ON DETERMINISM IN STATE-SYNCHRONIZED AUTOMATA SYSTEMS

Jiří Kučera
Doctoral Degree Programme (1), FIT BUT
E-mail: xkucer28@stud.fit.vutbr.cz

Supervised by: Alexander Meduna
E-mail: meduna@fit.vutbr.cz

Abstract: In this paper is studied the determinism in state-synchronized automata systems of degree n. It will be shown that every recursively enumerable language can be accepted by corresponding deterministic state-synchronized automata system containing at least two pushdown automata.

Keywords: determinism, deterministic state-synchronized automata system, DSCAS

1 INTRODUCTION

From the pragmatic point of view, the studying of determinism in theoretical computer science is crucially important because it is much easier to implement deterministic behaviour than handling the nondeterministic one in real applications. The nondeterministic state-synchronized automata systems were defined and investigated in [1] and it was proved that such the systems containing two or more pushdown automata are Turing complete. Furthermore, Example 4.6 in [1] demonstrates that deterministic variants of these systems are suitable for accepting languages which are not context-free.

In this paper we firstly recall the definition of state-synchronized automata system. Then, the complete proof of coincidence of the family of recursively enumerable languages with the family of languages accepted by deterministic state-synchronized automata systems containing at least two pushdown automata will be given. Finally, the paper is closed by discussion about the practical usage of mentioned systems and also the further investigation is outlined.

2 PRELIMINARIES AND DEFINITIONS

It is assumed that the reader is familiar with the basic notions of formal language theory [2]. 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, by ρ^* is denoted the reflexive and transitive closure of ρ, and by ρ^i is denoted the ith power of ρ, $i \geq 0$. Let w be a word over Σ. Then, the set of all subwords contained in w is denoted by $\mathrm{subword}(w)$. By **RE** we denote the family of recursively enumerable languages.

A *pushdown automaton* (PDA), M, 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 finite set of symbols on pushdown, Q, Σ, and Γ are pairwise disjoint, $R \subseteq (\Gamma \cup \{\varepsilon\}) \times Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \times Q$ is a finite set of rules, $(A, p, a, x, q) \in R \overset{\text{def}}{\Longleftrightarrow} Apa \to xq \in R$, $s \in Q$ is the initial state, $S \in \Gamma \cup \{\varepsilon\}$ is the initial word on pushdown, and $F \subseteq Q$ is a set of final states. In further text, we use $\mathrm{lhs}(r) = u$ and $\mathrm{rhs}(r) = v$ to denote the left-hand side of r and the right-hand side of r for some $r = u \to v \in R$, respectively. 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 $uApaw \vdash_M uxqw$ in M. The language accepted by M, $L(M)$, is defined as $L(M) = \{w \in \Sigma^* \mid Ssw \vdash_M^* f, f \in F\}$. The M is said to be a *finite automaton* (FA) if $\Gamma = \emptyset$.

A *two-pushdown automaton* (2PDA), M, is defined as a septuple $M = (Q, \Sigma, \Gamma, R, s, S, F)$, where Q, Σ, s, and F are defined as in PDA, Γ is a pushdown alphabet, $S \in \Gamma$ is the initial symbol on both pushdowns, and $R \subseteq \Gamma \times \Gamma \times Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \times \Gamma^* \times Q$ is a finite set of rules, $(A, B, p, a, x, y, q) \in R \overset{\text{def}}{\iff} A\#Bpa \to x\#yq \in R$, 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 $uA\#vBpaw \vdash_M ux\#vyqw$ in M. The language accepted by M, $L(M)$, is defined as $L(M) = \{w \in \Sigma^* \mid S\#Ssw \vdash_M^* \#f, f \in F\}$. If for every rule $A\#Bpa \to x\#yq \in R$ holds $\text{card}(\{\alpha \to \gamma \in R \mid \alpha \in \text{subword}(A\#Bpa)\}) = 1$, then M is a *deterministic two-pushdown automaton* (D2PDA). By $\mathscr{L}(\text{2PDA})$ is denoted the family of languages accepted by 2PDA, analogously for D2PDA.

Definition 2.1 (State-synchronized automata system)**.** A *state-synchronized automata system* of degree n ($\text{SCAS}_{(t_1,t_2,\ldots,t_n)}$), Γ, is defined as an $(n+1)$-tuple $\Gamma = (M_1, M_2, \ldots, M_n, \Psi)$, where $M_i = (Q_i, \Sigma, \Gamma_i, R_i, s_i, S_i, F_i)$, called ith *component* of Γ, is a t_i, $t_i \in \{\text{FA}, \text{PDA}\}$, for all $1 \leq i \leq n$, and $\Psi \subseteq Q_1 Q_2 \ldots Q_n$ is a *control language* of Γ. Moreover, $\text{card}\,(\bigcup_{i=1}^n Q_i) = \sum_{i=1}^n \text{card}(Q_i)$. By Ψ_f we denote $\Psi \cup F_1 F_2 \ldots F_n$. 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 $1 \leq i \leq n$. Given a two configurations of Γ, $\alpha = (\chi_1, \chi_2, \ldots, \chi_n)$ and $\alpha' = (\chi_1', \chi_2', \ldots, \chi_n')$, the relation of *direct move* in Γ, $\vdash_\Gamma$, is defined as follows: if for every $1 \leq i \leq n$ holds $\chi_i \vdash_{M_i} \chi_i'$ in M_i, and $\pi_1(\chi_1)\pi_2(\chi_2)\ldots\pi_n(\chi_n) \in \Psi$, $\pi_1(\chi_1')\pi_2(\chi_2')\ldots\pi_n(\chi_n') \in \Psi_f$, where π_i is a homomorphism from $\Gamma_i^* Q_i \Sigma^*$ to Q_i such that for $q_i \in Q_i$ holds $\pi_i(q_i) = q_i$, otherwise $\pi_i(a) = \varepsilon$, for all $1 \leq i \leq n$, then $\alpha \vdash_\Gamma \alpha'$ in Γ. The *language* accepted by Γ, $L(\Gamma)$, is defined as $L(\Gamma) = \{w \in \Sigma^* \mid (S_1 s_1 w, S_2 s_2, \ldots, S_n s_n) \vdash_\Gamma^* (f_1, f_2, \ldots, f_n)\}$. The *family of languages* accepted by $\text{SCAS}_{(t_1,t_2,\ldots,t_n)}$ is denoted by $\mathscr{L}(\text{SCAS}_{(t_1,t_2,\ldots,t_n)})$.

Given some $\text{SCAS}_{(t_1,t_2,\ldots,t_n)}$, Γ, define a set of all mappings from $\{1, 2, \ldots, n\}$ to $\bigcup_{i=1}^n R_i$, $\mathscr{R}_\Gamma$, such that $\alpha \in \mathscr{R}_\Gamma$ iff for all $1 \leq i \leq n$ holds $\alpha(i) \in R_i$. Furthermore, define two other mappings, π_l and π_r, from $\mathscr{R}_\Gamma$ to $Q_1 Q_2 \ldots Q_n$ as follows: $\pi_l(\alpha) = \pi_1(\text{lhs}(\alpha(1)))\pi_2(\text{lhs}(\alpha(2)))\ldots\pi_n(\text{lhs}(\alpha(n)))$ and analogously $\pi_r(\alpha) = \pi_1(\text{rhs}(\alpha(1)))\pi_2(\text{rhs}(\alpha(2)))\ldots\pi_n(\text{rhs}(\alpha(n)))$.

Definition 2.2 (Deterministic SCAS)**.** Let Γ be an $\text{SCAS}_{(t_1,t_2,\ldots,t_n)}$. Then Γ is said to be *deterministic* (DSCAS) if for every mapping $\alpha \in \mathscr{R}_\Gamma$, such that $\pi_l(\alpha) \in \Psi$ and $\pi_r(\alpha) \in \Psi_f$, holds $\text{card}(\{\alpha' \in \mathscr{R}_\Gamma \mid \pi_l(\alpha') \in \Psi, \pi_r(\alpha') \in \Psi_f, \text{lhs}(\alpha'(i)) \in \text{subword}(\text{lhs}(\alpha(i))), 1 \leq i \leq n\}) = 1$. By $\mathscr{L}(\text{DSCAS}_{(t_1,t_2,\ldots,t_n)})$ we denote the family of languages accepted by $\text{DSCAS}_{(t_1,t_2,\ldots,t_n)}$.

3 THEORETICAL RESULTS

Theorem 3.1. $\mathscr{L}(\text{DSCAS}_{(\text{PDA,PDA})}) = \mathbf{RE}$.

Proof of Theorem 3.1. Inclusion $\mathscr{L}(\text{DSCAS}_{(\text{PDA,PDA})}) \subseteq \mathbf{RE}$ follows directly from Church-Turing thesis. To prove that $\mathbf{RE} \subseteq \mathscr{L}(\text{DSCAS}_{(\text{PDA,PDA})})$, we use the result from [2] that $\mathscr{L}(\text{D2PDA}) = \mathbf{RE}$, and present here an algorithm that converts every D2PDA $\hat{M}$ to an equivalent $\text{DSCAS}_{(\text{PDA,PDA})}$ Γ, so $L(\hat{M}) = L(\Gamma)$.

Given D2PDA $\hat{M} = (\hat{Q}, \Sigma, \hat{\Gamma}, \hat{R}, \hat{s}, \hat{S}, \hat{F})$, construct a $\text{DSCAS}_{(\text{PDA,PDA})}$ $\Gamma = (A, B, \Psi)$, where component $A = (Q_A, \Sigma, \hat{\Gamma}, R_A, s, \hat{S}, F_A)$ and component $B = (Q_B, \Sigma, \hat{\Gamma}, R_B, \bar{s}, \hat{S}, F_B)$, such that $L(\hat{M}) = L(\Gamma)$, by the following way:

1. Set $Q_A = \emptyset$, $Q_B = \emptyset$, and $\Psi = \emptyset$.
2. For every rule $r = A\#B\hat{p}a \to x\#y\hat{q} \in \hat{R}$:

- add states p, $\langle r\rangle$, and q to Q_A,
- add states $\bar{p}$, $\langle r\rangle'$, and $\bar{q}$ to Q_B,
- add rules $Apa \to A\langle r\rangle$ and $A\langle r\rangle \to xq$ to R_A,
- add rules $B\bar{p} \to B\langle r\rangle'$ and $B\langle r\rangle' \to y\bar{q}$ to R_B, and
- add words $p\bar{p}$, $q\bar{q}$, and $\langle r\rangle\langle r\rangle'$ to Ψ.

3. Set $F_A = \{f \mid \hat{f} \in \hat{F}\}$ and $F_B = \{\bar{f} \mid \hat{f} \in \hat{F}\}$.

The algorithm above always halts because $\hat{R}$ and $\hat{F}$ are finite sets. To show that $L(\hat{M}) = L(\Gamma)$ we firstly prove the following claim.

Claim 3.2. *If $u_0\#v_0\hat{q}_0w_0$ and $u_i\#v_i\hat{q}_iw_i$ are two configurations of $\hat{M}$, then*

$$u_0\#v_0\hat{q}_0w_0 \vdash_{\hat{M}}^{i} u_i\#v_i\hat{q}_iw_i \quad \text{iff} \quad (u_0q_0w_0, v_0\bar{q}_0) \vdash_{\Gamma}^{2i} (u_iq_iw_i, v_i\bar{q}_i)$$

for all $i \geq 0$.

Proof of Claim 3.2. The proof is established by induction on $i \geq 0$.

Basis. For $i = 0$ we have

$$u_0\#v_0\hat{q}_0w_0 \vdash_{\hat{M}}^{0} u_0\#v_0\hat{q}_0w_0 \quad \text{iff} \quad (u_0q_0w_0, v_0\bar{q}_0) \vdash_{\Gamma}^{0} (u_0q_0w_0, v_0\bar{q}_0)$$

which is obviously true.

Induction hypothesis. Suppose that the claim holds for every $0 \leq i \leq j$, for some $j \geq 0$.

Induction step. From induction hypothesis we know that the claim holds for every $0 \leq i \leq j$, for some $j \geq 0$. For $j+1$ we have

$$\begin{array}{ll}
 & u_0\#v_0\hat{q}_0w_0 \vdash_{\hat{M}} u_1\#v_1\hat{q}_1w_1 \vdash_{\hat{M}}^{j} u_{j+1}\#v_{j+1}\hat{q}_{j+1}w_{j+1} \\
\text{iff} & u_0 = uA, v_0 = vB, w_0 = aw_1, u_1 = ux, v_1 = vy, \\
 & r = A\#B\hat{q}_0a \to x\#y\hat{q}_1 \in \hat{R} \\
\text{iff} & Aq_0a \to A\langle r\rangle, A\langle r\rangle \to xq_1 \in R_A, \\
 & B\bar{q}_0 \to B\langle r\rangle', B\langle r\rangle' \to y\bar{q}_1 \in R_B, \\
 & q_0\bar{q}_0, q_1\bar{q}_1, \langle r\rangle\langle r\rangle' \in \Psi \\
\text{iff} & (u_0q_0w_0, v_0\bar{q}_0) \vdash_{\Gamma} (u_0\langle r\rangle w_1, v_0\langle r\rangle') \vdash_{\Gamma} (u_1q_1w_1, v_1\bar{q}_1) \\
 & \vdash_{\Gamma}^{2j} (u_{j+1}q_{j+1}w_{j+1}, v_{j+1}\bar{q}_{j+1})
\end{array}$$

and therefore the claim holds for all $j \geq 0$. ■

Now it is clear that $\hat{S}\#\hat{S}\hat{s}w \vdash_{\hat{M}}^{i} \#\hat{f}$ iff $(\hat{S}sw, \hat{S}\bar{s}) \vdash_{\Gamma}^{2i} (f, \bar{f})$, and we have $L(\hat{M}) = L(\Gamma)$. Therefore the algorithm is correct and the theorem holds. □

Theorem 3.3. $\mathbf{RE} = \mathscr{L}(\mathrm{DSCAS}_{(t_1,t_2,\ldots,t_n)})$, *where* $t_1, t_2 = \mathrm{PDA}$, $t_i \in \{\mathrm{FA}, \mathrm{PDA}\}$, $3 \leq i \leq n$, $n \geq 3$.

Proof of Theorem 3.3. Inclusion $\supseteq$ follows directly from Church-Turing thesis. We prove the opposite inclusion $\subseteq$ as follows: Given some $\mathrm{DSCAS}_{(\mathrm{PDA},\mathrm{PDA})}$ $\Gamma' = (M_1', M_2', \Psi')$, where M_i' is defined as $M_i' = (Q_i', \Sigma, \hat{\Gamma}, R_i', s_i', \hat{S}, F_i')$, $i \in \{1,2\}$, and some $n \geq 3$, construct a $\mathrm{DSCAS}_{(t_1,t_2,\ldots,t_n)}$ $\Gamma = (M_1, M_2, \ldots, M_n, \Psi)$, where $t_1, t_2 = \mathrm{PDA}$, $t_i \in \{\mathrm{FA}, \mathrm{PDA}\}$, $3 \leq i \leq n$, and

1. $M_i = (Q_i, \Sigma, \Gamma_i, R_i, s_i, S_i, F_i)$, $1 \leq i \leq n$,

2. $Q_i = \{q \mid q' \in Q'_i\}$, $Q_j = \{s_j, f_j\}$, $i \in \{1,2\}$, $3 \le j \le n$,

3. $(\Gamma_i, S_i) = (\hat{\Gamma}, \hat{S})$, $(\Gamma_j, S_j) \in \{(\{\hat{S}\}, \hat{S}), (\emptyset, \varepsilon)\}$, $i \in \{1,2\}$, $3 \le j \le n$,

4. $R_i = \{Apa \to xq \mid Ap'a \to xq' \in R'_i\}$, $R_j = \{S_j s_j \to S_j s_j, S_j s_j \to f_j\}$, $i \in \{1,2\}$, $3 \le j \le n$,

5. $F_i = \{f \mid f' \in F'_i\}$, $F_j = \{f_j\}$, $i \in \{1,2\}$, $3 \le j \le n$,

6. $\Psi = \{q_1 q_2 s_3 \dots s_n \mid q'_1 q'_2 \in \Psi'\}$.

In construction above, we suppose without loss of generality that for every rule $r' \in R'_k$ holds: if $\mathrm{rhs}(r') \in F'_k$, then $\mathrm{lhs}(r') = \hat{S}q'$, $q' \in Q'_k$, $k \in \{1,2\}$. Now, by induction on $i \ge 1$, we prove that

$$(u_0 p'_0 w, v_0 q'_0) \vdash^i_{\Gamma'} (f'_1, f'_2) \quad \text{iff} \quad (u_0 p_0 w, v_0 q_0, S_3 s_3 \dots, S_n s_n) \vdash^i_{\Gamma} (f_1, f_2, \dots, s_n). \tag{1}$$

Basis. For $i = 1$ we have

$$\begin{array}{ll} & (u_0 p'_0 w, v_0 q'_0) \vdash_{\Gamma'} (f'_1, f'_2) \\ \text{iff} & u_0 = A, v_0 = B, w = a, p'_0 q'_0 \in \Psi', \\ & Ap'_0 a \to f'_1 \in R'_1, Bq'_0 \to f'_2 \in R'_2 \\ \text{iff} & Ap_0 a \to f_1 \in R_1, Bq_0 \to f_2 \in R_2, S_i s_i \to f_i \in R_i, 3 \le i \le n, \\ & p_0 q_0 s_3 \dots s_n \in \Psi \\ \text{iff} & (u_0 p_0 w, v_0 q_0, S_3 s_3, \dots, S_n s_n) \vdash_{\Gamma} (f_1, f_2, \dots, f_n) \end{array}$$

and then (1) holds for $i = 1$.

Induction hypothesis. Suppose that (1) holds for every $1 \le i \le j$, for some $j \ge 1$.

Induction step. For $j+1$ we have

$$\begin{array}{ll} & (u_0 A p'_0 a w, v_0 B q'_0) \vdash_{\Gamma'} (u_0 x p'_1 w, v_0 y q'_1) \vdash^j_{\Gamma'} (f'_1, f'_2) \\ \text{iff} & Ap'_0 a \to x p'_1 \in R'_1, Bq'_0 \to y q'_1 \in R'_2, p'_0 q'_0 \in \Psi' \\ \text{iff} & Ap_0 a \to x p_1 \in R_1, Bq_0 \to y q_1 \in R_2, S_i s_i \to S_i s_i \in R_i, 3 \le i \le n, \\ & p_0 q_0 s_3 \dots s_n \in \Psi \\ \text{iff} & (u_0 A p_0 a w, v_0 B q_0, S_3 s_3, \dots, S_n s_n) \vdash_{\Gamma} (u_0 x p_1 w, v_0 y q_1, S_3 s_3, \dots, S_n s_n) \vdash^j_{\Gamma} (f_1, f_2, \dots, f_n) \end{array}$$

and then (1) holds for all $i \ge 1$. Therefore, $(\hat{S}s'_1 w, \hat{S}s'_2) \vdash^*_{\Gamma'} (f'_1, f'_2)$ iff $(S_1 s_1 w, S_2 s_2, \dots, S_n s_n) \vdash^*_{\Gamma} (f_1, f_2, \dots, f_n)$ from which immediately follows $L(\Gamma') = L(\Gamma)$ and the theorem holds. □

Corollary 3.4. $\mathbf{RE} = \mathscr{L}(\mathrm{DSCAS}_{(t_1,t_2,\dots,t_n)})$, *where* $\mathrm{card}(\{i \mid t_i = \mathrm{PDA}, 1 \le i \le n\}) \ge 2$, $n \ge 2$.

Proof of Corollary 3.4. Follows directly from Theorem 3.1 and Theorem 3.3. □

4 EXAMPLE OF APPLICATION

The main advantage of DSCAS is that the all components of DSCAS make their computation steps concurrently and deterministically. Since two cooperating pushdown automata can be used as Turing machine, we can use DSCAS with n pairs of PDAs as a model for the simulation of n processors working concurently. In this section we present a simpler example — a vector instruction simulation.

Let $\Gamma = (M_c, M_1, M_2, \Psi)$ be a $\mathrm{DSCAS}_{(\mathrm{FA,PDA,PDA})}$, where M_c is a *controller*, and M_1 and M_2 are *processing units*. Here, Γ represents a simulator of the vector instruction for increment and works with words over alphabet $\{0,1\}$. The definitions of M_1, M_2 and M_c can be determined from the following table:

R_c	R_1	R_2	Ψ
$s0 \to q_1$	$Xs \to Xq_1$	$Xs \to Xq_1$	sss
$q_1 1 \to q_1$	$Xq_1 \to X1q_1$	$Xq_1 \to Xq_1$	$q_1q_1q_1$
$q_1 0 \to q_2$	$Xq_1 \to Xq_2$	$Xq_1 \to Xq_2$	
$q_2 1 \to q_2$	$Xq_2 \to Xq_2$	$Xq_2 \to X1q_2$	$q_2q_2q_2$
$q_2 0 \to q_x$	$Xq_2 \to X1q_x$	$Xq_2 \to X1q_x$	
$q_x \to e_0$	$Xq_x \to Xe_0$	$Xq_x \to Xe_0$	$q_xq_xq_x$
$e_0 \to e_0$	$1e_0 \to e_0$	$Xe_0 \to Xe_0$	$e_0e_0e_0$
$e_0 \to e_1$	$Se_0 \to Se_1$	$Xe_0 \to Xe_1$	
$e_1 \to e_1$	$Xe_1 \to Xe_1$	$1e_1 \to e_1$	$e_1e_1e_1$
$e_1 \to f$	$Se_1 \to f$	$Se_1 \to f$	

where $X \in \{S, 1\}$.

Since DSCASs are accepting devices, we need a way how to obtain a result of simulation. For this purpose we define a homomorphism from R_c^* to $\{0,1\}^*$, h, as follows: $h(p \to q) = 0$ if $p \neq q$, $p, q \in Q_c$, $h(p \to p) = 1$ if $p \in Q_c$, and $h(r) = \varepsilon$ for other rules r from R_c. With this extension, DSCAS can work as transducer. The process of parallel incrementation is demonstrated below:

$$
\begin{array}{lllll}
 & (s0010, Ss, Ss) & & & \\
\vdash_\Gamma & (q_1010, Sq_1, Sq_1) & & & \\
\vdash_\Gamma & (q_210, Sq_2, Sq_2) & & & \\
\vdash_\Gamma & (q_20, Sq_2, S1q_2) & & & \\
\vdash_\Gamma & (q_x, S1q_x, S11q_x) & & & \\
\vdash_\Gamma & (e_0, S1e_0, S11e_0) & [q_x \to e_0] & \approx & 0 \\
\vdash_\Gamma & (e_0, Se_0, S11e_0) & [e_0 \to e_0] & \approx & 1 \\
\vdash_\Gamma & (e_1, Se_1, S11e_1) & [e_0 \to e_1] & \approx & 0 \\
\vdash_\Gamma & (e_1, Se_1, S1e_1) & [e_1 \to e_1] & \approx & 1 \\
\vdash_\Gamma & (e_1, Se_1, Se_1) & [e_1 \to e_1] & \approx & 1 \\
\vdash_\Gamma & (f, f, f) & [e_1 \to f] & \approx & 0
\end{array}
$$

Clearly, $\tau_\Gamma = \{(01^x01^y0, 01^{x+1}01^{y+1}0) \mid x \geq 0, y \geq 0\}$ is the translation defined by DSCAS Γ together with homomorphism h.

5 CONCLUSION

In this paper was proved that deterministic state-synchronized automata systems with at least two pushdown automata as their components are Turing complete. Also, the example of application of DSCAS as a formal model for parallel computation was given. As the topic for further investigation is planned to study how to restrict presented systems to get a new language families or language families from the area of regulated grammars.

Acknowledgement: This work was supported by Research and application of advanced methods in ICT (FIT-S-14-2299).

REFERENCES

[1] KUČERA, Jiří. *A Combination of Automata and Grammars.* Brno (Czech Republic), 2013. Master's thesis. Brno University of Technology, Faculty of Information Technology, Department of Information Systems.

[2] MEDUNA, Alexander. *Automata and Languages: Theory and Applications.* Springer, 2000. ISBN 81-8128-333-3.