ON SE-SYSTEMS AND MONADIC STRING REWRITING SYSTEMS

F.L. ŢIPLEA AND ERKKI MÄKINEN

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES
UNIVERSITY OF TAMPERE

REPORT A-2000-15

UNIVERSITY OF TAMPERE
DEPARTMENT OF COMPUTER AND
INFORMATION SCIENCES
SERIES OF PUBLICATIONS A
A-2000-15, NOVEMBER 2000

ON SE-SYSTEMS AND MONADIC STRING REWRITING SYSTEMS

F.L. ŢIPLEA AND ERKKI MÄKINEN

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

ISBN 951-44-4987-8
ISSN 1457-2060

On SE-Systems and Monadic String Rewriting Systems

Ferucio Laurenţiu Ţiplea [1]

Faculty of Computer Science, "Al.I.Cuza" University of Iasi, 6600 Iasi, Romania

Erkki Mäkinen [2]

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

Abstract

A *synchronized extension system* is a powerful and elegant rewriting formalism. In this paper we show how it can be used to improve a well-known result concerning monadic string rewriting systems. We give a new proof for the fact that if the rewriting rules of a monadic string rewriting system are applied to the strings of a regular set L, the set so obtained is also regular. We obtain the result with better time and space bounds than the earlier proofs.

Keywords: formal languages, monadic string rewriting systems, synchronized extension system.

1 Introduction and Preliminaries

A synchronized extension system [8,6,7] is a new powerful and elegant rewriting formalism which has proved to be useful in various kinds of problems in formal language theory [8], and especially in "pure" systems [7]. In this paper we show how synchronized extension systems can be used to improve a well-known result concerning monadic string rewriting systems.

In [2] (see also [1]), Book and Otto show, among many other results, that if the rewriting rules of a monadic string rewriting system are applied to the strings of a regular set L, the set so obtained (the set of descendants of L) is also regular (see [3] for historical remarks and related problems). Book and Otto ([2]) prove the result by transforming the finite automaton accepting

[1] E-mail:fltiplea@infoiasi.ro

[2] E-mail: em@cs.uta.fi. Work supported by the Academy of Finland (Project 35025).

L to the automaton accepting the set of descendants. The time and space complexities of the original algorithm by Book and Otto are later improved ([4,3,5]). Esparza et al. ([5]) prove $\mathcal{O}(ps^3)$ time and $\mathcal{O}(ps^2)$ space bounds where p is the number of rules in the monadic string rewriting system and s is the number of states in the automaton accepting L. We show that synchronized extension systems provide a new insight to the problem and allow a $\mathcal{O}(pr)$ time and space solution, where p is as above and r is the number of the rules in the grammar generating L.

1.1 String Rewriting Systems

We start by recalling some concepts from [2] related to string rewriting systems. A *string rewriting system over an alphabet* Σ (shortly, an *STS over* Σ) is an non-empty subset $T \subseteq \Sigma^* \times \Sigma^*$. Each element $(\alpha, \beta) \in T$ is called a *(rewrite) rule*; they are usually denoted by $\alpha \to \beta$. The *rewriting* (or *step derivation*) *relation induced by* T is the binary relation $\Rightarrow_T$ on Σ^* given by

$$u \Rightarrow_T v \quad \Leftrightarrow \quad u = u_1 \alpha u_2 \ \wedge \ v = u_1 \beta u_2 \ \wedge \ \alpha \to \beta \in T,$$

for all $u, v \in \Sigma^*$. The reflexive and transitive closure of $\Rightarrow_T$, denoted by $\stackrel{*}{\Rightarrow}_T$, is called the *derivation relation induced by* T.

A *(non-empty) derivation of* u *into* v by T is a sequence of step derivations

$$u = u'_0 \alpha_1 u''_0 \Rightarrow_T u'_0 \beta_1 u''_0 = u_1 = u'_1 \alpha_2 u''_1 \Rightarrow_T \cdots \Rightarrow_T u'_{n-1} \beta_n u''_{n-1} = u_n = v,$$

where $n \geq 1$ and $r_i : \alpha_i \to \beta_i \in T$ for all $1 \leq i \leq n$. Sometimes we will write $u \xRightarrow{r_1 \cdots r_n}_T v$ to denote the fact that u is rewritten into v by the rules $r_1, \ldots, r_n$ used in this order (this notation is ambiguous because it does not take into consideration the places where the rules are applied. However, we will use it in conjunction with a derivation explicitly given as above in order to simplify the notation and, therefore, the ambiguities will be avoided.)

For a language L over an alphabet Σ and an STS T over Σ, we denote by $\Delta_T^*(L)$ the language

$$\Delta_T^*(L) = \{v \in \Sigma^* | \exists u \in L : \ u \stackrel{*}{\Rightarrow}_T v\}.$$

A *monadic STS* (over an alphabet Σ) is an STS T having the property $|\alpha| > |\beta|$ and $|\beta| \leq 1$, for all $\alpha \to \beta \in T$.

In [2] it has been proved that for any non-deterministic finite automaton A and monadic STS T one can construct in polynomial time a non-deterministic finite automaton B such that $L(B) = \Delta_T^*(L(A))$. In this paper, we give a new proof for this result.

1.2 Synchronized Extension Systems

Synchronized extension systems (SE-systems, for short) are introduced in [8] as 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. L_1 is called the *initial language*, L_2 the *extending language*, and S the *synchronization set* of G. For an SE-system G, define the binary relations $\Rightarrow_{G,r}$, $\Rightarrow_{G,r^-}$, $\Rightarrow_{G,l}$ and $\Rightarrow_{G,l^-}$ over V^* as follows:

- $u \Rightarrow_{G,r} v$ iff $(\exists w \in L_2)(\exists s \in S)(\exists x, y \in V^*)(u = xs \ \wedge \ w = sy \ \wedge \ v = xsy)$;
- $u \Rightarrow_{G,r^-} v$ iff $(\exists w \in L_2)(\exists s \in S)(\exists x, y \in V^*)(u = xs \ \wedge \ w = sy \ \wedge \ v = xy)$;
- $u \Rightarrow_{G,l} v$ iff $(\exists w \in L_2)(\exists s \in S)(\exists x, y \in V^*)(u = sx \ \wedge \ w = ys \ \wedge \ v = ysx)$;
- $u \Rightarrow_{G,l^-} v$ 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, and r, r^-, l, and l^- are called (basic) *modes of synchronizations.* In what follows, we restrict ourselves to the mode r^-.

We say that an SE-system $G = (V, L_1, L_2, S)$ is *of type* (p_1, p_2, p_3) if the L_1, L_2, and S are languages having the properties p_1, p_2, and p_3, respectively. We use the abbreviations f and *reg* for the properties of finiteness and regularity, respectively.

A derivation $u \stackrel{*}{\Rightarrow}_{r^-} v$ is called an r^-*-derivation.* The *language of type* r^- generated by an *SE-system* $G = (V, L_1, L_2, S)$ is defined as

$$L^{r^-}(G) = \{v \in V^* | \exists u \in L_1 : \ u \stackrel{*}{\Rightarrow}_{G,r^-} v\}$$

(naturally, the other modes of synchronization as well define their own classes of languages, but we do not need them here.)

The following result is essential for this note.

Theorem 1 ([8]) *For any SE-system G of type (reg, reg, f), the language $L^{r^-}(G)$ is regular.*

2 Left-to-Right Derivations in STS's

In this section we give technical results concerning the form of derivations in finite STS's. The concept of a "derivation from u on x within the decomposition $u = u_1xu_2$" is aimed to capture the idea that the subwords u_1 and u_2 are not used (neither partially nor totally) in a derivation. Alternatively, one can say that the derivation $u = u_1xu_2 \overset{*}{\Rightarrow}_T u_1yu_2$ from u on x is obtained from the (normal) derivation $x \overset{*}{\Rightarrow}_T y$ by catenating to each step the word u_1 to the left and the word u_2 to the right.

Definition 2 *Let T be a finite STS over an alphabet Σ, $u \in \Sigma^+$ and x a subword of u. A* derivation from u on x within a decomposition $u = u_1xu_2$ *is defined inductively as follows:*

– *if $u = u_1xu_2 = u_1x'\alpha x''u_2$ and $\alpha \to \beta \in T$, then*

$$u = u_1xu_2 = u_1x'\alpha x''u_2 \Rightarrow_T u_1x'\beta x''u_2$$

is a derivation on x;

– *if $u = u_1xu_2 \overset{*}{\Rightarrow}_T u_1yu_2$ is a derivation on x and*

$$u = u_1yu_2 = u_1y'\alpha y''u_2 \Rightarrow_T u_1y'\beta y''u_2$$

is a derivation on y (within the decomposition $u = u_1yu_2$), then

$$u = u_1xu_2 \overset{*}{\Rightarrow}_T u_1yu_2 = u_1y'\alpha y''u_2 \Rightarrow_T u_1y'\beta y''u_2$$

is a derivation on x.

Now we are ready to define left-to-right derivations of a finite STS.

Definition 3 *Let T be a finite STS over an alphabet Σ, $u \in \Sigma^+$, and let $\mathcal{D}$ the derivation*

$$\mathcal{D}: \quad u \xRightarrow{r_1\cdots r_{i-1}} u_1\alpha_iu_2 \overset{r_i}{\Rightarrow} u_1\beta_iu_2 \xRightarrow{r_{i+1}\cdots r_n} v.$$

Let $i < j \le n$.

(1) The step j of $\mathcal{D}$ is said to be to the left of the step i *if there is a decomposition $u_1 = u_1'u_1''$ of u_1 such that the derivation*

$$u_1\beta_iu_2 = u_1'u_1''\beta_iu_2 \xRightarrow{r_{i+1}\cdots r_{j-1}} xy$$

is on u_1' or $u_1''\beta_iu_2$ and the step j (of $\mathcal{D}$) is on x.

(2) The step j of $\mathcal{D}$ is said to be to the right of the step i *if there is a decomposition $u_2 = u_2'u_2''$ of u_2 such that the derivation*

$$u_1\beta_iu_2 = u_1\beta_iu_2'u_2'' \xRightarrow{r_{i+1}\cdots r_{j-1}} xy$$

is on $u_1\beta_i u_2'$ or u_2'' and the step j (of $\mathcal{D}$) is on y.

(3) The step j of the derivation $\mathcal{D}$ is said to be dependent on the step i *if it is neither to the left nor to the right of the step i.*

(4) The derivation $\mathcal{D}$ is called a left-to-right (right-to-left) derivation of u into v *if for every i, $1 \leq i < n$, the step $i+1$ is not to the left (right) of the step i.*

The following lemma states that it is sufficient to consider left-to-right derivations in finite STS's.

Lemma 4 *Let T be a finite STS over an alphabet Σ. Then, for every derivation $\mathcal{D}$ of a word u into a word v one can effectively construct a left-to-right derivation $\mathcal{D}'$ of u into v. Moreover, the derivation $\mathcal{D}'$ can be obtained by changing only the order of steps in the original derivation $\mathcal{D}$.*

Proof. Let $\mathcal{D}$ be a derivation of u into v,

$$\mathcal{D}: \quad u \overset{s}{\Rightarrow}_T v,$$

where $s = r_1 \ldots r_n \in T^+$.

Define inductively a sequence $s' = r_{i_1} \cdots r_{i_n}$, where $i_1, \ldots, i_n \in \{1, \ldots, n\}$ are pairwise distinct, as follows:

(i) Initially, set $s' := r_1$;

(ii) Assume that s' is the sequence obtained by rearranging the subsequence $r_1 \cdots r_k$ of s, where $k < n$;

(iii) Consider the rule r_{k+1} and the following possible cases:

- (a) r_{k+1} is to the left of all rules in s'. Then, define $s' = r_{k+1}s'$;
- (b) r_{k+1} does not depend on any rule in s'. Then, find the biggest j such that r_{k+1} is to the right of $s'(j)$ and insert r_{k+1} immediately after $s'(j)$ (that is, $s' := s'(1) \cdots s'(j) r_{k+1} s'(j+1) \cdots s'(p)$, where $|s'| = p$);
- (c) r_{k+1} depends on some rule in s', and let j be the biggest index such that r_{k+1} depends on $s'(j)$. Then, insert r_{k+1} immediately after $s'(j)$ (as above).

The fact that $\mathcal{D}'$ defined by s' is a left-to-right derivation follows directly from the construction above.

Example 5 *Let $T = \{p_1 : abb \to ab, p_2 : aba \to bb, p_3 : aa \to a, p_4 : bab \to bb\}$*

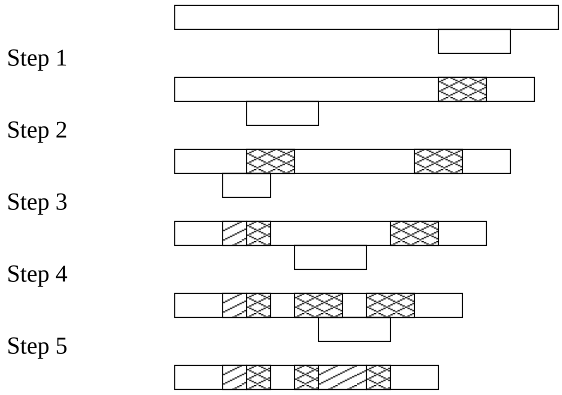

Fig. 1. The steps of the derivation $\mathcal{D}$

be an STS over $\Sigma = \{a, b\}$. *Consider the derivation*

$$
\begin{aligned}
\mathcal{D}: \quad u = abaabbaabaaababb &\overset{p_2}{\Rightarrow}_T abaabbaabaabbbb \\
&\overset{p_1}{\Rightarrow}_T abaabaabaabbbb \\
&\overset{p_3}{\Rightarrow}_T ababaabaabbbb \\
&\overset{p_2}{\Rightarrow}_T abababbabbbb \\
&\overset{p_4}{\Rightarrow}_T abababbbbbb = v
\end{aligned}
$$

whose steps are pictorially presented in Figure 5.

It is easy to see that

- *the step 2 is to the left of the step 1;*
- *the step 3 depends on step 2, and is to the left of the step 1;*
- *the step 4 is to the right of the steps 2 and 3, but to the left of the step 1;*
- *the step 5 is to the right of the steps 2 and 3, and depends on the steps 1 and 4.*

The algorithm in the proof of Lemma 4 outputs

$$
\begin{aligned}
s' &:= r_1 && (= p_2) \\
s' &:= r_2 r_1 && (= p_1 p_2) \\
s' &:= r_2 r_3 r_1 && (= p_1 p_3 p_2) \\
s' &:= r_2 r_3 r_4 r_1 && (= p_1 p_3 p_2 p_2) \\
s' &:= r_2 r_3 r_4 r_1 r_5 && (= p_1 p_3 p_2 p_2 p_4).
\end{aligned}
$$

The last value of s' *defines a left-to-right derivation* $\mathcal{D}'$ *of* u *into* v.

Remark 6 *In [2], Book and Otto introduce the concept of a* leftmost derivation *for STS's. Let* T *be an STS over an alphabet* Σ*. A derivation step* $u \Rightarrow_T v$ *is called a* leftmost derivation step *if the following hold:*

(i) there is a rule $\alpha \to \beta \in T$ *such that* $u = u_1\alpha u_2$ *and* $v = u_1\beta u_2$*;*
(ii) for every rule $\alpha' \to \beta' \in T$ *such that* $u = u_1'\alpha' u_2'$ *we have*
  - $u_1\alpha$ *is a proper prefix of* $u_1'\alpha'$*, or*
  - $u_1\alpha = u_1'\alpha'$ *and* u_1 *is a proper prefix of* u_1'*, or*
  - $u_1 = u_1'$ *and* $\alpha = \alpha'$*.*

A derivation is called leftmost *if each step of it is a leftmost derivation step.*

Every two consecutive leftmost derivation steps have the property that the latter one is not to the left of the first one (otherwise, (ii) is contradicted). Therefore, every leftmost derivation is a left-to-right derivation, but the converse does not hold (it is not difficult to construct a left-to-right derivation which is not a leftmost derivation of the STS from Example 5).

As a conclusion, derivations of STS are not generally equivalent to leftmost derivations.

3 The Case of Monadic STS

If the step j (using the rule $r_j : \alpha_j \to \beta_j$) of a derivation depends on a step i (using the rule $r_i : \alpha_i \to \beta_i$), then α_j uses, directly or indirectly, subwords of β_i. For instance, this is the case of the steps 3 and 5 in Example 5.

Left-to-right derivations of monadic STS have the interesting property that whenever a step j depends on a step i then it uses the all right hand side of the rule r_i. This property is crucial for the results to be proved in this section.

Let $G = (V_N, V_T, X_0, P)$ be a regular (right-linear) grammar without unit productions (i.e., rules $A \to B$ where $A, B \in V_N$), and let T be a finite monadic STS over V_T. We consider the SE-system $H = (V, L_1, L_2, S)$, where

- $V = V_N \cup V_T$,
- $L_1 = \{X_0\}$,
- $L_2 = \{A\beta | A \to \beta \in P\} \cup \{\alpha A \beta A | A \in V_N \ \wedge \ \alpha \to \beta \in T\}$,
- $S = V_N \cup \{\alpha A | A \in V_N \ \wedge \ (\exists \beta)(\alpha \to \beta \in T)\}$.

Then, H is an SE-system of type (f, f, f) and, from Theorem 1 it follows that $L^{r^-}(H)$ is regular. We will prove that $\Delta_T^*(L(G)) = L^{r^-}(H) \cap V_T^*$.

Theorem 7 *Let* G*,* T *and* H *be as above. Then,* $\Delta_T^*(L(G)) = L^{r^-}(H) \cap V_T^*$*.*

Proof. A derivation in G,

$$X_0 \Rightarrow_G a_1 A_1 \Rightarrow_G \cdots \Rightarrow_G a_1 \cdots a_{n-1} A_{n-1} \Rightarrow_G a_1 \cdots a_{n-1} a_n,$$

is simulated in H by synchronized extensions to the right, that is

$$X_0 \Rightarrow_{r-} a_1 A_1 \Rightarrow_{r-} \cdots \Rightarrow_{r-} a_1 \cdots a_{n-1} A_{n-1} \Rightarrow_{r-} a_1 \cdots a_{n-1} a_n$$

(for some variables $A_1, \ldots, A_{n-1}$).

The action of T on $u = a_1 \cdots a_n$ is simulated at the time of generating u. Assume that u is rewritten into v by the sequnce $s = r_1 \cdots r_m$ of rules of T, and let $\mathcal{D}$ be the derivation

$$\mathcal{D}: \quad X_0 \overset{*}{\Rightarrow}_G u \overset{s}{\Rightarrow}_T v.$$

By Lemma 4 we may assume that the derivation of u into v is left-to-right. Then, $\mathcal{D}$ can be simulated by a derivation in H using the following remarks:

(1) if $u = u_1 \alpha u_2 \alpha' u_3$ and $r : \alpha \to \beta, r' : \alpha' \to \beta' \in T$, then the derivation

$$X_0 \overset{*}{\Rightarrow}_G u = u_1 \alpha u_2 \alpha' u_3 \overset{rr'}{\Longrightarrow}_T u_1 \beta u_2 \beta' u_3$$

can be simulated in H by

$$\begin{aligned} X_0 &\overset{*}{\Rightarrow}_{r-} u_1 \alpha A \\ &\Rightarrow_{r-} u_1 \beta A \\ &\overset{*}{\Rightarrow}_{r-} u_1 \beta u_2 \alpha' B \\ &\Rightarrow_{r-} u_1 \beta u_2 \beta' B \\ &\overset{*}{\Rightarrow}_{r-} u_1 \beta u_2 \beta' u_3, \end{aligned}$$

for some variables A and B;

(2) if $u = u_1 u_2 \alpha u_3 u_4$, $r : \alpha \to \beta$, $\alpha' = u_2 \beta u_3$ and $r' : \alpha' \to \beta' \in T$, then the derivation

$$X_0 \overset{*}{\Rightarrow}_G u = u_1 u_2 \alpha u_3 u_4 \overset{r}{\Rightarrow}_T u_1 u_2 \beta u_3 u_4 \overset{r'}{\Rightarrow}_T u_1 \beta' u_4$$

can be simulated in H by

$$\begin{aligned} X_0 &\overset{*}{\Rightarrow}_{r-} u_1 u_2 \alpha A \\ &\Rightarrow_{r-} u_1 u_2 \beta A \\ &\overset{*}{\Rightarrow}_{r-} u_1 u_2 \beta u_3 B \\ &\Rightarrow_{r-} u_1 \beta' B \\ &\overset{*}{\Rightarrow}_{r-} u_1 \beta' u_4; \end{aligned}$$

(3) since the right hand side of each rule in T is either a symbol or the empty word, every two rules r and r' that are applied successively can be related either as in (1) or as in (2).

Therefore, every derivation in G from X_0 to a word $u \in V_T^*$ followed by a derivation in T from u into a word v can be simulated by a derivation in H from X_0 into v.

Conversely, it is trivial to see that every derivation in H leading to a word $v \in V_T^*$ is a combination between a derivation in G from X_0 into a word $u \in V_T^*$ followed then by a derivation in T from u into v.

As a conclusion, $\Delta_T^*(L(G)) = L^{r^-}(H) \cap V_T^*$.

Corollary 8 *For every right-linear grammar $G = (V_N, V_T, X_0, P)$ and every monadic STS T over V_N, the language $\Delta_T^*(L(G))$ is regular.*

As already mentioned, Esparza et al. ([5]) prove the bounds $\mathcal{O}(ps^3)$ and $\mathcal{O}(ps^2)$ for time and space, respectively, for the algorithm constructing the finite automaton accepting the descendants of a given regular set.

Consider now the complexity of our construction. To the extending language L_2 we take a string for each production in G, and a string for each production in G and a rule in T. Hence, the cardinality of L_2 is $\mathcal{O}(\mid P \mid \cdot \mid T \mid)$. To the synchronization set S we take a string for each nonterminal in G, and a string for each nonterminal in G and a left hand side of a rule in T. Thus, the sum of the cardinalities of L_2 and S is $\mathcal{O}(\mid P \mid \cdot \mid T \mid)$. This gives the time and space bounds for our construction.

Theorem 9 *Let G and T be as above. Then, an SE-system H of type (f, f, f) simulating the computation of $\Delta_T^*(L(G))$ can be constructed in $\mathcal{O}(\mid P \mid \cdot \mid T \mid)$ time and space.*

Theorem 9 gives the complexity of *constructing* H. Note that the complexity is dominated by the size of the output; the algorithm itself is straightforward. Implementing the computation defined by H is also possible to perform very efficiently. Namely, we can store V_N, V_T, S, and T in arrays, and maintain the current string in a stack. A simulation step simply rewrites the right hand side end of the current string.

4 Final Remarks

Bouajjani et al. ([3]) motivate their study from a pedagogical point of view: the algorithm constructing the finite automaton accepting the set of descendants can be used as a uniform basis for several independent algorithms for standard problems on context-free grammars. The pedagogical relevance is obvious also in the case of SE-systems. Namely, as shown above and in [6–8], SE-systems are notationally simple, but yet very powerful formalism which can simulate various other formalisms of theoretical computer science.

References

[1] R.V. Book, F. Otto, Cancellation rules and extended word problems. *Inform. Process. Lett.* 20 (1985), 5–11.

[2] R.V. Book, F. Otto, *String Rewriting Systems.* Springer-Verlag, 1993.

[3] A. Bouajjani, J. Esparza, A. Finkel, O. Maler, P. Rossmanith, B. Willems, P. Wolper, An efficient automata approach to some problems on context-free grammars. To appear in *Inform. Process. Lett.*

[4] J. Esparza, P. Rossmanith, An automata approach to some problems on context-free grammars. In *Foundations of Computer Science: Potential, Theory, Cognition*, Lecture Notes in Computer Science 1337, 1997, 143–152.

[5] J. Esparza, P. Rossmanith, S. Schwoon, A uniform framework for problems on context-free grammars. *Bull. EATCS* 72 (2000), 169–177.

[6] F.L. Ţiplea, E. Mäkinen, A Note on synchronized extension systems. To appear in *Inform. Process. Lett.*

[7] F.L. Ţiplea, E. Mäkinen, A note on SE-systems and regular canonical systems. Dept. of Computer and Information Sciences, University of Tampere, Tech. Report A-2000-14, October 2000. Submitted.

[8] F.L. Ţiplea, E. Mäkinen, C. Apachiţe. Synchronized extension systems. To appear in *Acta Inform.*