SCHEDAE INFORMATICAE
XXXXXXXXX VOLUME nn Year

On State-Synchronized Automata Systems

Alexander Meduna, Jiří Kučera
Formal Language Research Group, Department of Information Systems
Faculty of Information Technology, Brno University of Technology,
Božetěchova 2, 612 66 Brno, Czech Republic
e-mail: {*meduna, ikucera*}*@fit.vutbr.cz*

Abstract. In this paper, we introduce a new kind of automata systems, called *state-synchronized automata systems* of degree n. In general, they consists of n pushdown automata, referred to as their components. These systems can perform a computation step provided that the concatenation of the current states of all their components belongs to a prescribed control language. As its main result, the paper demonstrates that these systems characterize the family of recursively enumerable languages. In fact, this characterization is demostrated in both deterministic and nondeterministic versions of these systems. Restricting their components, these systems provides less computational power.

Keywords: state-synchronized automata systems, automata systems, pushdown automata, determinism, recursively enumerable languages

1. Introduction

At present, processing information in a largely discontinuous and concurrent way represents a common computational phenomenon. Indeed, consider a process p that deals with information i. During a single computational step, p can simultaneously perform several concurrent subprocesses, each of which works with a different piece of information in i. As obvious, computation like this necessitates a highly sophisticated communication and synchronization between the subprocesses. As a result, to explore computation of this kind systematically and rigorously, computer science needs formal models that adequately reflect this way of computation.

Traditionally, formal language theory has always provided computer science with language-defining models to explore various information processors mathematically, so it should do so for the purpose sketched above as well. However, the classical versions of these models, such as automata and grammars, work in a completely single and isolated way, so they can hardly serve as appropriate models of the computation sketched above. Therefore, a proper formalization of processors that work in the way described above necessitates an adaptation of classical automata and grammars so they work on words in a multiple and synchronised way. At the same time, any adaptation of this kind should conceptually maintain the original structure of these models as much as possible so computer science can quite naturally base its investigation upon these newly adapted models by analogy with the standard approach based upon their classical versions. Simply put, these new models should work on words in a communicating and synchronized way while keeping their structural conceptualization unchanged. That is why, over the past two decades or so, formal language theory has introduced several versions of these models that work in the above-mentioned way to some extend. More specifically, this theory has conceptualized systems whose components consist of several grammars or automata, whose communication is based on some prespecified strategy, usually referred to as communication protocols. Let us give a broad overview of the most important systems of this kind.

The early work on grammar systems, inspired by problem solving theory, was focused to model an agent systems, where agents use for cooperation so called "blackboard model". Such a systems, called *cooperating distributed grammar systems*[3], consist of finite number of Chomsky grammars[2] ("agents"). The "blackboard" here is represented by a sentential form shared by all grammars. The communication in these systems is performed in the way that when some grammar ends its derivation on the shared sentential form, another grammar continues. This task switching between grammars is driven by *derivation modes* (see [3]).

The *parallel communicating grammar systems*[3] use a different strategy. Each grammar works independently on its own sentential form. When some grammar generates a *query symbol*, all derivations are paused and this query symbol is substituted by corresponding sentential form from requested grammar. After that, the derivation process may continue. A similar strategy is used in *parallel communicating finite automata systems*[7], where the communication between finite automata components is realized by query states. If some finite automaton performs a move to a query state, then this state is replaced by the current state of corresponding requested component. The *parallel communicating pushdown automata systems*[4] use the query symbols in the same manner as in parallel communicating grammar systems, except that all operations are performed on pushdowns.

Another ways of communication between components can be found in [1] and [10]. In *multigenerative grammar systems*[10], the cooperation between components is synchronized either by rules or by nonterminal symbols. In these systems, all components work simultaneously on their own sentecial forms. If some component is unable to do its derivation, remaining components are blocked. The direct derivation in these systems is performed if and only if all components can perform their direct derivations and the rules used during these derivations form a tuple from a prescribed control set (for nonterminal symbols, the situation is similar).

In *n-accepting state-restricted pushdown automata systems*[1], the states of the components determine which components can do their moves and which components leave their configurations unchanged. This activity of components is changed during each computation step depending on the fact if an n-tuple formed from states is the left-hand side of some *switch rule* from the prescribed control set.

The present paper contributes to this vivid trend of formal langauge theory by introducing new automata systems whose behavior is elegantly synchronized by state-based restrictions. More specifically, these systems, called *state-synchronized automata systems* of degree n, have pushdown automata as their components. The synchronization is performed by finite control language containing words formed from states of particular components. In these systems, only first component can read from input tape. A computation step is performed if and only if all components can simultaneously do their moves and their states form a word from the control language.

The organization of this paper is as follows. After the next section with preliminary definitions, we give the definitions of state-synchronized automata systems, deterministic and nondeterministic. In the section with theoretical results, we study the accepting power of these systems with various types of components. We show that state-synchronized automata systems are able to accept every recursively enumerable language. It remains an open problem whether or not a deterministic state-synchronized automata systems with two or more one-turn pushdown automata can describe the whole family of recursively enumerable languages.

2. Preliminaries

We assume that the reader is familiar with the basic notions of formal language theory (see [5] and [8]). Let S be a set. Then, the cardinality of S is denoted by $\mathrm{card}(S)$. Let Σ be an alphabet. Then, Σ^* represents the free monoid generated by Σ under the operation of concatenation, with ε as the unit of Σ^*. Let ϱ be a (binary) relation. Then, ϱ^* denotes the reflexive and transitive closure of ϱ, and ϱ^i denotes the ith power of ϱ, $i \geq 0$. Let w be a word over Σ. Then, the length of w is denoted by $|w|$, and the set of all subwords contained in w is denoted by $\mathrm{subword}(w)$. **LIN**, **CF**, and **RE** denotes the family of linear languages, the family of context-free languages, and the family of recursively enumerable languages, respectively.

A *finite automaton* (FA) is a quintuple $M = (Q, \Sigma, R, s, F)$, where Q is a finite set of states, Σ is an input alphabet, $Q \cap \Sigma = \emptyset$, $R \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ is a finite set of rules, $s \in Q$ is the initial state, and $F \subseteq Q$ is a set of final states. Instead of $(p, a, q) \in R$, we write $pa \to q \in R$ throughout. A *configuration* of M is a word from $Q\Sigma^*$. The relation of *direct move* $\vdash_M \subseteq Q\Sigma^* \times Q\Sigma^*$ is defined as follows: if $w \in \Sigma^*$ and $pa \to q \in R$, then $paw \vdash_M qw$. The *language accepted by* M, $L(M)$, is defined as $L(M) = \{w \in \Sigma^* \mid sw \vdash_M^* f, f \in F\}$.

A *pushdown automaton* (PDA), is a septuple $M = (Q, \Sigma, \Gamma, R, s, S, F)$, where Q is a finite set of states, Σ is an input alphabet, Γ is a pushdown alphabet, Q,

Σ, and Γ are pairwise disjoint, $R \subseteq \Gamma \times Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \times Q$ is a finite set of rules, $s \in Q$ is the initial state, $S \in \Gamma$ is the initial symbol on pushdown, and $F \subseteq Q$ is a set of final states. Instead of $(A, p, a, x, q) \in R$, we write $Apa \to xq \in R$ throughout. A *configuration* of M is a word from $\Gamma^* Q \Sigma^*$. The relation of *direct move* $\vdash_M \subseteq \Gamma^* Q \Sigma^* \times \Gamma^* Q \Sigma^*$ is defined as follows: if $u \in \Gamma^*$, $w \in \Sigma^*$, and $Apa \to xq \in R$, then $uApaw \vdash_M uxqw$. The *language accepted by M by final state*, $L(M)_f$, is defined as $L(M)_f = \{w \in \Sigma^* \mid Ssw \vdash_M^* \gamma f, \gamma \in \Gamma^*, f \in F\}$. The *language accepted by M by empty pushdown*, $L(M)_\varepsilon$, is defined as $L(M)_\varepsilon = \{w \in \Sigma^* \mid Ssw \vdash_M^* q, q \in Q\}$. The *language accepted by M by final state and empty pushdown*, $L(M)_{f\varepsilon}$, is defined as $L(M)_{f\varepsilon} = \{w \in \Sigma^* \mid Ssw \vdash_M^* f, f \in F\}$. A pushdown automaton M is said to be *deterministic* (dPDA) if for every rule $Apa \to xq \in R$ it holds $\mathrm{card}(\{\alpha \to \gamma \in R \mid \alpha \in \mathrm{subword}(Apa)\}) = 1$.

The family of languages accepted by X by *final state*, *empty pushdown*, and *final state and empty pushdown*, where X is PDA or dPDA, is denoted by $\mathscr{L}(X)_f$, $\mathscr{L}(X)_\varepsilon$, and $\mathscr{L}(X)_{f\varepsilon}$, respectively.

Let $M = (Q, \Sigma, \Gamma, R, s, S, F)$ be a pushdown automaton, and let $xoabw \vdash_M ypbw \vdash_M zqw$, where $x, y, z \in \Gamma^*$, $o, p, q \in Q$, $a, b \in \Sigma \cup \{\varepsilon\}$, and $w \in \Sigma^*$. If $|y| \geq |x|$ and $|y| > |z|$, then during the move $ypbw \vdash_M zqw$, M makes *a turn* in its pushdown. We say that M is a *one-turn pushdown automaton* (one-turn PDA) if it makes no more than one turn in its pushdown during any computation.

A *two-pushdown automaton* is an 8-tuple $M = (Q, \Sigma, \Gamma, R, s, S_1, S_2, F)$, where Q, Σ, Γ, s, and F are defined as in PDA, Q, Σ, and Γ are pairwise disjoint, $S_1 \in \Gamma$ is the initial symbol on pushdown 1, $S_2 \in \Gamma$ is the initial symbol on pushdown 2, and $R \subseteq \Gamma \times \Gamma \times Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \times \Gamma^* \times Q$ is a finite set of rules. Instead of $(A, B, p, a, x, y, q) \in R$, we write $A\#Bpa \to x\#yq \in R$ throughout, where $\# \notin \Sigma \cup \Gamma \cup Q$. A *configuration* of M is a word from $\Gamma^*\{\#\}\Gamma^* Q \Sigma^*$. The relation of *direct move* $\vdash_M \subseteq \Gamma^*\{\#\}\Gamma^* Q \Sigma^* \times \Gamma^*\{\#\}\Gamma^* Q \Sigma^*$ is defined as follows: if $u, v \in \Gamma^*$, $w \in \Sigma^*$, and $A\#Bpa \to x\#yq \in R$, then $uA\#vBpaw \vdash_M ux\#vyqw$. The *language accepted by M by final state*, $L(M)_f$, is defined as $L(M)_f = \{w \in \Sigma^* \mid S_1\#S_2sw \vdash_M^* \gamma_1\#\gamma_2 f, \gamma_1, \gamma_2 \in \Gamma^*, f \in F\}$. The *language accepted by M by empty pushdown*, $L(M)_\varepsilon$, is defined as $L(M)_\varepsilon = \{w \in \Sigma^* \mid S_1\#S_2sw \vdash_M^* \#q, q \in Q\}$. The *language accepted by M by final state and empty pushdown*, $L(M)_{f\varepsilon}$, is defined as $L(M)_{f\varepsilon} = \{w \in \Sigma^* \mid S_1\#S_2sw \vdash_M^* \#f, f \in F\}$. A two-pushdown automaton M is said to be *deterministic* if for every rule $A\#Bpa \to x\#yq \in R$ it holds $\mathrm{card}(\{\alpha \to \gamma \in R \mid \alpha \in \mathrm{subword}(A\#Bpa)\}) = 1$.

Let $M = (Q, \Sigma, \Gamma, R, s, S_1, S_2, F)$ be a two-pushdown automaton. Let

$$x_1\#x_2oabw \vdash_M y_1\#y_2pbw \vdash_M z_1\#z_2qw$$

in M, where $o, p, q \in Q$, $a, b \in \Sigma \cup \{\varepsilon\}$, $w \in \Sigma^*$, and $x_1, x_2, y_1, y_2, z_1, z_2 \in \Gamma^*$. If $|y_i| \geq |x_i|$ and $|y_i| > |z_i|$ for some $i \in \{1, 2\}$, then during the move $y_1\#y_2pbw \vdash_M z_1\#z_2qw$, M makes *a turn* in pushdown i. If $|y_i| \geq |x_i|$ and $|y_i| > |z_i|$ for both $i = 1, 2$, then during move $y_1\#y_2pbw \vdash_M z_1\#z_2qw$, M makes *a simultaneous turn* in both pushdowns. We say that M is *one-turn* if it makes no more than one turn in either of its pushdowns during any computation. We say that M is *simultaneously one-turn* if it makes either no turn or one simultaneous turn in both pushdowns during any computation.

3. Definitions and Examples

This section recalls the definition of state-synchronized automata system from [6].

DEFINITION 1. *Let n be a positive integer. A* state-synchronized automata system *of degree n (*SCAS_n *for short) is an $(n+1)$-tuple $\Gamma = (M_1, M_2, \ldots, M_n, \Psi)$, where M_i is an FA or a PDA, and it is referred to as the ith* component *of Γ, for all $1 \leq i \leq n$. $\Psi \subseteq Q_1 Q_2 \ldots Q_n$ is a* control language *of Γ, where Q_i is the set of states in M_i, $1 \leq i \leq n$. Furthermore, Σ, Γ_i, s_i, S_i, and F_i denote the input alphabet of Γ, the pushdown alphabet of M_i, the initial state of M_i, the initial symbol on M_i's pushdown, and the set of final states in M_i, respectively, for all $1 \leq i \leq n$. If M_i is an FA, we set $\Gamma_i = \emptyset$ and $S_i = \varepsilon$, for all $1 \leq i \leq n$.*

A configuration *of Γ is an n-tuple $(\chi_1, \chi_2, \ldots, \chi_n)$, where χ_i is a configuration of M_i, for all $1 \leq i \leq n$.*

Let π_i be the mapping from $\Gamma_i^* Q_i \Sigma^*$ to Q_i such that $\pi_i(x_i q_i w) = q_i$, $x_i \in \Gamma_i^*$, $q_i \in Q_i$, $w \in \Sigma^*$, for all $1 \leq i \leq n$. Furthermore, let $\alpha = (\chi_1, \chi_2, \ldots, \chi_n)$ and $\alpha' = (\chi_1', \chi_2', \ldots, \chi_n')$ be two configurations of Γ. The relation of direct move *in Γ, $\vdash_\Gamma$, is defined as follows: if for every $1 \leq i \leq n$ it holds $\chi_i \vdash_{M_i} \chi_i'$, and $\pi_1(\chi_1)\pi_2(\chi_2)\ldots\pi_n(\chi_n) \in \Psi$, then $\alpha \vdash_\Gamma \alpha'$.*

Analogously to PDA, define three types of languages accepted by Γ *as follows:*

$$
\begin{aligned}
L(\Gamma)_f &= \{w \in \Sigma^* \mid (S_1 s_1 w, S_2 s_2, \ldots, S_n s_n) \vdash_\Gamma^* (\gamma_1 f_1, \gamma_2 f_2, \ldots, \gamma_n f_n), \\
&\qquad \gamma_i \in \Gamma_i^*, f_i \in F_i, 1 \leq i \leq n\}; \\
L(\Gamma)_\varepsilon &= \{w \in \Sigma^* \mid (S_1 s_1 w, S_2 s_2, \ldots, S_n s_n) \vdash_\Gamma^* (q_1, q_2, \ldots, q_n), \\
&\qquad q_i \in Q_i, q_i \in F_i \text{ if } M_i \text{ is an FA}, 1 \leq i \leq n\}; \\
L(\Gamma)_{f\varepsilon} &= \{w \in \Sigma^* \mid (S_1 s_1 w, S_2 s_2, \ldots, S_n s_n) \vdash_\Gamma^* (f_1, f_2, \ldots, f_n), \\
&\qquad f_i \in F_i, 1 \leq i \leq n\}.
\end{aligned}
$$

The following example demonstrates the capability of SCAS_n to accept a language which is not context-free.

EXAMPLE 1. *Let $\Gamma = (M, M', \Psi)$ be an SCAS_2, where M, M' are PDAs defined as*

- $M = (\{s, q_a, q_b, q_c, f\}, \{a, b, c\}, \{S, A, B, C\}, R, s, S, \{f\})$;
- $M' = (\{s', q_a', q_b', q_c', f'\}, \{a, b, c\}, \{S, A, B, C\}, R', s', S, \{f'\})$;
- $R = \{Ssa \to SAq_a, Aq_a a \to AAq_a, Aq_a b \to Aq_b, Aq_b b \to Aq_b, Aq_b c \to q_c, Aq_c c \to q_c, Sq_c \to f\}$;
- $R' = \{Ss' \to Sq_a', Sq_a' \to Sq_a', Sq_a' \to SBq_b', Bq_b' \to BBq_b', Bq_b' \to q_c', Bq_c' \to q_c', Sq_c' \to f'\}$.

Finally, Ψ is a control language of Γ defined as $\Psi = \{ss', q_a q_a', q_b q_b', q_c q_c'\}$. The word

aaabbbccc is accepted by Γ *in this way*

$$\begin{array}{rll}(Ssaaabbbccc, Ss') & \vdash_\Gamma & (SAq_aaabbbccc, Sq'_a)\\ & \vdash_\Gamma & (SAAq_aabbbccc, Sq'_a)\\ & \vdash_\Gamma & (SAAAq_abbbccc, Sq'_a)\\ & \vdash_\Gamma & (SAAAq_bbbccc, SBq'_b)\\ & \vdash_\Gamma & (SAAAq_bbccc, SBBq'_b)\\ & \vdash_\Gamma & (SAAAq_bccc, SBBBq'_b)\\ & \vdash_\Gamma & (SAAq_ccc, SBBq'_c)\\ & \vdash_\Gamma & (SAq_cc, SBq'_c)\\ & \vdash_\Gamma & (Sq_c, Sq'_c)\\ & \vdash_\Gamma & (f, f')\end{array}$$

Clearly, $L(\Gamma)_f = L(\Gamma)_\varepsilon = L(\Gamma)_{f\varepsilon} = \{a^n b^n c^n \mid n \geq 1\}$.

Observe that SCAS$_n$ Γ, $n \geq 1$, to make a successful final computation step, do not need to contain a word in Ψ which is formed from states from a final configuration. On the other hand, as will be shown later in Definition 2, it would be useful to introduce Ψ_f as

$$\begin{array}{rcl}\Psi_f = \Psi & \cup & \{q_1q_2\ldots q_n \mid (\gamma_1q_1, \gamma_2q_2, \ldots, \gamma_nq_n) \text{ is a final configuration of } \Gamma,\\ & & \gamma_i \in \Gamma_i^*, q_i \in Q_i, 1 \leq i \leq n\}.\end{array}$$

Let Γ be an SCAS$_n$, for some $n \geq 1$. By $\mathcal{R}_\Gamma$, we denote the Cartesian product $\mathcal{R}_\Gamma = R_1 \times R_2 \times \cdots \times R_n$, where R_i is the set of rules of the ith component of Γ, for all $1 \leq i \leq n$. If $\alpha \in \mathcal{R}_\Gamma$, then by $\alpha(i)$, we denote the ith element of α, and clearly $\alpha(i) \in R_i$, for all $1 \leq i \leq n$. For a rule $r = u \to v$, we use lhs(r) and rhs(r) as an abbrevation for the left-hand side and right-hand side of r, respectively, so lhs$(r) = u$ and rhs$(r) = v$.

Define the mapping π_l from $\mathcal{R}_\Gamma$ to $Q_1Q_2\ldots Q_n$ as follows:

$$\pi_l(\alpha) = \pi_1(\text{lhs}(\alpha(1)))\pi_2(\text{lhs}(\alpha(2)))\ldots\pi_n(\text{lhs}(\alpha(n))),$$

that is π_l maps an n-tuple of rules

$$(A_1p_1a_1 \to x_1q_1, A_2p_2 \to x_2q_2, \ldots, A_np_n \to x_nq_n)$$

to the word $p_1p_2\ldots p_n$. The mapping π_r from $\mathcal{R}_\Gamma$ to $Q_1Q_2\ldots Q_n$ is defined analogously as $\pi_r(\alpha) = \pi_1(\text{rhs}(\alpha(1)))\pi_2(\text{rhs}(\alpha(2)))\ldots\pi_n(\text{rhs}(\alpha(n)))$.

We are now at position to introduce the definition of determinism in SCAS$_n$, which is given formally in Definition 2.

Definition 2. *Let* Γ *be an* SCAS$_n$*, for some* $n \geq 1$*. Then* Γ *is said to be* deterministic *(*dSCAS$_n$*) if for every* $\alpha \in \mathcal{R}_\Gamma$*, such that* $\pi_l(\alpha) \in \Psi$ *and* $\pi_r(\alpha) \in \Psi_f$*, it holds*

$$\text{card}(\{\alpha' \in \mathcal{R}_\Gamma \mid \pi_r(\alpha') \in \Psi_f, \text{lhs}(\alpha'(i)) \in \text{subword}(\text{lhs}(\alpha(i))), 1 \leq i \leq n\}) = 1.$$

Clearly, the SCAS$_2$ from Example 1 is deterministic. The following example shows the difference between deterministic and nondeterministic SCAS$_n$ according to Definition 2.

EXAMPLE 2. *Consider we have* SCAS_2 $\Gamma = (M_1, M_2, \Psi)$, *where* M_1, M_2 *are PDAs defined as*

- $M_1 = (\{s, q_1, q_2, f\}, \{a, b\}, \{S, A, B\}, \{r_1, r_2, r_3, r_4\}, s, S, \{f\})$ *and*
- $M_2 = (\{s', q_1', q_2', f'\}, \{a, b\}, \{S, A, B\}, \{r_1', r_2', r_3', r_4'\}, s', S, \{f'\})$, *where*

$$\begin{array}{lcllcl} r_1 & = & Ssa \to Sq_1; & r_1' & = & Ss' \to Sq_1'; \\ r_2 & = & Ssb \to Sq_2; & r_2' & = & Ss' \to Sq_2'; \\ r_3 & = & Sq_1 \to Sf; & r_3' & = & Sq_1' \to SAf'; \\ r_4 & = & Sq_2 \to Sf; & r_4' & = & Sq_2' \to SBf'. \end{array}$$

Let $\Psi = \{ss', q_1q_1', q_2q_2'\}$. *This* SCAS_2, *depending on its input, modify either its first or second pushdown. According to Definition 2, the allowed combinations of rules are:*

$$\begin{array}{l} (r_1, r_1'): \ \mathrm{card}(\{(r_1, r_1')\}) = 1; \\ (r_2, r_2'): \ \mathrm{card}(\{(r_2, r_2')\}) = 1; \\ (r_3, r_3'): \ \mathrm{card}(\{(r_3, r_3')\}) = 1; \\ (r_4, r_4'): \ \mathrm{card}(\{(r_4, r_4')\}) = 1; \end{array}$$

so Γ *is a* dSCAS_2. *Observe that condition* $\pi_r(\alpha') \in \Psi_f$ *is necessary because its omission leads to*

$$\begin{array}{l} (r_1, r_1'): \ \mathrm{card}(\{(r_1, r_1'), (r_1, r_2')\}) = 2; \\ (r_2, r_2'): \ \mathrm{card}(\{(r_2, r_2'), (r_2, r_1')\}) = 2; \\ (r_3, r_3'): \ \mathrm{card}(\{(r_3, r_3')\}) = 1; \\ (r_4, r_4'): \ \mathrm{card}(\{(r_4, r_4')\}) = 1; \end{array}$$

which is in contradiction with the fact that Γ *is deterministic.*

4. Theoretical Results

Since deterministic SCAS_ns are special cases of ordinary SCAS_ns, we show firstly that every recursively enumerable language can be accepted by some deterministic SCAS_n.

THEOREM 1. *For every recursively enumerable language* L *over an alphabet* Σ, *there exists a deterministic* SCAS_2, $\Gamma = (M_1, M_2, \Psi)$, *where*

$$M_i = (Q_i, \Sigma, \Gamma_i, R_i, s_i, S_i, F_i)$$

is a PDA, $1 \leq i \leq 2$, *such that* $L(\Gamma)_f = L$.

PROOF. Let L be a recursively enumerable language over an alphabet Σ. Then there exists a deterministic two-pushdown automaton, M, such that $L = L(M)_f$ (see Theorem 8.2.3.3 in [8]). Let $M = (Q, \Sigma, \hat{\Gamma}, R, s, S_1, S_2, F)$ be a deterministic two-pushdown automaton such that $L = L(M)_f$. From M, we construct a dSCAS_2 Γ in the following way.

1. Set $Q_1 = \emptyset$, $Q_2 = \emptyset$, and $\Psi = \emptyset$.
2. Set $\Gamma_1 = \Gamma_2 = \hat{\Gamma}$.
3. For every rule $r = A\#Bpa \to x\#yq$ from R:
   - add states $\bar{p}$, $\langle\bar{r}\rangle$, and $\bar{q}$ to Q_1;
   - add states $\hat{p}$, $\langle\hat{r}\rangle$, and $\hat{q}$ to Q_2;
   - add rules $A\bar{p}a \to A\langle\bar{r}\rangle$ and $A\langle\bar{r}\rangle \to x\bar{q}$ to R_1;
   - add rules $B\hat{p} \to B\langle\hat{r}\rangle$ and $B\langle\hat{r}\rangle \to y\hat{q}$ to R_2;
   - add words $\bar{p}\hat{p}$, $\bar{q}\hat{q}$, and $\langle\bar{r}\rangle\langle\hat{r}\rangle$ to Ψ.
4. Set $F_1 = \{\bar{f} \mid f \in F\}$ and $F_2 = \{\hat{f} \mid f \in F\}$.

From the construction above follows that if M is deterministic, then Γ must be also deterministic. Next, we prove that $L(M)_f = L(\Gamma)_f$.

Claim 2. *If* $u_0\#v_0q_0w_0 \vdash_M^i u_i\#v_iq_iw_i$, *then*

$$(u_0\bar{q}_0w_0, v_0\hat{q}_0) \vdash_\Gamma^{2i} (u_i\bar{q}_iw_i, v_i\hat{q}_i),$$

where $u_i \in \Gamma_1^*$, $v_i \in \Gamma_2^*$, $q_i \in Q$, $\bar{q}_i \in Q_1$, $\hat{q}_i \in Q_2$, *and* $w_i \in \Sigma^*$, *for all* $i \geq 0$.

Proof. The proof is established by induction on $i \geq 0$.
Basis: For $i = 0$, $u_0\#v_0q_0w_0 \vdash_M^0 u_0\#v_0q_0w_0$ implies that

$$(u_0\bar{q}_0w_0, v_0\hat{q}_0) \vdash_\Gamma^0 (u_0\bar{q}_0w_0, v_0\hat{q}_0).$$

Thus, the basis holds.

Induction Hypothesis: Suppose that the claim holds for all $0 \leq i \leq k$, for some $k \geq 0$.

Induction Step: If

$$u_0\#v_0q_0w_0 \vdash_M^k u_k\#v_kq_kw_k \vdash_M u_{k+1}\#v_{k+1}q_{k+1}w_{k+1},$$

then there exists a rule $r = A\#Bq_ka \to x\#yq_{k+1}$ in R such that $u_k = \gamma A$, $v_k = \delta B$, $u_{k+1} = \gamma x$, $v_{k+1} = \delta y$, and $w_k = aw_{k+1}$, where $\gamma \in \Gamma_1^*$, $\delta \in \Gamma_2^*$. Then, there also exist rules $A\bar{q}_ka \to A\langle\bar{r}\rangle \in R_1$, $A\langle\bar{r}\rangle \to x\bar{q}_{k+1} \in R_1$, $B\hat{q}_k \to B\langle\hat{r}\rangle \in R_2$, $B\langle\hat{r}\rangle \to y\hat{q}_{k+1} \in R_2$, and words $\bar{q}_k\hat{q}_k, \bar{q}_{k+1}\hat{q}_{k+1}, \langle\bar{r}\rangle\langle\hat{r}\rangle \in \Psi$, which implies that

$$\begin{aligned}(u_0\bar{q}_0w_0, v_0\hat{q}_0) &\vdash_\Gamma^{2k} (u_k\bar{q}_kw_k, v_k\hat{q}_k)\\ &\vdash_\Gamma (u_k\langle\bar{r}\rangle w_{k+1}, v_k\langle\hat{r}\rangle)\\ &\vdash_\Gamma (u_{k+1}\bar{q}_{k+1}w_{k+1}, v_{k+1}\hat{q}_{k+1}),\end{aligned}$$

and the claim holds for $k+1$, too. Therefore, Claim 2 holds. □

Claim 3. *If*

$$(u_0\bar{q}_0w_0, v_0\hat{q}_0) \vdash_\Gamma^{2i} (u_i\bar{q}_iw_i, v_i\hat{q}_i),$$

then $u_0\#v_0q_0w_0 \vdash_M^i u_i\#v_iq_iw_i$, *where* $u_i \in \Gamma_1^*$, $v_i \in \Gamma_2^*$, $q_i \in Q$, $\bar{q}_i \in Q_1$, $\hat{q}_i \in Q_2$, *and* $w_i \in \Sigma^*$, *for all* $i \geq 0$.

PROOF. The proof is established by induction on $i \geq 0$. Observe that only even number of computation steps are possible in Γ (see the construction of Γ from M above).

Basis: For $i = 0$,

$$(u_0\bar{q}_0w_0, v_0\hat{q}_0) \vdash_{\Gamma}^{0} (u_0\bar{q}_0w_0, v_0\hat{q}_0)$$

implies that $u_0\#v_0q_0w_0 \vdash_M^0 u_0\#v_0q_0w_0$. Thus, the basis holds.

Induction Hypothesis: Suppose that the claim holds for all $0 \leq i \leq k$, for some $k \geq 0$.

Induction Step: If

$$\begin{array}{lll} (u_0\bar{q}_0w_0, v_0\hat{q}_0) & \vdash_{\Gamma}^{2k} & (u_k\bar{q}_kw_k, v_k\hat{q}_k) \\ & \vdash_{\Gamma} & (u_k\bar{\langle r\rangle}w_{k+1}, v_k\hat{\langle r\rangle}) \\ & \vdash_{\Gamma} & (u_{k+1}\bar{q}_{k+1}w_{k+1}, v_{k+1}\hat{q}_{k+1}), \end{array}$$

then there exist rules $A\bar{q}_ka \to A\bar{\langle r\rangle} \in R_1$, $A\bar{\langle r\rangle} \to x\bar{q}_{k+1} \in R_1$, $B\hat{q}_k \to B\hat{\langle r\rangle} \in R_2$, $B\hat{\langle r\rangle} \to y\hat{q}_{k+1} \in R_2$, and words $\bar{q}_k\hat{q}_k, \bar{q}_{k+1}\hat{q}_{k+1}, \bar{\langle r\rangle}\hat{\langle r\rangle} \in \Psi$, such that $u_k = \gamma A$, $v_k = \delta B$, $u_{k+1} = \gamma x$, $v_{k+1} = \delta y$, $\gamma \in \Gamma_1^*$, $\delta \in \Gamma_2^*$, $w_k = aw_{k+1}$, and $r = A\#Bq_ka \to x\#yq_{k+1} \in R$. This immediately implies that

$$u_0\#v_0q_0w_0 \vdash_M^k u_k\#v_kq_kw_k \vdash_M u_{k+1}\#v_{k+1}q_{k+1}w_{k+1},$$

which proves that the claim holds for $k + 1$, too. Therefore, Claim 3 holds. □

From Claim 2 and Claim 3, it immediately follows that for every $w \in \Sigma^*$,

$$S_1\#S_2sw \vdash_M^* \gamma_1\#\gamma_2f \quad \text{iff} \quad (S_1s_1w, S_2s_2) \vdash_{\Gamma}^* (\gamma_1f_1, \gamma_2f_2),$$

where $\gamma_1 \in \Gamma_1^*$, $\gamma_2 \in \Gamma_2^*$. Therefore, $L(M)_f = L(\Gamma)_f$, and Theorem 1 holds. □

THEOREM 4. *For every recursively enumerable language L over an alphabet Σ, there exists a deterministic* SCAS_2, $\Gamma = (M_1, M_2, \Psi)$, *where*

$$M_i = (Q_i, \Sigma, \Gamma_i, R_i, s_i, S_i, F_i)$$

is a PDA, $1 \leq i \leq 2$, such that $L(\Gamma)_\varepsilon = L$.

PROOF. Let L be a recursively enumerable language over an alphabet Σ. Then there exists a deterministic two-pushdown automaton, M, such that $L = L(M)_\varepsilon$ (see Chapter 8 in [8]). The rest of proof is analogous as in Theorem 1. □

THEOREM 5. *For every recursively enumerable language L over an alphabet Σ, there exists a deterministic* SCAS_2, $\Gamma = (M_1, M_2, \Psi)$, *where*

$$M_i = (Q_i, \Sigma, \Gamma_i, R_i, s_i, S_i, F_i)$$

is a PDA, $1 \leq i \leq 2$, such that $L(\Gamma)_{f\varepsilon} = L$.

PROOF. The proof is analogous to the proof of Theorem 4. □

THEOREM 6. *Let $k \geq 3$. For every recursively enumerable language L over an alphabet Σ, there exists a deterministic* SCAS_k*,*

$$\Gamma = (M_1, M_2, \ldots, M_k, \Psi),$$

where M_i is a PDA, for all $1 \leq i \leq k$, such that $L = L(\Gamma)_f$.

PROOF. Let L be a recursively enumerable language over an alphabet Σ. Then, by Theorem 1, there exists a deterministic SCAS_2, Γ', such that all its components are PDAs and $L = L(\Gamma')_f$. Let $\Gamma' = (M'_1, M'_2, \Psi')$ be a deterministic SCAS_2, where M'_i is a PDA, $1 \leq i \leq 2$, such that $L = L(\Gamma')_f$. Let $k \geq 3$. From Γ', we construct a deterministic SCAS_k,

$$\Gamma = (M_1, M_2, \ldots, M_k, \Psi),$$

where M_i is a PDA, for all $1 \leq i \leq k$, in the following way.

1. $M_1 = M'_1$, $M_2 = M'_2$;
2. $M_i = (\{s_i\}, \Sigma, \{S_i\}, \{S_i s_i \to S_i s_i\}, s_i, S_i, \{s_i\})$, for all $3 \leq i \leq k$;
3. $\Psi = \Psi'\{s_3 s_4 \ldots s_k\}$.

It is obvious that components M_3 to M_k are redudant in Γ. Therefore, $L(\Gamma)_f = L(\Gamma')_f$, and Theorem 6 holds. □

THEOREM 7. *Let $k \geq 3$. For every recursively enumerable language L over an alphabet Σ, there exists a deterministic* SCAS_k*,*

$$\Gamma = (M_1, M_2, \ldots, M_k, \Psi),$$

where M_i is a PDA, for all $1 \leq i \leq k$, such that $L = L(\Gamma)_\varepsilon$.

PROOF. Let L be a recursively enumerable language over an alphabet Σ. Then, by Theorem 4, there exists a deterministic SCAS_2, Γ', such that all its components are PDAs and $L = L(\Gamma')_\varepsilon$. Let $\Gamma' = (M'_1, M'_2, \Psi')$ be a deterministic SCAS_2, where M'_i is a PDA, $1 \leq i \leq 2$, such that $L = L(\Gamma')_\varepsilon$. Let $k \geq 3$. From Γ', we construct a deterministic SCAS_k,

$$\Gamma = (M_1, M_2, \ldots, M_k, \Psi),$$

where M_i is a PDA, for all $1 \leq i \leq k$, in the following way.

1. Let $M'_1 = (Q'_1, \Sigma, \Gamma'_1, R'_1, s'_1, S'_1, F'_1)$. Then $M_1 = (Q_1, \Sigma, \Gamma_1, R_1, s_1, S_1, F'_1)$, where
   - $Q_1 = Q'_1 \cup \{s_1, q_1\}$, where $s_1, q_1 \notin Q'_1$;
   - $\Gamma_1 = \Gamma'_1 \cup \{S_1\}$, where $S_1 \notin \Gamma'_1$;
   - $R_1 = R'_1 \cup \{S_1 s_1 \to S_1 S'_1 s'_1, S_1 p \to q_1 \mid p \in Q'_1\}$.
2. Analogously, we construct M_2 from M'_2.
3. For all $3 \leq i \leq k$, set $M_i = (\{s_i, p_i, q_i\}, \Sigma, \{S_i\}, \{S_i s_i \to S_i p_i, S_i p_i \to S_i p_i, S_i p_i \to q_i\}, s_i, S_i, \emptyset)$.

4. Set $\Psi = \Psi'\{p_3 p_4 \dots p_k\} \cup \{s_1 s_2 \dots s_k, q_1 q_2 \dots q_k\}$.

Γ works in this way:

1. During its move from s_i to s'_i, M_i pushes S'_i on its pushdown, $1 \leq i \leq 2$. Simultaneously, M_j moves from s_j to p_j, for all $3 \leq j \leq k$.

2. Γ accepts (or rejects) its input word. During this phase, M_1 and M_2 perform their moves by using the same sequences of rules like M'_1 and M'_2, respectively. M_3 through M_k do loops over states p_3 through p_k, respectively.

3. When M'_1 and M'_2 in Γ' empty their pushdowns, the pushdowns of M_1 and M_2 in Γ have S_1 and S_2 on their tops, respectively. At this point, M_1 through M_k deterministically empty their pushdowns by moving to states q_1 through q_k, respectively.

Thus, Γ accepts its input word if and only if Γ' accepts its input word, which completes the proof of Theorem 7. □

THEOREM 8. *Let $k \geq 3$. For every recursively enumerable language L over an alphabet Σ, there exists a deterministic* SCAS_k*,*

$$\Gamma = (M_1, M_2, \dots, M_k, \Psi),$$

where M_i is a PDA, for all $1 \leq i \leq k$, such that $L = L(\Gamma)_{f\varepsilon}$.

PROOF. Prove this by analogy with Theorem 7 except that states q_1 through q_k are final. □

COROLLARY 9. *Let $n \geq 2$. For every recursively enumerable language L, there exists a deterministic* SCAS_n*, Γ, such that all its components are PDAs and $L = L(\Gamma)_f$.*

PROOF. This follows from Theorem 1 and Theorem 6. □

COROLLARY 10. *Let $n \geq 2$. For every recursively enumerable language L, there exists a deterministic* SCAS_n*, Γ, such that all its components are PDAs and $L = L(\Gamma)_\varepsilon$.*

PROOF. This follows from Theorem 4 and Theorem 7. □

COROLLARY 11. *Let $n \geq 2$. For every recursively enumerable language L, there exists a deterministic* SCAS_n*, Γ, such that all its components are PDAs and $L = L(\Gamma)_{f\varepsilon}$.*

PROOF. This follows from Theorem 5 and Theorem 8. □

The family of recursively enumerable languages can be also characterized by SCAS_n, where $n \geq 2$, such that all its components are one-turn PDAs.

THEOREM 12. *Let $x \in \{f, \varepsilon, f\varepsilon\}$. For every recursively enumerable language L over an alphabet Σ, there exists an* SCAS_2, *$\Gamma = (M_1, M_2, \Psi)$, where both M_1 and M_2 are one-turn pushdown automata, such that $L = L(\Gamma)_x$.*

PROOF. Let $x \in \{f, \varepsilon, f\varepsilon\}$. Let L be a recursively enumerable language over an alphabet Σ. In [9], it was shown that there exists a simultaneously one-turn two-pushdown automaton, M, such that $L = L(M)_x$. We can construct Γ from M in the exactly same way as demonstrated in Theorem 1, and we can use the same proof to show the accepted language identity of Γ and M. From the construction of Γ and from the definition of one-turn two pushdown automaton follows that M_1 and M_2 are both one-turn pushdown automata. □

THEOREM 13. *Let $x \in \{f, \varepsilon, f\varepsilon\}$ and $k \geq 3$. For every recursively enumerable language L over an alphabet Σ, there exists an* SCAS_k,

$$\Gamma = (M_1, M_2, \ldots, M_k, \Psi),$$

where M_i is a one-turn PDA, for all $1 \leq i \leq k$, such that $L = L(\Gamma)_x$.

PROOF. Let $x \in \{f, \varepsilon, f\varepsilon\}$ and $k \geq 3$. Let L be a recursively enumerable language over an alphabet Σ. Then, by Theorem 12, there exists an SCAS_2, $\Gamma' = (M_1', M_2', \Psi')$, where M_1' and M_2' are both one-turn PDAs, such that $L = L(\Gamma')_x$. From Γ', we can construct Γ, and then proof the identity of languages accepted by Γ' and Γ, in the same way as demonstrated in proofs of Theorem 6, Theorem 7, and Theorem 8. Observe that components M_3 through M_k of Γ are all one-turn PDAs. □

COROLLARY 14. *Let $x \in \{f, \varepsilon, f\varepsilon\}$ and $n \geq 2$. For every recursively enumerable language L, there exists an* SCAS_n, *Γ, such that all its components are one-turn PDAs and $L = L(\Gamma)_x$.*

PROOF. This follows from Theorem 12 and Theorem 13. □

The previous results are summarized in the following theorem.

THEOREM 15. *For every $L \in \mathbf{RE}$, there are*

a) an SCAS_n *Γ such that all its components are PDAs and $L = L(\Gamma)_x$, where $n \geq 2$ and $x \in \{f, \varepsilon, f\varepsilon\}$;*

b) an dSCAS_n *Γ such that all its components are PDAs and $L = L(\Gamma)_x$, where $n \geq 2$ and $x \in \{f, \varepsilon, f\varepsilon\}$;*

c) an SCAS_n *Γ such that all its components are one-turn PDAs and $L = L(\Gamma)_x$, where $n \geq 2$ and $x \in \{f, \varepsilon, f\varepsilon\}$.*

PROOF. This theorem follows from Corollary 9, Corollary 10, Corollary 11, Corollary 14, and from the obvious observation that deterministic SCAS_ns are no more powerful than nondeterministic SCAS_ns. □

Next, we study the case when no more than one pushdown component is permitted in SCAS_n.

LEMMA 16. *Let $x \in \{f, \varepsilon, f\varepsilon\}$ and $n \geq 1$. For every SCAS_n,*

$$\Gamma = (M_1, M_2, \ldots, M_n, \Psi),$$

where for some $1 \leq i \leq n$, M_i is a PDA, and for all $1 \leq j \leq n$, $i \neq j$, M_j is an FA, there exists a PDA, M, such that $L(M)_x = L(\Gamma)_x$.

PROOF. Let $x \in \{f, \varepsilon, f\varepsilon\}$. For $n = 1$, Γ has only one component, which is a PDA, and therefore Lemma 16 holds immediately. Further is shown that the lemma holds also for $n \geq 2$, where the case that the only first component of Γ can be a PDA is considered (the other situations can be proved analogously).

Let $n \geq 2$ and $\Gamma = (M_1, M_2, \ldots, M_n, \Psi)$ be an SCAS_n, where M_1 is a PDA, and M_2 trough M_n are FAs. From Γ, we construct a PDA, M, such that $L(M)_x = L(\Gamma)_x$, in the following way.

1. Let $M_1 = (Q_1, \Sigma, \Gamma_1, R_1, s_1, S_1, F_1)$ be a PDA from Γ, and let

   $$M_i = (Q_i, \Sigma, R_i, s_i, F_i)$$

   be an FA from Γ, for all $2 \leq i \leq n$.

2. Set $M = (Q, \Sigma, \Gamma_1, R, \langle s_1 s_2 \ldots s_n \rangle, S_1, F)$, where

   - $Q = \{\langle \omega \rangle \mid \omega \in \Psi \cup F_1 F_2 \ldots F_n\}$;
   - $R = \{A\langle \omega_1 \rangle a \to x\langle \omega_2 \rangle \mid Ap_1 a \to xq_1 \in R_1, p_2 \to q_2 \in R_2, \ldots, p_n \to q_n \in R_n, \omega_1 = p_1 p_2 \ldots p_n, \omega_2 = q_1 q_2 \ldots q_n, \langle \omega_1 \rangle, \langle \omega_2 \rangle \in Q\}$;
   - $F = \{\langle \omega \rangle \mid \omega \in F_1 F_2 \ldots F_n\}$.

To proof that $L(\Gamma)_x = L(M)_x$, we first establish the following two claims.

CLAIM 17. *If*

$$(up_1 w, p_2, \ldots, p_n) \vdash_\Gamma^i (u' q_1 w', q_2, \ldots, q_n),$$

then $u\langle p_1 p_2 \ldots p_n \rangle w \vdash_M^i u'\langle q_1 q_2 \ldots q_n \rangle w'$, where $u, u' \in \Gamma_1^*$, $w, w' \in \Sigma^*$, $p_i, q_i \in Q_i$, $1 \leq i \leq n$, and $\langle p_1 p_2 \ldots p_n \rangle, \langle q_1 q_2 \ldots q_n \rangle \in Q$.

PROOF. The proof is made by induction on $i \geq 0$.

Basis: For $i = 0$, we have

$$(up_1 w, p_2, \ldots, p_n) \vdash_\Gamma^0 (up_1 w, p_2, \ldots, p_n)$$

implies $u\langle p_1 p_2 \ldots p_n \rangle w \vdash_M^0 u\langle p_1 p_2 \ldots p_n \rangle w$, so the claim holds for $i = 0$.

Induction Hypothesis: Suppose that the claim holds for all $0 \leq i \leq k$, for some $k \geq 0$.

Induction Step: If

$$(uAp_1aw, p_2, \ldots, p_n) \vdash_\Gamma (uxo_1w, o_2, \ldots, o_n) \vdash_\Gamma^k (u'q_1w', q_2, \ldots, q_n),$$

then there exist rules $Ap_1a \to xo_1 \in R_1$, $p_j \to o_j \in R_j$, $2 \leq j \leq n$, and words $p_1p_2 \ldots p_n, o_1o_2 \ldots o_n \in \Psi \cup F_1F_2 \ldots F_n$. According to the construction of M from Γ above, this implies that there also exists a rule $A\langle p_1p_2 \ldots p_n\rangle a \to x\langle o_1o_2 \ldots o_n\rangle \in R$, so

$$uA\langle p_1p_2 \ldots p_n\rangle aw \vdash_M ux\langle o_1o_2 \ldots o_n\rangle w \vdash_M^k u'\langle q_1q_2 \ldots q_n\rangle w',$$

and the claim holds for $k+1$, too. Therefore, Claim 17 holds. □

Claim 18. *If*

$$u\langle p_1p_2 \ldots p_n\rangle w \vdash_M^i u'\langle q_1q_2 \ldots q_n\rangle w',$$

then $(up_1w, p_2, \ldots, p_n) \vdash_\Gamma^i (u'q_1w', q_2, \ldots, q_n)$, *where* $u, u' \in \Gamma_1^*$, $w, w' \in \Sigma^*$, $p_i, q_i \in Q_i$, $1 \leq i \leq n$, *and* $\langle p_1p_2 \ldots p_n\rangle, \langle q_1q_2 \ldots q_n\rangle \in Q$.

Proof. The proof is made by induction on $i \geq 0$.

Basis: For $i = 0$, we have that

$$u\langle p_1p_2 \ldots p_n\rangle w \vdash_M^0 u\langle p_1p_2 \ldots p_n\rangle w$$

implies $(up_1w, p_2, \ldots, p_n) \vdash_\Gamma^0 (up_1w, p_2, \ldots, p_n)$, so the claim holds for $i = 0$.

Induction Hypothesis: Suppose that the claim holds for all $0 \leq i \leq k$, for some $k \geq 0$.

Induction Step: If

$$uA\langle p_1p_2 \ldots p_n\rangle aw \vdash_M ux\langle o_1o_2 \ldots o_n\rangle w \vdash_M^k u'\langle q_1q_2 \ldots q_n\rangle w',$$

then there exists a rule $A\langle p_1p_2 \ldots p_n\rangle a \to x\langle o_1o_2 \ldots o_n\rangle \in R$, which implies that there must exist words $p_1p_2 \ldots p_n, o_1o_2 \ldots o_n \in \Psi \cup F_1F_2 \ldots F_n$, and rules $Ap_1a \to xo_1 \in R_1$, $p_j \to o_j \in R_j$, $2 \leq j \leq n$, in Γ, so

$$(uAp_1aw, p_2, \ldots, p_n) \vdash_\Gamma (uxo_1w, o_2, \ldots, o_n) \vdash_\Gamma^k (u'q_1w', q_2, \ldots, q_n),$$

and the claim holds for $k+1$, too. Therefore, Claim 18 holds. □

From Claim 17 and Claim 18, it immediately follows that for every $w \in \Sigma^*$,

$$\begin{array}{lll}
(S_1s_1w, s_2, \ldots, s_n) \vdash_\Gamma^* (\gamma f_1, f_2 \ldots, f_n) & \text{iff} & S_1\langle s_1s_2 \ldots s_n\rangle w \vdash_M^* \gamma\langle f_1f_2 \ldots f_n\rangle; \\
(S_1s_1w, s_2, \ldots, s_n) \vdash_\Gamma^* (q_1, q_2, \ldots, q_n) & \text{iff} & S_1\langle s_1s_2 \ldots s_n\rangle w \vdash_M^* \langle q_1q_2 \ldots q_n\rangle; \\
(S_1s_1w, s_2, \ldots, s_n) \vdash_\Gamma^* (f_1, f_2, \ldots, f_n) & \text{iff} & S_1\langle s_1s_2 \ldots s_n\rangle w \vdash_M^* \langle f_1f_2 \ldots f_n\rangle;
\end{array}$$

where $\gamma \in \Gamma_1^*$, $q_i \in Q_i$, $f_i \in F_i$, $1 \leq i \leq n$. Therefore, $L(M)_x = L(\Gamma)_x$, and Lemma 16 holds. □

Lemma 19. *Let* $x \in \{f, \varepsilon, f\varepsilon\}$ *and* $n \geq 1$. *For every PDA,* M, *there exists an* SCAS$_n$, $\Gamma = (M_1, M_2, \ldots, M_n, \Psi)$, *where* M_1 *is a PDA, and* M_2 *through* M_n *are FAs, such that* $L(\Gamma)_x = L(M)_x$.

PROOF. Let $x \in \{f, \varepsilon, f\varepsilon\}$. Let $M = (Q, \Sigma, \Gamma_M, R, s, S, F)$ be a PDA. For $n = 1$, $\Gamma = (M, Q)$, and the lemma holds immediately. For some $n \geq 2$, we construct an SCAS_n,

$$\Gamma = (M_1, M_2, \ldots, M_n, \Psi),$$

where M_1 is a PDA, and M_i is an FA, for all $2 \leq i \leq n$, such that $L(\Gamma)_x = L(M)_x$, in the following way.

1. Set $M_1 = M$.

2. For every $2 \leq i \leq n$, set $M_i = (\{s_i\}, \Sigma, \{s_i \to s_i\}, s_i, \{s_i\})$.

3. Set $\Psi = Q\{s_2 s_3 \ldots s_n\}$.

Thus, for every $w \in \Sigma^*$,

$$\begin{array}{lll} (Ssw, s_2, \ldots, s_n) \vdash_\Gamma^* (\gamma f, s_2, \ldots, s_n) & \text{iff} & Ssw \vdash_M^* \gamma f; \\ (Ssw, s_2, \ldots, s_n) \vdash_\Gamma^* (q, s_2, \ldots, s_n) & \text{iff} & Ssw \vdash_M^* q; \\ (Ssw, s_2, \ldots, s_n) \vdash_\Gamma^* (f, s_2, \ldots, s_n) & \text{iff} & Ssw \vdash_M^* f; \end{array}$$

where $\gamma \in \Gamma_M^*$, $q \in Q$, $f \in F$. Therefore, $L(\Gamma)_x = L(M)_x$, and Lemma 19 holds. □

THEOREM 20. *For every $L \in \mathbf{CF}$, there is an SCAS_n Γ containing a PDA and $n-1$ FAs as its components such that $L = L(\Gamma)_x$, where $n \geq 1$ and $x \in \{f, \varepsilon, f\varepsilon\}$.*

PROOF. This theorem follows from Lemma 16 and Lemma 19. □

THEOREM 21. *For every $L \in \mathbf{LIN}$, there is an SCAS_n Γ containing a one-turn PDA and $n-1$ FAs as its components such that $L = L(\Gamma)_x$, where $n \geq 1$ and $x \in \{f, \varepsilon, f\varepsilon\}$.*

PROOF. Let $L \in \mathbf{LIN}$ and $n \geq 1$. Let $x \in \{f, \varepsilon, f\varepsilon\}$. Then, there exists a one-turn PDA, M, such that L is accepted by M. From M, as is demonstrated in Lemma 19, construct an SCAS_n, Γ, such that $L(M)_x = L(\Gamma)_x$. Thus, Γ has only one PDA as its component, which must be a one-turn PDA.

Conversely, let Γ be an SCAS_n with at most one one-turn PDA and arbitrary number of FAs as its components. Let $x \in \{f, \varepsilon, f\varepsilon\}$. By Lemma 16, there exists a PDA, M, such that $L(\Gamma)_x = L(M)_x$. By Definition 1, if only one turn is possible to perform in some PDA component of Γ, then no more than one turn is possible to perform in Γ. Therefore, from the construction of M from Γ in Lemma 16, it follows that M must be a one-turn PDA, which implies that $L(M)_x \in \mathbf{LIN}$. □

COROLLARY 22. *For every $L \in \mathscr{L}(\mathrm{dPDA})_x$, where $x \in \{f, \varepsilon, f\varepsilon\}$, there is an dSCAS_n Γ containing a PDA and $n-1$ FAs as its components such that $L = L(\Gamma)_y$, where $n \geq 1$ and $y \in \{f, \varepsilon, f\varepsilon\}$.*

PROOF. Observe that both Lemma 16 and Lemma 19 also hold for deterministic variants of SCAS_ns and PDAs (see their proofs). □

5. Concluding Remarks

The result stating that state-synchronized automata systems are computational complete is expectable, because Theorem 1 says that they can be simply transformed into well-known computational complete model. Theorem 12 demonstrates that even nondeterministic state-synchronized automata systems with one-turn pushdown automata are computational complete. For the deterministic case, we state the following conjecture, which we have not been able to verify rigorously, however.

CONJECTURE 1. *Let $n \geq 2$. Then, there exists a recursively enumerable language that cannot be accepted by any* dSCAS$_n$ *such that all its components are one-turn PDAs.* □

6. Acknowledgements

This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), the TAČR grant TE01020415, and the BUT grant FIT-S-14-2299.

7. References

[1] Čermák M., Meduna A.; *n-Accepting restricted pushdown automata systems*, in: *13th International Conference on Automata and Formal Languages*, Computer and Automation Research Institute, Hungarian Academy of Sciences, Nyíregyháza, 2011, pp. 168–183.

[2] Chomsky N.; *Three models for the description of language*, IRE Transactions on Information Theory, 2(3), 1956, pp. 113–124.

[3] Csuhaj-Varjú E., Dassow J., Kelemen J., Păun Gh.; *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.

[4] Csuhaj-Varjú E., Martín-Vide C., Mitrana V., Vaszil G.; *Parallel communicating pushdown automata systems*, International Journal of Foundations of Computer Science, 11(4), 2000, pp. 631–650.

[5] Harrison M.A.; *Introduction to Formal Language Theory*, Addison-Wesley, Boston, 1978.

[6] Kučera J.; *On state-synchronized automata systems*, in: Drahanský M., Orság F. (eds.), *Proceedings of the 19th Conference STUDENT EEICT 2013*, Faculty of Information Technology Brno University of Technology, Brno, 2013, pp. 216–218.

[7] Martín-Vide C., Mateescu A., Mitrana V.; *Parallel finite automata systems communicating by states*, International Journal of Foundations of Computer Science, 13(5), 2002, pp. 733–749.

[8] Meduna A.; *Automata and Languages: Theory and Applications*, Springer, London, 2000.

[9] Meduna A.; *Simultaneously one-turn two-pushdown automata*, International Journal of Computer Mathematics, 2003(80), 2003, pp. 679–687.

[10] Meduna A., Lukáš R.; *Multigenerative grammar systems*, Schedae Informaticae, 2006, pp. 175–188.