SE-SYSTEMS, TIMING MECHANISMS, AND TIME-VARYING CODES

F.L. ŢIPLEA, ERKKI MÄKINEN
AND CONSTANTIN ENEA

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES
UNIVERSITY OF TAMPERE

REPORT A-2000-17

UNIVERSITY OF TAMPERE
DEPARTMENT OF COMPUTER AND
INFORMATION SCIENCES
SERIES OF PUBLICATIONS A
A-2000-17, DECEMBER 2000

SE-SYSTEMS, TIMING MECHANISMS, AND TIME-VARYING CODES

F.L. ŢIPLEA, ERKKI MÄKINEN
AND CONSTANTIN ENEA

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

ISBN 951-44-5002-7
ISSN 1457-2060

SE-systems, Timing Mechanisms, and Time-Varying Codes

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

Constantin Enea

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

Abstract

We show that *synchronize extension systems* [11] can be succesfully used to simulate timing mechanisms incorporated into grammars and automata [2,9,5–7]. Further, we introduce the concept of a *time-varying code* as a natural generalization of L-codes, and the relationship with classical codes, gsm codes and SE-codes is established. Finally, a decision algorithm for periodically time-varying codes is given.

Keywords: formal languages, time-varying grammars, codes, synchronized extension systems.

1 Introduction and Preliminaries

A *synchronized extension system* (SE-system, for short) is a new powerful and elegant rewriting formalism which has proved to be useful in various kinds of problems in formal language theory [11–14].

In this paper we show how SE-systems can be used to simulate timing mechanisms used in grammars and automata. Further, we introduce the concept of a *time-varying code* as a natural generalization of L-codes, and the relationship

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

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

with classical codes, gsm codes and SE-codes is established. Finally, a decision algorithm for periodically time-varying codes is given.

We assume the reader to be familiar with the basic concepts of formal languages and automata as given e.g. in [4,9]. For the sake of self-containment, we recall some notations.

An *alphabet* is a finite non-empty set of *symbols*. For an alphabet V, V^* denotes the free monoid generated by V with the unit λ; V^+ is then $V^* - \{\lambda\}$. The elements of V^* are called *words*.

For a binary relation ρ over a set A, ρ^+ (ρ^*) denotes the transitive (reflexive and transitive) closure of ρ. $\mathbf{N}$ denotes the set of natural numbers and $\mathcal{P}(A)$ is the powerset of the set A. Given two natural numbers i and $p \geq 1$, $i \ mod \ p$ denotes the *remainder (residue) of* i *modulo* p.

Recall now some basic concepts from [11]. An *SE-system* 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. 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 this paper 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 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 of* v (from u). The *language of type* r^- generated by an SE-system $G = (V, L_1, L_2, S)$, denoted by $L^{r^-}(G)$, is the set of all words v having at least one r^--derivation, that is

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

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

The following important result has been proved in [11].

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

2 SE-Systems and Time-Varying Grammars

A *time-varying grammar* ([9]) is a couple (G, φ), where $G = (V_N, V_T, X_0, P)$ is a grammar and φ is a function from $\mathbf{N}$ into $\mathcal{P}(P)$. For a number $i \in \mathbf{N}$ and for words u and v, we write

$$(u, i) \Rightarrow_{(G,\varphi)} (v, i+1)$$

iff there is a rule $\alpha \to \beta \in \varphi(i)$ such that $u = u_1\alpha u_2$ and $v = u_1\beta u_2$.

The language generated by (G, φ) is defined by

$$L(G, \varphi) = \{w \in V_T^* \mid (X_0, 0) \overset{*}{\Rightarrow}_{(G,\varphi)} (w, i),\ \text{for some } i \in \mathbf{N}\}.$$

If the *timing function* φ is not restricted, then time-varying regular grammars (TVRG, for short) generate all the recursively enumerable languages ([9]). SE-systems can also generate all the recursively enumerable languages - we can easily construct an SE-system "simulating" a Chomsky grammar of type 0. However, for our purposes it is preferable to simulate TVRG's by SE-systems. In order to do that we will write natural numbers in a *unary notation* in which i is encoded by a sequence of $i+1$ copies of 1. For example, the unary notation of 4 is 11111. For notational convenience, we use the notation $[i]$ for the unary encoding of i.

Theorem 2 *Every recursively enumerable language can be expressed as an intersection between a language of type r^- generated by an SE-system and a regular language.*

Proof. Let L be a recursively enumerable language. Then there is a TVRG (G, φ), where $G = (V_N, V_T, X_0, P)$, such that $L = L(G, \varphi)$. Consider the SE-system $H = (V, L_1, L_2, S)$ given by

- $V = V_N \cup V_T \cup \{1\}$,
- $L_1 = \{X_0[0]\}$,
- $L_2 = \{A[i]aB[i+1] \mid i \in \mathbf{N}\ \wedge\ A \to aB \in \varphi(i)\} \cup$
  $\{A[i]a \mid i \in \mathbf{N}\ \wedge\ A \to a \in \varphi(i)\}$,
- $S = \{A[i] \mid i \in \mathbf{N}\ \wedge\ A \in lhs(\varphi(i))\}$, where $lhs(\varphi(i))$ is the set of all left hand sides of the rules in $\varphi(i)$.

It is clear that $L(G, \varphi) = L^{r^-}(H) \cap V_T^*$, which proves the theorem.

The construction in the proof of Theorem 2 can be easily adapted to simulate *time-varying non-deterministic finite automata with* λ*-moves* (TVNFA with λ-moves, for short) defined as in [5]. Such a device is a system $A = (Q, \Sigma, \delta, q_0, Q_f)$, where Q is the set of states, $q_0 \in Q$ is the initial state, $Q_f \subseteq Q$ is the set of final states, Σ is the (input) alphabet, and δ is a function from $Q \times \mathbf{N} \times (\Sigma \cup \{\lambda\})$ into $\mathcal{P}(Q)$. The computation defined by A is given by

$$(q, i, aw) \vdash_A (q', i+1, w) \quad \Leftrightarrow \quad q' \in \delta(q, i, a),$$

for all $q, q' \in Q$, $i \geq 0$, $w \in \Sigma^*$, and $a \in \Sigma \cup \{\lambda\}$.

An SE-system simulating a TVNFA with λ-moves can be constructed by associating to each move $q' \in \delta(q, i, a)$ the extending word $q[i]aq'[i+1]$ and the synchronization word $q[i]$.

If the timing function φ of a time-varying grammar is periodic (that is, there is $p \geq 1$ such that $\varphi(i) = \varphi(i \ mod \ p)$ for all $i \geq p$), the construction of an SE-system simulating a time-varying grammar can be simplified by replacing each occurrence of $[j]$ by $[j \ mod \ p]$, for all $j \geq 0$. Then, the languages L_2 and S become finite and, therefore, the SE-system obtained is of type (f, f, f). A similar construction can be done for periodic time-varying automata. Therefore, by Theorem 1, the following result holds.

Theorem 3 *All the languages generated by periodic TVRG's or accepted by periodic TVNFA's with* λ*-moves are regular.*

The regularity of languages accepted by periodic time-varying deterministic finite automata has been proved already in [5], but the result in Theorem 3 is more general.

3 Time-Varying Codes

In this section we introduce the concept of a time-varying code which is a natural generalization of the concept of an L-code [8]. First, we recall the concept of a code (for details, the reader is referred to [3,10]).

Let Δ be an alphabet. A *code over* Δ is any subset $C \subseteq \Delta^+$ such that each word $w \in \Delta^+$ has at most one decomposition over C. Alternatively, one can say that C is a code over Δ if there is an alphabet Σ and a function $h : \Sigma \to \Delta^+$ such that the unique homomorphic extension $\bar{h} : \Sigma^* \to \Delta^*$ of h defined by $\bar{h}(\lambda) = \lambda$ and $\bar{h}(a_0 \cdots a_{n-1}) = h(a_0) \cdots h(a_{n-1})$, for all $a_0 \cdots a_{n-1} \in \Sigma^+$, is injective.

Definition 4 *Let* Σ *and* Δ *be alphabets. A function* $h : \Sigma \times \mathbf{N} \to \Delta^+$ *is* *called a* time-varying code over Δ *(TV-code over Δ, for short) if the function $\bar{h} : \Sigma^* \to \Delta^*$ given by $\bar{h}(\lambda) = \lambda$ and*

$$\bar{h}(a_0 \cdots a_{n-1}) = h(a_0, 0) \cdots h(a_{n-1}, n-1),$$

for all $a_0 \cdots a_{n-1} \in \Sigma^+$, is injective.

A TV-code $h : \Sigma \times \mathbf{N} \to \Delta^+$ is called *periodic* if there is $p \geq 1$ such that $h(a, i) = h(a, i \bmod p)$, for all $a \in \Sigma$ and $i \geq p$; the number p is called a *period* of h.

Remark 5 *Let Σ and Δ be alphabets.*

(1) Any code $g : \Sigma \to \Delta^+$ is a TV-code. Indeed, let $h : \Sigma \times \mathbf{N} \to \Delta^+$ be defined by $h(a, i) = g(a)$ for all $a \in \Sigma$ and $i \in \mathbf{N}$. Then, it is clear that $\bar{g} = \bar{h}$.
(2) Let $h : \Sigma \times \mathbf{N} \to \Delta^+$ be a function. If the set $h(\Sigma \times \mathbf{N})$ is a code then h is a TV-code, but the converse does not hold generally.

In what follows, we relate TV-codes to different classes of codes introduced in the literature.

TV-codes and L-codes. *L-codes* have been introduced in [8] as functions $g : \Sigma \to \Sigma^+$ such that $\bar{g} : \Sigma^* \to \Sigma^*$ given by $\bar{g}(\lambda) = \lambda$ and

$$\bar{g}(a_0 \cdots a_{n-1}) = g^1(a_0) \cdots g^n(a_{n-1}),$$

for all $a_0 \cdots a_{n-1} \in \Sigma^+$, is injective. Here, g^i denotes the i^{th} iteration of the unique homomorphic extension of g, for all $i \geq 1$. (If g denotes also the unique homomorphic extension of g on Σ^*, then $g^1 = g$ and $g^{i+1} = g^i \circ g$ for all $i \geq 1$, where "$\circ$" is the function composition.)

Any L-code $g : \Sigma \to \Sigma^+$ is a TV-code. Indeed, let $h : \Sigma \times \mathbf{N} \to \Sigma^+$ be defined by $h(a, i) = g^{i+1}(a)$, for all $a \in \Sigma$ and $i \in \mathbf{N}$. Then, it is clear that $\bar{g} = \bar{h}$.

Proposition 6 *There are TV-codes that are not L-codes.*

Proof. Notice first that for each L-code $g : \Sigma \to \Sigma^+$ and each symbol $a \in \Sigma$ such that $g(a) = a^k$, for some $k > 1$, we have $g^i(a) = a^{k^i}$, for all $i \geq 1$.

Consider $h : \Sigma \times \mathbf{N} \to \Sigma^+$ defined by $h(a, 1) = a^2$ and $h(a, 2) = a$, for some $a \in \Sigma$. (The values $h(i, x)$, $(x, i) \in \Sigma \times \mathbf{N}$, are not of interest, provided that h is a TV-code.)

If there were an L-code g with the property $\bar{h} = \bar{g}$, the relation $\bar{h}(a) = \bar{g}(a)$ would imply $g(a) = a^2$, and $\bar{h}(aa) = \bar{g}(aa)$ would imply

$$aaa = \bar{h}(aa) = \bar{g}(aa) = g(a)g^2(a) = a^6,$$

which is a contradiction.

TV-codes and gsm-codes. Generalized Sequential Machines can be used in a very natural way as coders (see for example [1]): the input is the sequence to be encoded, and the output is the result.

A *generalized sequential machine* (*gsm*, for short) is a 6-tuple [4]

$$M = (Q, \Sigma, \Delta, \delta, q_0, F),$$

where Q is the set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, Σ is the input alphabet, Δ is the output alphabet, and δ is a function from $Q \times \Sigma$ into the powerset of $Q \times \Delta^*$.

We consider only gsm's with the following properties:

- F is the empty set; therefore, we omit it from the notation above;
- $\delta(q, a)$ is a singleton subset of $Q \times \Delta^+$, for all $q \in Q$ and $a \in \Sigma$; therefore, we write $\delta : Q \times \Sigma \to Q \times \Delta^+$ and say that M is *deterministic* and *λ-free.*

Notice that under these considerations δ is a total function (defined for all pairs $(q, a) \in Q \times \Sigma$).

A gsm M defines a function $g_M : \Sigma^* \to \Delta^*$ by letting $g_M(\lambda) = \lambda$ and

$$g_M(wa) = g_M(w)pr_2(\delta(pr_1(\tilde{\delta}(q_0, w)), a)),$$

for all $w \in \Sigma^*$ and $a \in \Sigma$, where pr_1 (pr_2) is the first (second) projection function and $\tilde{\delta}$ is the usual extension of δ to $Q \times \Sigma^*$.

A *gsm coder* is a gsm M such that g_M is injective; in this case, g_M is called a *gsm code.*

In order to relate gsm-codes to TV-codes we encounter a problem similar to that in Figure 3. That is, there are two states q_1 and q_2 in M which both can be reached from q_0 in equal number of steps (here in one step), and in these states the symbol a is encoded in two different ways. In such a case, we can not associate a TV-code h to g_M. For example, in the case of Figure 3, we have to define $h(a, 1) = ab$ and $h(a, 1) = ba$.

Definition 7 *A gsm M is called* equal *if there are two distinct states q and q' and an input symbol a such that q and q' can both be reached from q_0 in equal number of steps, and $pr_2(\delta(q, a)) \neq pr_2(\delta(q', a))$.*

If a gsm is not equal we call it *equal-free.* Now, we can prove:

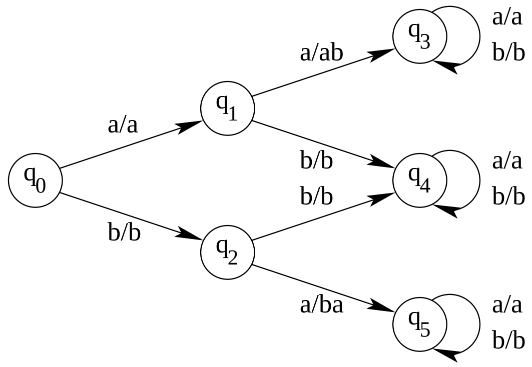

Fig. 1. An equal gsm

Proposition 8 *If an equal-free gsm M is a coder, then there is a TV-code h such that $\bar{g}_M = \bar{h}$.*

Proof. Let $M = (Q, \Sigma, \Delta, \delta, q_0)$ be an equal-free gsm. Define $h : \Sigma \times \mathbf{N} \to \Delta^+$ by

$$h(a, i) = pr_2(\delta(q, a)),$$

for all $a \in \Sigma$ and $i \in \mathbf{N}$, where q is a state reachable in i steps from q_0 (q_0 is reachable from itself in 0 steps).

It follows from the equal-freeness of M that h is well-defined. Then, we can easily check that $\bar{g}_M = \bar{h}$.

Not all gsm coders are equal-free as the gsm in Figure 3 shows us (it is a coder but it does not have the equal-freeness property).

The equal-freeness can be effectively checked. Indeed, for a gsm M we define the sequence of sets A_i, $i \geq 0$, as follows:

(i) $A_0 = \{q_0\}$;
(ii) $A_{i+1} = \{pr_1(\delta(q, a)) \mid q \in A_i,\ a \in \Sigma\}$, for all $i \geq 0$.

The sets A_i are finite because they are subsets of the finite set Q and, therefore, there are k and i_0 such that $k < i_0$ and $A_k = A_{i_0}$. Then, for each $j < i_0$, check for each pair of distinct states $q, q' \in A_j$, and for each input symbol $a \in \Sigma$, whether or not $\delta(q, a) = \delta(q', a)$. If the relation $\delta(q, a) = \delta(q', a)$ holds at least once, then M is equal; otherwise, it is equal-free.

A gsm coder can encode a symbol a only by the maximum of its outputs. Therefore, by using a similar idea than that in the previous paragraph, we can show that there are gsm codes (defined for equal-free gsm's) that are not L-codes.

TV-codes and SE-codes. Next we show that TV-codes are particular cases of SE-codes and, in case of a periodic function $h : \Sigma \times \mathbf{N} \to \Delta^+$, we can effectively decide whether or not h is a TV-code.

Two r^--derivations

$$u_1 \Rightarrow_{r^-} u_2 \Rightarrow_{r^-} \cdots \Rightarrow_{r^-} u_n$$

and

$$u'_1 \Rightarrow_{r^-} u'_2 \Rightarrow_{r^-} \cdots \Rightarrow_{r^-} u'_m$$

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

An SE-system G is called r^-*-ambiguous* if there is a word v having at least two distinct r^--derivations in G. If G is not r^--ambiguous then we say that it is r^-*-nonambiguous*.

An r^--derivation $u_1 \Rightarrow_{r^-} u_2 \Rightarrow_{r^-} \cdots \Rightarrow_{r^-} u_n$ 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 r^--derivation can be reduced in different ways. For example, the r^--derivation

$$u_1 \Rightarrow_{r^-} u_2 \Rightarrow_{r^-} u_3 \Rightarrow_{r^-} u_1 \Rightarrow_{r^-} u_4 \Rightarrow_{r^-} u_5 \Rightarrow_{r^-} u_3,$$

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

$$u_1 \Rightarrow_{r^-} u_4 \Rightarrow_{r^-} u_5 \Rightarrow_{r^-} u_3$$

or to

$$u_1 \Rightarrow_{r^-} u_2 \Rightarrow_{r^-} u_3.$$

If an SE-system has the property that for every word v there is at most a reduced r^--derivation of v, then it is called *weak* r^-*-nonambiguous*.

It is clear that an r^--nonambiguous SE-system is also weak r^--nonambiguous, but the converse does not hold in general. That is, there exist SE-systems G and words v with more than two r^--derivations. But, in this case, all the r^--derivations of such a word can be reduced, by removing cycles, to a unique reduced r^--derivation.

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

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

In [11] it has been proved that the (weak) r^--nonambiguity property is decidable for non-returning SE-systems of type (f, f, f). The proof is based on constructing a finite graph and checking the existence of some paths (with some properties). The relationship between codes and weak nonambiguous SE-systems has been also pointed out in [11]. That is, a set $C \subseteq \Delta^+$ is a code over Δ if and only if the SE-system $(V, C, C, \{\lambda\})$ is (weak) r^--nonambiguous.

Let $h : \Sigma \times \mathbf{N} \to \Delta^+$ be a function. We associate to h the SE-system $H = (V, L_1, L_2, S)$ given by:

- $V = \Sigma \cup \{1\}$,
- $L_1 = \{h(a, 0)[1] \mid a \in \Sigma\}$,
- $L_2 = \{[i]h(a, i)[i + 1] \mid (a, i) \in \Sigma \times \mathbf{N}\} \cup \{[i]h(a, i) \mid (a, i) \in \Sigma \times \mathbf{N}\}$,
- $S = \{[i] \mid i \in \mathbf{N}\}$

($[i + 1]$ in a word $[i]h(a, i)[i + 1]$ indicates the "next time").

Proposition 9 *Let $h : \Sigma \times \mathbf{N} \to \Delta^+$ be a function and H be the SE-system associated to h. Then, the following properties hold true:*

(1) H is a non-returning SE-system;
(2) h is a TV-code iff H is (weak) r^--nonambiguous.

Proof. Claim (1) follows directly from the definitions, and Claim (2) is an straightforward consequence of the following equivalences:

$$\begin{aligned} h \text{ is a TV-code iff } & (\forall v \in \Delta^+)(\text{there is at most an } u \in \Sigma^+ \text{ s.t. } \bar{h}(u) = v) \\ \text{iff } & (\forall v \in \Delta^+)(\text{there is at most an } r^-\text{-derivation of } v \text{ in } H). \end{aligned}$$

Consider now a periodic function $h : \Sigma \times \mathbf{N} \to \Delta^+$, and $p \geq 1$ a period of h. Modify the SE-system H associated to h by replacing each unary notation $[j]$ by $[j \ mod \ p]$, for all $j \geq 0$. Let H_p be the SE-system such obtained.

Proposition 10 *Let $h : \Sigma \times \mathbf{N} \to \Delta^+$ be a periodic function with period p, and let H_p be the SE-system associated to h as above. Then the following properties hold true:*

(1) H_p is a non-returning SE-system of type (f, f, f);
(2) h is a TV-code iff H_p is (weak) r^--nonambiguous.

Proof. Similar to that of Proposition 9 with the exception that there are only a finite number of residues modulo p.

Now, we can obtain the following result regarding periodic TV-codes.

Theorem 11 *It is decidable whether a periodic function $h : \Sigma \times \mathbf{N} \to \Delta^+$ is a TV-code or not.*

Proof. Let $p \geq 1$ be a period of h. Then, from Proposition 10 it follows that h is a TV-code if and only if H_p is r^--nonambiguous. Because H_p is a non-returning SE-system of type (f, f, f), it follows, by Theorem 4.2 of [11], that it is decidable whether or not H_p is r^--nonambiguous.

The proof of Theorem 11 suggests the following algorithm to check whether a periodic function $h : \Sigma \times \mathbf{N} \to \Delta^+$ is a TV-code or not.

Algorithm.
input: a periodic function $h : \Sigma \times \mathbf{N} \to \Delta^+$ with period p;
output: "yes" if h is a TV-code, otherwise "no";
begin
1. construct the SE-system H_p;
2. check whether or not H_p is r^--nonambiguous;
3. **if** H_p is r^--nonambiguous **then** answer "yes" **else** answer "no"

end.

The correctness of the algorithm above follows immediately from Proposition 10 and Theorem 11 (the checking operation from line 2 can be performed by an algorithm as the one in [11], Theorem 4.2).

References

[1] A. Atanasiu, A class of coders based on gsm, Acta Inform. 29 (1992) 779–791.

[2] R.M. Baer, E.H. Spanier, Referenced automata and metaregular families, J. Comput. Syst. Sci. 3 (1969) 423–446.

[3] J. Berstel, D. Perrin, Theory of Codes, Academic Press, 1985.

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

[5] K. Krithivasan, A. Das, Time varying finite automata, Int. J. Comput. Math. 19 (1986) 103–123.

[6] K. Krithivasan, V. Srinivasan, Time varying pushdown automata, Int. J. Comput. Math 24 (1988) 223–236.

[7] C. Matei, Time-varying grammars and referenced automata, In Gh. Păun (ed.), Mathematical Aspects of Natural and Formal Languages, World Scientific Series in Computer Science vol. 43 World Scientific, 1994, 285–292.

[8] H.A. Maurer, A. Salomaa, D. Wood, L codes and number systems, Theoret. Comput. Sci. 22 (1983) 331–346.

[9] A. Salomaa, Formal Languages, Academic Press, 1973.

[10] A. Salomaa, Jewels of Formal Language Theory, Computer Science Press, 1981.

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

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

[13] F.L. Ţiplea, E. Mäkinen, A note on SE-systems and regular canonical systems. Technical Report A-2000-14, Department of Computer and Information Sciences, University of Tampere (Finland), October 2000, `http://www.cs.uta.fi/reports/r2000.html` (Submitted).

[14] F.L. Ţiplea, E. Mäkinen, On SE-systems and monadic string rewriting systems. Technical Report A-2000-15, Department of Computer and Information Sciences, University of Tampere (Finland), November 2000, `http://www.cs.uta.fi/reports/r2000.html` (Submitted).