburg by a grant from *Deutschen Akademien der Wissenschaften* (Germany).
The work of the second author was supported by the Academy of Finland (Project 35025).

V, called *alphabet*, V^* is the free monoid generated by V under the *catenation operation*, and λ is its unity (*empty word*). V^+ stands for $V^* - \{\lambda\}$. The notation $u \leq_{pref} v$ ($u \leq_{suff} v$) means that the word u is a *prefix* (*suffix*) of the word v.

(Chomsky) grammars will be considered as 4-tuples $G = (V_N, V_T, X_0, P)$, where V_N and V_T are the alphabets of *nonterminals* and *terminals*, resp., $X_0 \in V_N$ is the *axiom*, and P is the set of *productions*. Nonterminals will be denoted by capital letters, while lower case letters are used for terminals. The derivation relation associated to G is denoted by $\Rightarrow_G$, or by $\Rightarrow$, when G is clear from the context.

2 SE-systems

2.1 Definitions ans Examples

Definition 2.1.1 *A* synchronized extension system *(SE-system, for short) is a 4-tuple $G = (V, L_1, L_2, S)$, where V is an alphabet and L_1, L_2 and S are languages over V. L_1 is called the* initial language*, L_2 the* extending language*, and S the* synchronization set *of G.*

Definition 2.1.2 *Let $G = (V, L_1, L_2, S)$ be an SE-system. Define the binary relations $\Rightarrow_{G,r}$, $\Rightarrow_{G,r^-}$, $\Rightarrow_{G,l}$ and $\Rightarrow_{G,l^-}$ over V^* as follows:*

(i) $u \Rightarrow_{G,r} v$ *iff* $(\exists w \in L_2)(\exists s \in S)(\exists x, y \in V^*)(u = xs \wedge w = sy \wedge v = xsy)$;

(ii) $u \Rightarrow_{G,r^-} v$ *iff* $(\exists w \in L_2)(\exists s \in S)(\exists x, y \in V^*)(u = xs \wedge w = sy \wedge v = xy)$;

(iii) $u \Rightarrow_{G,l} v$ *iff* $(\exists w \in L_2)(\exists s \in S)(\exists x, y \in V^*)(u = sx \wedge w = ys \wedge v = ysx)$;

(iv) $u \Rightarrow_{G,l^-} v$ *iff* $(\exists w \in L_2)(\exists s \in S)(\exists x, y \in V^*)(u = sx \wedge w = ys \wedge v = yx)$.

In an SE-system $G = (V, L_1, L_2, S)$, the words in S act as synchronization words; they can be kept or neglected in the final result. r, r^-, l and l^- are called (basic) *modes of synchronizations*. They can be used to define another four new modes of synchronization, (l, r), (l^-, r), (l, r^-) and (l^-, r^-), by relational union. For example, the relation $\Rightarrow_{G,(l^-,r)}$ is defined by:

$$u \Rightarrow_{G,(l^-,r)} v \text{ iff } u \Rightarrow_{G,l^-} v \vee u \Rightarrow_{G,r} v.$$

A derivation $u \overset{*}{\Rightarrow}_x v$, where x is a mode of synchronization, is called an *x-derivation* (*of u into v*, or *of v from u*, or *of v*, or *from u*). Two x-derivations

$$u_1 \Rightarrow_x u_2 \Rightarrow_x \cdots \Rightarrow_x u_n$$

and

$$u'_1 \Rightarrow_x u'_2 \Rightarrow_x \cdots \Rightarrow_x u'_m$$

are called *distinct* if $n \neq m$ or there is an i such that $u_i \neq u'_i$.

Definition 2.1.3 *The* language of type x *generated by an* SE-system $G = (V, L_1, L_2, S)$*, where x is a mode of synchronization, is defined by:*

$$L^x(G) = \{v \in V^* | \exists u \in L_1 : \ u \overset{*}{\Rightarrow}_{G,x} v\}.$$

Definition 2.1.4 *Let $G = (V, L_1, L_2, S)$ be an SE-system and let p_1, p_2 and p_3 be predicates on $\mathcal{P}(V^*)$* [2]*. We say that G is* of type (p_1, p_2, p_3) *if the formula $p_1(L_1) \ \wedge \ p_2(L_2) \ \wedge \ p_3(S)$ holds true.*

We shall use the abbreviation f (reg) for the predicate "L is finite (regular)".

Example 2.1.1 *Let $G = (V_N, V_T, X_0, P)$ be a context-free grammar. We consider the* SE-system $G' = (V, L_1, L_2, S)$*, where:*

- $V = V_N \cup V_T$,
- $L_1 = \{X_0\}$,
- $L_2 = \{u\alpha uA | u \in V_T^*, \ A \to \alpha \in P\}$,
- $S = \{uA | u \in V_T^*, \ A \in V_N\}$.

Then, $u_1 A u_2 \Rightarrow_G u_1 \alpha u_2$ iff $u_1 A u_2 \Rightarrow_{G',l-} u_1 \alpha u_2$, for every $u_1 \in V_T^*$, $u_2, \alpha \in V^*$ and $A \in V_N$ (the derivations in G are considered leftmost). Therefore we have $L(G) = L^{l-}(G') \cap V_T^*$.

Example 2.1.2 *Let $G = (V_N, V_T, X_0, P)$ be a regular (right-linear) grammar. We consider the* SE-system $G' = (V, L_1, L_2, S)$*, where:*

- $V = V_N \cup V_T$,
- $L_1 = \{X_0\}$,
- $L_2 = \{AaB | A \to aB \in P\} \cup \{Aa | A \to a \in P\} \cup \{A | A \to \lambda \in P\}$,
- $S = V_N$.

Then, G' is of type (f, f, f) and $L(G) = L^{r-}(G') \cap V_T^*$.

2.2 SE-systems and Pure Grammars

It is argued [3, 11] that the custom of dividing the alphabet of a grammatical system originates from the linguistic background of formal language theory and in the fact, it would be more natural to study rewriting systems that do not make a difference between terminals and nonterminals. So called pure grammars do not have nonterminal symbols, and their generative capacity differs from that of the corresponding Chomsky-type grammars.

[2]A predicate on a non-empty set A is a function from A into the set $\{0, 1\}$.

A *pure grammar* is a system $H = (\Sigma, P, \sigma)$ where Σ is an alphabet, the set of axioms σ is a finite subset of Σ^+, and P is a finite set of productions of the form $x \to y$, where x and y are words over Σ. Relation $\Rightarrow_H$ ("yields directly") and its reflexive transitive closure $\Rightarrow_H^*$ are defined in Σ^* as usual. The language generated by a pure grammar $H = (\Sigma, P, \sigma)$ is defined as ([6, 11])

$$L(H) = \{\beta | \exists \alpha \in \sigma : \ \alpha \Rightarrow_H^* \beta\}.$$

SE-systems are "pure" in the sense they do not have separate alphabets of terminal and nonterminal symbols. In this section we discuss the relationship between SE-systems and pure grammars. Since we suppose that a pure grammar has a finite set σ of initial words, we consider here only SE-systems of the type (f, x, x), i.e., SE-systems with a finite set L_1.

Pure grammars are normally divided into subclasses depending on the form of their productions (e.g. pure context-free grammars and pure length increasing grammars [11]). However, the notation of SE-systems is so powerful that we can handle all pure grammars together.

Let $H = (\Sigma, P, \sigma)$ be a pure grammar. We define an SE-system $G = (V, L_1, L_2, S)$ such that $L(H) = L^{r-}(G) \cap \Sigma^*$. We set

- $V = \Sigma \cup \{\#\}$, where # is a new symbol,
- $L_1 = \sigma\{\#\} \cup \sigma$,
- $L_2 = \{xu\#yu\#,\ xu\#yu \mid u \in \Sigma^* \ \wedge \ x \to y \in P\}$,
- $S = \{xu\# | u \in \Sigma^* \ \wedge \ (\exists y)(x \to y \in P)\}$.

Therefore, we have

$$\alpha \overset{*}{\Rightarrow}_H \beta \quad \textit{iff} \quad \alpha\# \overset{*}{\Rightarrow}_{G,r^-} \beta,$$

for all $\alpha, \beta \in \Sigma^*$. Hence, we have $L(H) = L^{r-}(G) \cap \Sigma^*$.

We have proved the following result.

Theorem 2.2.1 *All pure languages can be written as an intersection between a language of type r^- generated by an SE-system and a regular language.*

A *pure regular system* ([4]) is a pure grammar $H = (\Sigma, P, \sigma)$ in which rewriting is restricted to the left hand side end of words, i.e., $x \Rightarrow_H y$ if and only if $x = uw$, $y = vw$ and $u \to v \in P$, for some $u, v, w \in \Sigma^*$.

Example 2.2.1 *Let $H = (\Sigma, P, \sigma)$ be a pure regular system. We can define an SE-system $G = (V, L_1, L_2, S)$ of type (f, f, f) such that $L(H) = L^{l-}(G) \cap \Sigma^*$. Namely, we can set*

- $V = \Sigma \cup \{\#\}$*, where # is a new symbol,*

- $L_1 = \{\#\}\sigma \cup \sigma$,
- $L_2 = \{\#v\#u,\ v\#u \mid u \to v \in P\}$,
- $S = \{\#u \mid \exists v :\ u \to v \in P\}$.

Now, $\alpha \overset{*}{\Rightarrow}_H \beta$ *iff* $\#\alpha \overset{*}{\Rightarrow}_{G,l^-} \beta$, *for all* $\alpha, \beta \in \Sigma^*$. *Hence,* $L(H) = L^{l-}(G) \cap \Sigma^*$.

2.3 SE-systems and Conditional Grammars

SE-systems of type (reg, f, f) in the mode r of synchronization have the power of regular grammars. We will show this by establishing a connection with conditional grammars ([5]).

Let $\mathcal{L}$ be a family of languages. An *$\mathcal{L}$-conditional grammar of type i* ($i = 0, 1, 2, 3$) is a couple $\gamma = (G, \varphi)$, where $G = (V_N, V_T, X_0, P)$ is a grammar of type i and φ is a function from P into $\mathcal{P}((V_N \cup V_T)^*) \cap \mathcal{L}$. The derivation relation induced by γ is defined by

$$x \Rightarrow_\gamma y \quad \Leftrightarrow \quad x = x_1 \alpha x_2 \ \wedge\ y = y_1 \beta y_2 \ \wedge\ \alpha \to \beta \in P \ \wedge\ x \in \varphi(\alpha \to \beta)$$

for every $x, y \in (V_N \cup V_T)^*$, and the language generated by γ, denoted by $L(\gamma)$, is defined as usual.

Theorem 2.3.1 *For each SE-system G of type (reg, f, f), it is possible to effectively construct an $\mathcal{L}_3$-conditional grammar γ of type 3 such that $L(\gamma) = L^r(G)$.*

Proof Let $G = (V, L_1, L_2, S)$ be an SE-system of type (reg, f, f). We define an $\mathcal{L}_3$-conditional grammar $\gamma = (G', \varphi)$ of type 3, as follows:

- $G' = (V_N, V_T, X_0, P)$,
- $V_N = \{X_0, Y, Y_s, Y_{s'}\}$,
- $V_T = V$,
- P contains the rules (for each rule we specify also the language associated with φ):
  1. for any $a \in V$ consider the rule $X_0 \to aX_0$ with the associated language $V^*\{X_0\}$;
  2. for any $a \in V$ consider the rule $X_0 \to a$ with the associated language $\partial_a^r(L_1)\{X_0\}$, where $\partial_a^r(L)$ stands for the set $\{w \mid aw \in L\}$;
  3. if $\lambda \in L_1$ then consider the rule $X_0 \to \lambda$ with the associated language $\{X_0\}$;
  4. for any $a \in V$ and any $s \in S$ consider the rule $X_0 \to aY_s$ with the associated language $\partial_a^r(L_1)\{X_0\}$;

5. for any $s, s' \in S$ and any w such that $sw \in L_2$ consider the rule $Y_s \to wY_{s'}$ with the associated language $V^*\{s\}\{Y_{s'}\}$;
6. for any $s \in S$ and any w such that $sw \in L_2$ consider the rule $Y_s \to w$ with the associated language $V^*\{s\}\{Y_s\}$.

It is easy to see that γ is an $\mathcal{L}_3$-conditional grammar of type 3 and $L(\gamma) = L^r(G)$. □

As languages generated by $\mathcal{L}_3$-conditional grammars of type 3 are regular ([12]), we obtain that languages generated by SE-systems of type (reg, f, f) in the mode r of synchronization, are regular, too. A more general result will be developed in the next section.

3 Some Basic Properties

For a word u, by $\widetilde{u}$ we denote *the mirror image* of u. We extend this unary operation to languages, as usual.

If $G = (V, L_1, L_2, S)$ is an SE-system, then $\widetilde{G} = (V, \widetilde{L_1}, \widetilde{L_2}, \widetilde{S})$ is called the *mirror image of* G. It is clear that if G is of type (p_1, p_2, p_3), then $\widetilde{G}$ is of the same type iff

$$(p_1(L_1) \Leftrightarrow p_1(\widetilde{L_1})) \wedge (p_2(L_2) \Leftrightarrow p_2(\widetilde{L_2})) \wedge (p_3(S) \Leftrightarrow p_3(\widetilde{S})).$$

Remark 3.1 *Let $G = (V, L_1, L_2, S)$ be an SE-system. Then, $L^l(G) = \widetilde{L^r(\widetilde{G})}$ and $L^{l^-}(G) = \widetilde{L^{r^-}(\widetilde{G})}$.*

Remark 3.2 *Let $G = (V, L_1, L_2, S)$ be an SE-system.*

(1) We may assume that $\lambda \notin L_2$ without changing any of the languages generated by G.

(2) If L_2 is finite, then S can be considered finite without changing any of the languages generated by G. Indeed, the set

$$S' = \{s \in S | \exists v \in L_2 : \ s \leq_{pref} v \ \vee \ s \leq_{suff} v\}$$

is finite and the only synchronization words that can be used in derivations are those from S'. Therefore, we can replace S by S' without changing any of the languages generated by G.

However, if S is finite then we cannot generally replace L_2 by a finite language.

(3) If $S = \{\lambda\}$, then $L^r(G) = L^{r^-}(G) = L_1 L_2^*$.

(4) If S is a prefix code [3]*, then $L^r(G) = L^{r^-}(G')$, where $G' = (V, L_1, L_2', S)$ and $L_2' = \{ssv | s \in S,\ sv \in L_2\}$.*

(5) If $S = L_2$ and L_2 is a prefix code, then $L^r(G) = L_1$ and $L^{r^-}(G) = L_1 \cup (L_1/L_2)$ [4]*.*

(6) Let $G' = (V \cup \{\#\}, L_1\{\#\}, L_2'\{\#\}, S\{\#\})$, where $L_2' = \{s\#sv\# | s \in S,\ sv \in L_2\}$. Then,

$$u \Rightarrow_{G,r} v \quad \Leftrightarrow \quad u\# \Rightarrow_{G',r^-} v\#,$$

for all $u, v \in V^*$, which shows that $L^r(G) = \partial^r_{\#}(L^{r^-}(G'))$.

Let $G = (V, L_1, L_2, S)$ be an SE-system. We want to define a new SE-system G' such that the r^--synchronization in G' is always possible by using at most two symbols and, each time, either one symbol is deleted or the last one is changed or one new symbol is appended. Define the following sets:

- $V_{\triangleleft} = \{\triangleleft_{s'} | \exists s \in S:\ s' \leq_{suff} s\}$, and
  $V_{\triangleright} = \{\triangleright_{v'} | \exists s \in S, \exists v \in V^*:\ sv \in L_2\ \wedge\ v' \leq_{suff} v\}$;
- $S' = \{a \triangleleft_{s'} | a \in V, \exists s \in S:\ as' \leq_{suff} s\} \cup \{\triangleleft_s | s \in S\} \cup V_{\triangleright}$;
- $L_1' = \{u \triangleleft_{\lambda} | u \in L_1\}$;
- $L_2' = \{a \triangleleft_{s'} \triangleleft_{as'} | a \triangleleft_{s'} \in S'\} \cup$
  $\{\triangleleft_s \triangleright_v | s \in S, v \in V^*:\ sv \in L_2\} \cup$
  $\{\triangleright_{av'} a \triangleright_{v'} | a \in V, \triangleright_{av'} \in S'\} \cup$
  $\{\triangleright_{\lambda} \triangleleft_{\lambda}\}$.

Let $G' = (V \cup V_{\triangleleft} \cup V_{\triangleright}, L_1', L_2', S')$. Any r^--derivation in G,

$$u = u's \Rightarrow_{G,r^-} u'v,$$

where $s \in S$ and $sv \in L_2$, can be simulated in G' by the following r^--derivation:

1. if $s = \lambda$, then
$$u \triangleleft_{\lambda} \Rightarrow_{G',r^-} u \triangleright_v$$
   (a) if $v = \lambda$, the above derivation can be continued by
$$u \triangleright_{\lambda} \Rightarrow_{G',r^-} u \triangleleft_v = uv \triangleleft_{\lambda}$$

[3]A *prefix code* over an alphabet V is a subset of words over V such that no word in this subset is a proper prefix of any other word in this subset.

[4]A/B denotes the right quotient of A by B.

(b) if $v = v_1 \cdots v_l \in V^l$, $l \geq 1$, the above derivation can be continued by

$$u\triangleright_{v_1\cdots v_l} \Rightarrow_{G',r^-} uv_1\triangleright_{v_2\cdots v_l} \overset{*}{\Rightarrow}_{G',r^-} uv_1\cdots v_l\triangleright_\lambda \Rightarrow_{G',r^-} uv\triangleleft_\lambda$$

2. if $s = s_1 \cdots s_k \in V^k$, $k \geq 1$, then

$$u's_1\cdots s_k\triangleleft_\lambda \Rightarrow_{G',r^-} u's_1\cdots s_{k-1}\triangleleft_{s_k} \overset{*}{\Rightarrow}_{G',r^-} u'\triangleleft_{s_1\cdots s_k} \Rightarrow_{G',r^-} u'\triangleright_v .$$

From this point the derivation is continued as in the cases above.

Then it is easy to see that we have

$$L^{r^-}(G) = \partial^r_{\triangleleft_\lambda}(L^{r^-}(G')).$$

If we replace the set L'_2 by $L'_2 \cup \{\triangleright_a a | a \in V,\ \triangleright_a \in S'\}$, then we have

$$L^{r^-}(G) = L^{r^-}(G') \cap V^*.$$

We say that in the first case G and G' are *∂-equivalent*, and in the second case, *$\cap$-equivalent*.

The SE-system G' has the following features:

(i) the alphabet is divided into three sets: V, $V_\triangleleft$, and $V_\triangleright$;

(ii) $L'_1 \subseteq V^*V_\triangleleft \cup V^*V_\triangleright$;

(iii) $L'_2 \subseteq VV_\triangleleft V_\triangleleft \cup V_\triangleright VV_\triangleright \cup V_\triangleleft V_\triangleright \cup V_\triangleright V_\triangleleft$;

(iv) $S' \subseteq V_\triangleleft \cup V_\triangleright \cup VV_\triangleleft$;

(v) there is a special symbol $\triangleleft_\lambda \in V_\triangleleft$ which "closes" any well-formed r^--derivation, and which is used to establish the ∂-equivalence.

The above facts lead to the following conclusions:

(c1) each intermediate word in any r^--derivation is of the form $u\triangleleft$ or $u\triangleright$, for some $u \in V^*$, $\triangleleft \in V_\triangleleft$ and $\triangleright \in V_\triangleright$;

(c2) each intermediate word $u\theta$ can be modified in one derivation step with one of the next three variants:

- if θ is a $\triangleleft$-symbol, then delete the last symbol of u (and move the $\triangleleft$-symbol one position to the left – the $\triangleleft$-symbol may be changed into a new $\triangleleft$-symbol);
- change the $\triangleleft$-symbol into a $\triangleright$-symbol, and vice versa;
- if θ is a $\triangleright$-symbol, then append one symbol from V to u (and move the $\triangleright$-symbol one position to the right – the $\triangleright$-symbol may be changed into a new $\triangleright$-symbol).

We say that an SE-system like G' defined above is in the *canonical form.* Therefore, any SE-system is equivalent to an SE-system in the canonical form.

If an SE-system in the canonical form has the additional property that L_1 is finite, then we say that it is in the *strong canonical form.* It is easily seen that such systems can be transformed into equivalent systems where $|L_1| = 1$ (by adding a finite number of words to the sets L_2 and S). From now on we will assume that any SE-system in the strong canonical form has this property. Moreover, we may assume that

$$L_1 = \begin{cases} \{u\triangleleft\}, & \text{if } u \neq \lambda \\ \{\triangleright\}, & \text{otherwise.} \end{cases}$$

Example 3.1 *The SE-system in Example 2.1.2 is in the strong canonical form; it is $\cap$-equivalent to a right-linear grammar. We can add a new symbol, $\triangleleft$, and transform each word $Aa \in L_2$, where $A \to a \in P$, into the word $Aa\triangleleft$, and each word $A \in L_2$, where $A \to \lambda \in P$, into the word $A\triangleleft$. Then, the system obtained is ∂-equivalent to a right-linear grammar.*

Lemma 3.1 *Any SE-system of type (reg, reg, f) is ∂-equivalent to an SE-system in the strong canonical form of type (f, f, f).*

Proof Let $G = (V, L_1, L_2, S)$ be an SE-system of type (reg, reg, f), and $G_1 = (V_N^1, V_T^1, X_0^1, P_1)$ and $G_2 = (V_N^2, V_T^2, X_0^2, P_2)$ be right-linear grammars generating the languages L_1 and L_2, respectively. We may assume that $V_T^1 \cup V_T^2 \subseteq V$ and $V \cap (V_N^1 \cup V_N^2) = \emptyset = V_N^1 \cap V_N^2$.

Define the SE-system $G' = (V \cup V_\triangleleft \cup V_\triangleright, L_1', L_2', S')$ as follows:

- $V_\triangleleft = \{\triangleleft_{s'} | \exists s \in S : \ s' \leq_{suff} s\}$, and
  $V_\triangleright = V_N^1 \cup V_N^2 \cup \{A_{s'} | A \in V_N^2, \exists s \in S : \ s' \leq_{suff} s\}$;
- $S' = \{a \triangleleft_{s'} | a \in V, \exists s \in S : \ as' \leq_{suff} s\} \cup$
  $\{\triangleleft_s | s \in S\} \cup$
  $\{A_{as'} | a \in V, A \in V_N^2, \exists s \in S : \ as' \leq_{suff} s\} \cup$
  $\{A_\lambda | A \in V_N^2\} \cup$
  $V_N^1 \cup V_N^2$;
- $L_1' = \{X_0^1\}$;
- $L_2' = \{AaB | A \to aB \in P_1\} \cup$
  $\{Aa \triangleleft_\lambda | A \to a \in P_1\} \cup$
  $\{a \triangleleft_{s'} \triangleleft_{as'} | a\triangleleft_{s'} \in S'\} \cup$
  $\{\triangleleft_s X_{0s}^2 | s \in S\} \cup$
  $\{A_{as'} B_{s'} | A \to aB \in P_2\} \cup$

$$\{B_\lambda B | B \in V_N^2\} \cup$$
$$\{AaB | A \to aB \in P_2\} \cup$$
$$\{Aa \lhd_\lambda | A \to a \in P_2\}.$$

G' is in the strong canonical form and it is of type (f, f, f). Moreover, $L^{r^-}(G) = \partial^r_{\lhd_\lambda}(L^{r^-}(G'))$. $\Box$

We will establish a connection between SE-systems and *pushdown automata* (*pda*, for short). For the definition of a pda, we follow [2]. That is, a *pda over an alphabet* V is a 5-tuple $\mathcal{A} = (Q, Z, i, K, T)$, where Q is the set of *states*, Z is the *stack alphabet*, $i \in Q \times Z^*$ is the *initial internal configuration*, $K \subseteq Q \times Z^*$ is a set of *accepting internal configurations*, and T is a subset of $(V \cup \{\lambda\}) \times Q \times Z \times Z^* \times Q$ (each element in T being called a *transition rule*).

The elements of $Q \times Z^*$ $(V^* \times Q \times Z^*)$ are called *internal configurations* (*configurations*) of $\mathcal{A}$. The set of internal accepting configurations are of the form $K = F \times Z^*$, where F is a subset of Q, called the *set of accepting states*. The *transition relation* over configurations, induced by $\mathcal{A}$, is defined by:

$$(ax, q, wz) \to (x, q', w\alpha) \quad \Leftrightarrow \quad (a, q, z, \alpha, q') \in T.$$

It is also convenient to denote

$$(q, w) \xrightarrow{x} (q', w')$$

instead of

$$(x, q, w) \xrightarrow{*} (\lambda, q', w').$$

The *stack language of* $\mathcal{A}$ is defined to be the language

$$Stack(\mathcal{A}) = \{w \in Z^* | \exists x, y \in V^*, \exists q \in Q, \exists k \in K : \ i \xrightarrow{x} (q, w) \xrightarrow{y} k\}.$$

Lemma 3.2 *For any SE-system G in the strong canonical form and of type (f, f, f), a pda $\mathcal{A}_G$ can be effectively constructed such that*

$$\partial^r_\lhd(L^{r^-}(G)) = \partial^l_{z_0}(Stack(\mathcal{A}_G)),$$

for some symbols $\lhd$ and z_0.

Proof Let $G = (V \cup V_\lhd \cup V_\rhd, \{X_0\}, L_2, S)$ be an SE-system in the strong canonical form and of type (f, f, f), where

- $L_2 \subseteq VV_\lhd V_\lhd \cup V_\rhd VV_\rhd \cup V_\lhd V_\rhd \cup V_\rhd V_\lhd$;
- $S \subseteq V_\lhd \cup V_\rhd \cup VV_\lhd$;
- $X_0 \in V_\rhd$.

Assume that $\lhd \in V_{\lhd}$ is the "closing" symbol of G. Define a pda $\mathcal{A} = (Q, Z, i, K, T)$ over an alphabet with one symbol x as follows:

- $Q = V_{\lhd} \cup V_{\rhd}$;
- $Z = V \cup \{z_0\}$, where z_0 is a new symbol;
- $i = (X_0, z_0)$;
- $K = \{\lhd\} \times Z^*$;
- T contains the following groups of rules:
  - $(x, \lhd_1, a, \lambda, \lhd_2)$, for any $a \lhd_1 \lhd_2 \in L_2$;
  - $(x, \rhd_1, z, za, \lhd_2)$, for any $z \in Z$ and any $\rhd_1 a \rhd_2 \in L_2$;
  - $(x, \lhd, z, z, \rhd)$, for any $z \in Z$ and $\lhd\rhd \in L_2$;
  - $(x, \rhd, z, z, \lhd)$, for any $z \in Z$ and $\rhd\lhd \in L_2$.

It is not difficult to see that this pda fulfils the lemma. $\Box$

The *rewriting problem* for SE-systems is the following:
Instance: An SE-system G in the strong canonical form and $\theta_1, \theta_2 \in V_{\lhd} \cup V_{\rhd}$;
Question: $\theta_1 \Rightarrow^*_{G,r^-} \theta_2$?

Theorem 3.1 *The rewriting problem for SE-systems in a strong canonical form and of type (f, f, f) is decidable.*

Proof Modify the pda in the proof of Lemma 3.2 as follows:

- $i = (\theta_1, z_0)$;
- $K = \{\theta_2\} \times \{\lambda\}$.

Then, it is easy to see that

$$\theta_1 \Rightarrow^*_{G,r^-} \theta_2 \quad \Leftrightarrow \quad L(\mathcal{A}_G) \neq \emptyset.$$

As the emptiness problem for context-free languages is decidable, we conclude that the rewriting problem for SE-systems in the strong canonical form and of type (f, f, f) is decidable, too. $\Box$

Theorem 3.2 *For every SE-system G in the strong canonical form and of type (f, f, f), the language $\partial^r_{\lhd}(L^{r^-}(G))$ is regular, where $\lhd$ is the closing symbol of G.*

Proof Let G be as in the proof of Lemma 3.2. For any word $w = a_1 \cdots a_n \in \partial^r_{\lhd}(L^{r^-}(G))$ there is an r^--derivation

$$X_0 \Rightarrow^*_{r^-} w \lhd .$$

Decompose this r^--derivation into steps such that, at the last of each of these steps, one letter or w is definitely set on its position (that is, no further derivation steps are applied):

$$X_0 \Rightarrow^*_{r^-} \rhd_1 \Rightarrow_{r^-} a_1\theta_1 \Rightarrow^*_{r^-} a_1\rhd_2 \Rightarrow_{r^-} a_1a_2\theta_2 \Rightarrow^*_{r^-}$$

$$\cdots \Rightarrow^*_{r^-} a_1 \cdots a_{n-1}\rhd_n \Rightarrow_{r^-} a_1 \cdots a_n\theta_n \Rightarrow_{r^-} a_1 \cdots a_n \lhd .$$

Now, each transitive sequence of steps (marked by "$*$") will be condensed just into a single derivation step. More precisely, we define the right-linear grammar $G' = (V_N, V_T, X_0, P)$ as follows:

- $V_T = V$, $V_N = V_{\lhd} \cup V_{\rhd}$;
- P contains the following rules:
  - $\rhd_1 \to \rhd_2$, for all $\rhd_1, \rhd_2 \in V_N$ such that $\rhd_1 \Rightarrow^*_{r^-} \rhd_2$; these rules can be effectively constructed because the rewriting problem is decidable for such SE-systems;
  - $\rhd_1 \to a\rhd_2$, for all $\rhd_1 a \rhd_2 \in L_2$;
  - $\rhd \to \lambda$, for all $\rhd\lhd \in L_2$.

It is clear that $L(G') = \partial^r_{\lhd}(L^{r^-}(G))$, which proves the theorem. □

Corollary 3.1 *For any SE-system G of type (reg, reg, f), the language $L^{r^-}(G)$ is regular.*

Proof From Lemma 3.1, Theorem 3.2 and the fact that the family of regular languages is closed under ∂_a-operation. □

Corollary 3.2 *For any SE-system G of type (reg, reg, f), the language $L^r(G)$ is regular.*

Proof From Remark 3.2(6) and Corollary 3.1. □

Corollary 3.3 *For any pda $\mathcal{A}$, Stack$(\mathcal{A})$ is regular.*

Proof For any pda $\mathcal{A}$ we can effectively construct a ∂-equivalent SE-system of type (f, f, f) to generate its stack language. Then, the result follows from Corollary 3.2. □

4 Ambiguous and Non-returning SE-systems

Definition 4.1 *An SE-system G is called x-*ambiguous*, where x is a mode of synchronization, if there is a word v having at least two distinct x-derivations in G.*

If an SE-system is not x-ambiguous then we will say that it is *x-nonambiguous.*

Remark 4.1 *The l^--ambiguity problem is undecidable because context-free grammars can be simulated step by step by SE-systems under the l^- mode of synchronization (Example 2.1.1), and the ambiguity problem for such grammars is undecidable ([9]).*

In follows from Remark 3.1 that the problem of r^--ambiguity is undecidable too (and, clearly, any problem of x-ambiguity, where x includes l^- or r^-).

Remark 3.2(6) shows us how to express L^r by means of L^{r^-}. We will consider now the converse of this.

Definition 4.2 *An* SE-system $G = (V, L_1, L_2, S)$ *is said to be* non-returning *if the following property holds:*

$$(\forall s_1 \in S)(\forall v \in L_2)(v = s_1 v' \quad \Rightarrow \quad (\forall s_2 \in S)(v' \not<_{suf} s_2)).$$

An SE-system which is not non-returning is called *returning.*

For an alphabet V let $\overline{V} = \{\overline{a} | a \in V\}$ be a copy of V ($V \cap \overline{V} = \emptyset$). Then, for $u = a_1 \dots a_n \in V^*$ denote by $\overline{u}$ the word $\overline{a}_1 \dots \overline{a}_n$.

If an SE-system G is non-returning, then any derivation $u \overset{*}{\Rightarrow}_{r-} v$ is of the form

$$u = u_1' s_1 \Rightarrow u_1' v_1' = u_1' v_1'' s_2 \Rightarrow u_1' v_1'' v_2' = u_1' v_1'' v_2'' s_3 \Rightarrow \cdots$$
$$u_1' v_1'' \dots v_{k-2}'' v_{k-1}' = u_1' v_1'' \dots v_{k-2}'' v_{k-1}'' s_k \Rightarrow u_1' v_1'' \dots v_{k-2}'' v_{k-1}'' v_k' = v,$$

where $s_i v_i' \in L_2$ and $v_j' = v_j'' s_{j+1}$ for any $1 \le i \le k$ and $1 \le j \le k-1$. This shows us that the word v is obtaining from u_1' by the catenation of the words $v_1'', \dots, v_{k-1}'', v_k'$. Let $v' = u_1' \overline{s}_1 v_1'' \overline{s}_2 v_2'' \cdots v_{k-1}'' \overline{s}_k v_k'$, $u' = u_1' \overline{s}_1$, and

- $L_1' = \{u' \overline{s} | u' s \in L_1,\ s \in S\}$,
- $L_2' = \{\overline{s}_1 v'' \overline{s}_2 | s_1 v'' s_2 \in L_2,\ s_1, s_2 \in S\} \cup \{\overline{s}_1 v' | s_1 v' \in L_2,\ s_1 \in S\}$,
- $\overline{S} = \{\overline{s} | s \in S\}$.

Then, it is easy to see that $u' \Rightarrow_{G',r} v'$, where $G' = (V \cup \overline{V}, L_1', L_2', \overline{S})$, and if we consider the homomorphism $h : (V \cup \overline{V})^* \to V^*$ defined by $h(x) = x$ and $h(\overline{x}) = \lambda$, for any $x \in V$, we get $h(u') = u$ and $h(v') = v$. Therefore, we have:

Theorem 4.1 *For any non-returning SE-system $G = (V, L_1, L_2, S)$ we have $L^{r-} = h(L^r(G'))$, where:*

- $G' = (V', L'_1, L'_2, S')$,
- $V' = V \cup \overline{V}$,
- $L'_1 = L_1 \cup (\bigcup_{s \in S}(\partial^r_s(L_1))\{\overline{s}\})$,
- $L'_2 = (\bigcup_{s_1,s_2 \in S}\{\overline{s}_1\}\partial^l_{s_1}(\partial^r_{s_2}(L_2))\{\overline{s}_2\}) \cup (\bigcup_{s \in S}\{\overline{s}\}(\partial^l_s(L_2)))$,
- $S' = \overline{S}$,
- $h(x) = x$ *and* $h(\overline{x}) = \lambda$ *for all* $x \in V$.

Theorem 4.2 *It is decidable whether or not a given non-returning SE-system of type* (f, f, f) *is* r^-*-ambiguous.*

Proof Let $G = (V, L_1, L_2, S)$ be a non-returning SE-system. We define a sequence of sets, $C_1, C_2, \ldots$, each of which consists of triples $(\theta_1, \theta_2, \theta_3) \in V^* \times (S \cup \{-\}) \times (S \cup \{-\})$, as follows (the symbol "$-$" is a new one):

1. C_1 contains all the triples of the form:
   (a) (x, s_1, s_2), where $x \in V^*$, $s_1, s_2 \in S$, and there are $u_1, u_2 \in L_1$ and $\alpha \in V^*$ such that $u_1 = \alpha x s_1$ and $u_2 = \alpha s_2$;
   (b) $(x, -, s_2)$, where $x \in V^*$, $s_2 \in S$, and there are $u_1, u_2 \in L_1$ and $\alpha \in V^*$ such that $u_1 = \alpha x$, $u_1 \notin V^*S$ and $u_2 = \alpha s_2$;
   (c) (x, λ, λ), where $x \in V^*$ and there are $u_1, u_2 \in L_1$ such that $u_1 = u_2 x$;
2. Suppose that the set C_i, $i \geq 1$, has been already defined and it contains triples only of the form (x, s_1, s_2) or $(x, -, s_2)$ or $(x, -, -)$, where $x \in V^*$ and $s_1, s_2 \in S$. Then, C_{i+1} is the set of all triples of the form:
   (a) (y, s_3, s_1), if there are $(x, s_1, s_2) \in C_i$ and a sequence
   $$s_2 v_1 s_{i_1}, \ldots, s_{i_{k-1}} v_k s_3 \in L_2,$$
   with $s_{i_1}, \ldots, s_{i_{k-1}} \in S$, such that $xy = v_1 \cdots v_k$ and $x \not\leq_{pref} v_1 \cdots v_{k-1}$;
   (b) $(y, -, s_1)$, if there are $(x, s_1, s_2) \in C_i$ and a sequence
   $$s_2 v_1 s_{i_1}, \ldots, s_{i_{k-1}} v_k \in L_2,$$
   with $s_{i_1}, \ldots, s_{i_{k-1}} \in S$ and $v_k \notin V^*S$, such that $xy = v_1 \cdots v_k$ and $x \not\leq_{pref} v_1 \cdots v_{k-1}$;
   (c) $(y, s_3, -)$, if there are $(x, -, s_2) \in C_i$ and a sequence as in (a);
   (d) $(y, -, -)$, if there are $(x, -, s_2) \in C_i$ and a sequence as in (b);

The sets defined above are finite and each of them can be effectively constructed. Moreover, there are $i < j$ $(i, j \geq 1)$ such that $C_j = C_i$. Therefore, $\bigcup_{i \geq 1} C_i$ is finite.

We construct a graph $\mathcal{G}$ which has a node for each element in the set $\bigcup_{k \geq 1} C_i$ and whose arcs are of the form $((x, s_1, s_2), (y, s_3, s_1))$, where $(x, s_1, s_2) \in C_i$ for some i and (y, s_3, s_1) is obtained from (x, s_1, s_2) as described above. The nodes of the form $(\lambda, \theta_2, \theta_3)$ with $\theta_2 = \theta_3$ or $\theta_2, \theta_3 \in \{\lambda, -\}$ are called *terminal nodes*. Otherwise, a node is a *nonterminal node*.

Then, it is easy to see that G is r^--ambiguous iff there is a path in $\mathcal{G}$ from a node in C_1 to a terminal node, containing at least one nonterminal node. $\Box$

Remark 4.2 *One can easily prove, using a similar construction as that in the proof of Theorem 4.1, that the l^--ambiguity problem is decidable for non-returning SE-systems of type (f, f, f).*

5 SE-systems and Codes

An x-derivation $u_1 \Rightarrow_x u_2 \Rightarrow_x \cdots \Rightarrow_x u_n$ is called *reduced* if it does not contain cycles, that is, there are no i and j such that $i \neq j$ and $u_i = u_j$. Clearly, any x-derivation can be reduced in different ways. For example, the x-derivation

$$u_1 \Rightarrow_x u_2 \Rightarrow_x u_3 \Rightarrow_x u_1 \Rightarrow_x u_4 \Rightarrow_x u_5 \Rightarrow_x u_3,$$

where $u_1, \ldots, u_5$ are assumed pairwise distinct, can be reduced either to

$$u_1 \Rightarrow_x u_4 \Rightarrow_x u_5 \Rightarrow_x u_3$$

or to

$$u_1 \Rightarrow_x u_2 \Rightarrow_x u_3.$$

If an SE-system has the property that for any word v there is at most a reduced x-derivation of v, then it is called *weak x-nonambiguous*. It is clear that in a weak x-nonambiguous SE-system G there can exist words v with more than two x-derivations. But, in this case, all these x-derivations can be reduced, by removing cycles, to a unique reduced x-derivation.

If a system G is x-nonambiguous, then it is also weak x-nonambiguous, but a weak x-nonambiguous system is not necessarily x-nonambiguous. It is possible to decide whether or not a non-returning SE-system of type (f, f, f) is weak r^--ambiguous by using the graph given in the proof of Theorem 4.1.

In what follows we will make a connection between codes and weak nonambiguous SE-systems. Recall first the concept of a *code* ([14]). A *code* (over an alphabet V) is a couple (V, C) between an alphabet V and a subset $C \subseteq V^+$, such that:

$$(\forall u_1, \ldots, u_m, v_1, \ldots, v_n \in C)(u_1 \cdots u_m = v_1 \cdots v_n \quad \Rightarrow \quad u_1 = v_1).$$

We can easily characterize codes by weak nonambiguous SE-systems, as follows.

Proposition 5.1 *A set $C \subseteq V^+$ is a code over V iff the SE-system $(V, C, C, \{\lambda\})$ is weak r-nonambiguous (or r^--nonambiguous).*

Recall now the concept of a *z-code* ([1], [10]). Let V be an alphabet, $X \subseteq V^+$, and let T_X be the set

$$T_X = \{((ux, v), (u, xv)) | u, v \in V^*, \ x \in X\} \subseteq (V^* \times V^*)^2.$$

We say that the pair of words (u, v) *produces in one step* the pair (u', v'), denoted by $(u, v) \rightarrow_X (u', v')$, if $((u, v), (u', v')) \in T_X$ or $((u', v'), (u, v)) \in T_X$.

A *z-factorization over X* of a word $w \in V^+$ is a sequence of derivation steps

$$(u_1, v_1) \rightarrow_X \cdots \rightarrow_X (u_m, v_m)$$

such that :

(i) $u_1 = v_m = \lambda$,

(ii) $v_1 = u_m = w$,

(iii) $(u_j, v_j) \neq (u_k, v_k), \forall j \neq k$.

A couple (V, X) between an alphabet V and a subset $X \subseteq V^+$ is called a *z-code* (over V) if any word $w \in V^+$ has at most one z-factorization over X.

Proposition 5.2 *Let V be an alphabet, $X \subseteq V^+$, and let # be a new symbol. Then, (V, X) is a z-*code *iff the SE-system*

$$G = (V \cup \{\#\}, X\{\#\}, \{\#\}X\{\#\} \cup X\{\#\#\}, \{\#\} \cup X\{\#\})$$

is weak r^--nonambiguous.

Proof Let G be the SE-system from proposition. It is easy to see that, for any $u, v \in V^*$ and $x \in X$, the following hold:

(a) $(u, xv) \rightarrow_X (ux, v)$ iff $u\# \Rightarrow_{G,r^-} ux\#$;

(b) $(ux, v) \rightarrow_X (u, xv)$ iff $ux\# \Rightarrow_{G,r^-} u\#$.

From these two facts we get the proposition. □

The characterization of (z-)codes by weak nonambiguous SE-systems leads us to adopt the terminology of an *x-code*, where x is a mode of synchronization, for any weak x-nonambiguous SE-system. Under this assumption, (z-)codes are particular cases of r^--codes.

6 A Generalization and Splicing Systems

The concept of an SE-system can be generalized in a natural way as follows.

Definition 6.1 *A* generalized synchronized extension system *(GSE-system, for short) is a 4-tuple $G = (V, L_1, L_2, S)$, where V is an alphabet, L_1 and L_2 are languages over V, and S a ternary relation on V^*.*

For the components of a GSE-system we use the same terminology as for SE-systems.

Let G be a GSE-system. G induces two derivation relations, $\Rightarrow_{G,r}$ and $\Rightarrow_{G,l}$, which are defined very similarly to the case of SE-systems. For example,

$$u \Rightarrow_{G,r} v \;\Leftrightarrow\; (\exists w \in L_2)(\exists (s_1, s_2, s_3) \in S)(u = xs_1 \;\wedge\; w = s_2y \;\wedge\; v = xs_3y).$$

Notice the roles of s_1, s_2, and s_3 in triples $(s_1, s_2, s_3) \in S$: s_1 and s_2 are used in order that u and v synchronize each other, whereas s_3 is the result of the synchronization.

We can easily see that SE-systems (and their derivation relations) are particular cases of GSE-systems: S has triples of the form (s, s, s) or (s, s, λ).

We will discuss now the "power" of GSE-system by comparing them with splicing schemes.

A *splicing rule* over an alphabet V is a 4-tuple (u_1, u_2, u_3, u_4) of words over V ([7], [8]). A *splicing scheme* is a pair $\sigma = (V, R)$ consisting of an alphabet V and a set R of splicing rules over V. The splicing scheme σ induces a (partially) binary operation on V^*, denoted also by σ, as follows:

$$\begin{aligned}\sigma(x, y) = \{x_1u_1u_4y_2 \quad | \quad & \exists (u_1, u_2, u_3, u_4) \in R : \; x = x_1u_1u_2x_2, \\ & y = y_1u_3u_4y_2, \; z = x_1u_1u_4y_2\},\end{aligned}$$

for any $x, y \in V^*$. This operation can be extended to languages over V^* as usual, and then can be iterated as follows:

- $\sigma^0(L_1, L_2) = L_1$,
- $\sigma^{i+1}(L_1, L_2) = \sigma(\sigma^i(L_1, L_2), L_2)$, $\forall i \geq 0$,

for all languages L_1 and L_2. Define then $\sigma^*(L_1, L_2) = \bigcup_{i \geq 0} \sigma^i(L_1, L_2)$.

Let $\sigma = (V, R)$ be a splicing scheme and L_1, L_2 be two languages over V. Define the set

$$S = \{(u_1u_2\alpha, \beta u_3u_4, u_1u_4) | (u_1, u_2, u_3, u_4) \in R, \; \alpha, \beta \in V^*\}$$

and consider the system $G_\sigma = (V, L_1, L_2, S)$. It is clear that we have

$$\sigma^*(L_1, L_2) = L^r(G_\sigma).$$

References

[1] Anselmo,M.: Sur les codes zigzag et leur décidabilité. Theoret. Comput. Sci. 74, 341–354 (1990)

[2] Autebert,J.-M., Berstel,J., Boasson,L.: Context-Free Languages and Pushdown Automata. In: Rozenberg, G., Salomaa, A. (eds.), Handbook of Formal Languages, vol. 1. Berlin Heidelberg New York: Springer, 1997, pp. 111–174

[3] Bucher,W., Hagauer,J.: It is decidable whether a regular language is pure context-free. Theoret. Comput. Sci. 26, 233–241 (1983)

[4] Büchi,J.R.: Regular canonical systems. Arch. Math. Logik Grundlagenforsch. 6, 91–111 (1964)

[5] Dassow,J., Păun,Gh.: Regulated Rewriting in Formal Language Theory. Berlin Heidelberg New York: Springer, 1989

[6] Gabrielian,A.: Pure grammars and pure languages. Intern. J. Computer Math. 9, 3–16 (1981)

[7] Head,T.: Formal Language Theory and DNA: an Analysis of the Generative Capacity of Specific Recombinant Behaviors. Bull. Math. Biology 49, 737–759 (1987)

[8] Head,T., Păun,Gh., Pixton,D.: Language Theory and Molecular Genetics. In: Rozenberg, G., Salomaa, A. (eds.), Handbook of Formal Languages, vol. 2. Berlin Heidelberg New York: Springer, 1997, pp. 295–360

[9] Hopcroft,J.E, Ullman,J.D.: Introduction to Automata Theory, Languages, and Computation. Reading, MA: Addison-Wesley, 1979

[10] Madonia,M., Salemi,S., Sportelli,T.: A Generalization of Sardinas and Patterson's Algorithm to z-codes. Theoret. Comput. Sci. 108, 251–270 (1993)

[11] Maurer,H.A., Salomaa,A., Wood,D.: Pure grammars. Inform. Contr. 44, 47–72 (1980)

[12] Păun, Gh.: On the Generative Capacity of Conditional Grammars. Inform. Contr. 43, 178–186 (1979)

[13] Rozenberg,G., Salomaa,A., (eds.): Handbook of Formal Languages, vol. 1. Berlin Heidelberg New York: Springer, 1997

[14] Salomaa,A.: Jewels of Formal Language Theory. Rockville, MD: Computer Science Press, 1981