ON STATE-SYNCHRONIZED AUTOMATA SYSTEMS

Jiří Kučera
Master Degree Programme (3), FIT BUT
E-mail: xkucer28@stud.fit.vutbr.cz

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

Abstract: This paper introduces a new system of formal models, a state-synchronized automata system of degree n. The computation in presented system is controlled by control words from control language, where control word is a sequence of states. Furthermore, this paper shows that for every recursively enumerable language there exists an equivalent state-synchronized automata system consisting of two or more pushdown automata.

Keywords: pushdown automata, automata system, state-synchronized automata system, SCAS, controlled computation, language properties

1 INTRODUCTION

The main motivation in systems of formal models approach is in parallelism and computation distribution, which are popular topics in modern computer science. Several systems of formal models are studied in [2, 1]. This paper introduces new automata systems such that these systems characterize the family of recursively enumerable languages.

2 PRELIMINARIES AND DEFINITIONS

In this paper, it is assumed that the reader is familiar with the basics of formal language theory [3]. 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 $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. By **RE** is denoted the family of recursively enumerable languages.

A *pushdown automaton*, 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 *pushdown alphabet*, Q, Σ and Γ are pairwise disjoint, $R \subseteq \Gamma Q(\Sigma \cup \{\varepsilon\}) \times \Gamma^* Q$ is a finite set of *rules*, $Apa \to xq \in R \stackrel{\text{def}}{\iff} (Apa, xq) \in R$, $s \in Q$ is the *initial state*, $S \in \Gamma$ is the *initial symbol* on pushdown and $F \subseteq Q$ is a set of *final states*. Let Φ be an alphabet of *rule labels*, $\mathrm{card}(\Phi) = \mathrm{card}(R)$, and let ϕ be a bijection from R to Φ, $\phi(Apa \to xq) = r \stackrel{\text{def}}{\iff} r : Apa \to xq$, $Apa \to xq \in R \wedge \phi(Apa \to xq) = r \stackrel{\text{def}}{\iff} r : Apa \to xq \in R$. Let χ be a word from $\Gamma^* Q \Sigma^*$. Then, χ is a *configuration* of M. If $u \in \Gamma^*$, $w \in \Sigma^*$ and $r : Apa \to xq \in R$, then $uApaw \vdash_M uxqw[r]$. Let $\vdash_M^*$ denote the transitive and reflexive closure of $\vdash_M$. Then, $L_X(M) = \{w \in \Sigma^* \mid Ssw \vdash_M^* \gamma q\}$ is a *langauge* accepted by M by *empty pushdown*, if $X = \varepsilon$, $\gamma = \varepsilon$ and $q \in Q$, or by *final state*, if $X = f$, $\gamma \in \Gamma^*$ and $q \in F$. Furthermore, $L(M) = L_\varepsilon(M) \cap L_f(M)$ denotes the language accepted by M by empty pushdown and final state. The definition of a *finite automaton*, M', and its terminology are very similar to the definitions associated with pushdown automaton M above, with the difference, that all parts related to the pushdown are omitted. So, $M' = (Q', \Sigma', R', s', F')$ and $L(M') = \{w \in \Sigma'^* \mid s'w \vdash_{M'}^* f, f \in F'\}$. For brevity, the indicies of M are inherited by its components, so if $\hat{M}_x$ denotes some pushdown automaton, then $\hat{M}_x = (\hat{Q}_x, \hat{\Sigma}_x, \hat{\Gamma}_x, \hat{R}_x, \hat{s}_x, \hat{S}_x, \hat{F}_x)$. Furthermore, let FA and PDA be an abbrevations for finite automaton and pushdown automaton, respectively.

The *state-synchronized automata system* of degree n ($\mathrm{SCAS}_{(m_1,m_2,\ldots,m_n)}$), Γ, is an $(n+1)$-tuple, $\Gamma = (M_1, M_2, \ldots, M_n, \Psi)$, where M_i, the ith *component* of Γ, is PDA or FA, for every $i \in \{1,2,\ldots,n\}$, $n \geq 1$, and $\Psi \subseteq Q_1 Q_2 \ldots Q_n$ is said to be a *control language*. Furthermore, the $m_i \in \{\mathrm{FA}, \mathrm{PDA}\}$, for every $i \in \{1,2,\ldots,n\}$, denotes the type of ith component of Γ. In the rest of document, let ${}_?\Gamma_i$ denote the pushdown alphabet of the ith component of Γ, if $m_i = \mathrm{PDA}$, otherwise ${}_?\Gamma_i = \emptyset$. A *configuration* of Γ is an n-tuple $(\chi_1, \chi_2, \ldots, \chi_n)$, where $\chi_i \in {}_?\Gamma_i^* Q_i \Sigma_i^*$, for every $i \in \{1,2,\ldots,n\}$. Define $\mathrm{statew}(X) \in Q_1 Q_2 \ldots Q_n$ as $\mathrm{statew}((\alpha_1 q_1 \beta_1, \alpha_2 q_2 \beta_2, \ldots, \alpha_n q_n \beta_n)) = q_1 q_2 \ldots q_n$, where $\alpha_i \in {}_?\Gamma_i^*$ and $\beta_i \in \Sigma_i^*$, for every $i \in \{1,2,\ldots,n\}$. Let $\Psi_f = \Psi \cup F_1 F_2 \ldots F_n$. If $\chi = (\chi_1, \chi_2, \ldots, \chi_n)$ and $\chi' = (\chi'_1, \chi'_2, \ldots, \chi'_n)$ are two configurations of Γ, $\mathrm{statew}(\chi) \in \Psi$, $\mathrm{statew}(\chi') \in \Psi_f$, and $\chi_i \vdash_{M_i} \chi'_i$, for every $i \in \{1,2,\ldots,n\}$, then Γ makes a *computation step* from χ to χ', written as $\chi \vdash_\Gamma \chi'$. Let $\vdash_\Gamma^*$ denote the transitive and reflexive closure of $\vdash_\Gamma$. The *language* accepted by Γ is defined as $L(\Gamma) = \{w \in \Sigma_1 \mid ({}_?S_1 s_1 w, {}_?S_2 s_2, \ldots, {}_?S_n s_n) \vdash_\Gamma^* (f_1, f_2, \ldots, f_n)\}$, where ${}_?S_i = S_i$, if ith component of Γ is PDA, otherwise ${}_?S_i = \varepsilon$, and $f_i \in F_i$, for every $i \in \{1,2,\ldots,n\}$. The SCAS, Γ, is said to be *deterministic*, denoted as dSCAS, iff the $\vdash_\Gamma$ has the properties of the function. The family of languages accepted by $\mathrm{SCAS}_{(m_1,m_2,\ldots,m_n)}$ is denoted by $\mathscr{L}(\mathrm{SCAS}_{(m_1,m_2,\ldots,m_n)})$.

Example 2.1: Let $\Gamma = (M, \hat{M}, \Psi)$ be a $\mathrm{dSCAS}_{(\mathrm{PDA},\mathrm{PDA})}$, where

$$
\begin{array}{rcl}
M & = & (\{s, q_1, q_2, q_3, f\}, \{a,b,c\}, \{S,a,b,c\}, R, s, S, \{f\}), \\
\hat{M} & = & (\{\hat{s}, \hat{q_1}, \hat{q_2}, \hat{q_3}, \hat{f}\}, \{a,b,c\}, \{\hat{S}, \hat{a}, \hat{b}, \hat{c}\}, \hat{R}, \hat{s}, \hat{S}, \{\hat{f}\}), \\
R & = & \{Ssa \to Saq_1, aq_1 a \to aaq_1, aq_1 b \to aq_2, aq_2 b \to aq_2, aq_2 c \to q_3, aq_3 c \to q_3, Sq_3 \to f\}, \\
\hat{R} & = & \{\hat{S}\hat{s} \to \hat{S}\hat{q_1}, \hat{S}\hat{q_1} \to \hat{S}\hat{q_1}, \hat{S}\hat{q_1} \to \hat{S}\hat{b}\hat{q_2}, \hat{b}\hat{q_2} \to \hat{b}\hat{b}\hat{q_2}, \hat{b}\hat{q_2} \to \hat{q_3}, \hat{b}\hat{q_3} \to \hat{q_3}, \hat{S}\hat{q_3} \to \hat{f}\}, \\
\Psi & = & \{s\hat{s}, q_1\hat{q_1}, q_2\hat{q_2}, q_3\hat{q_3}\}.
\end{array}
$$

It is easy to see, that $L(\Gamma) = \{a^n b^n c^n \mid n \geq 1\}$, which is not a context-free language. The word $aaabbbccc \in L(\Gamma)$ is accepted in this way:

$$
\begin{array}{lllll}
 & (Ssaaabbbccc, \hat{S}\hat{s}) & & & \\
\vdash_\Gamma & (Saq_1 aabbbccc, \hat{S}\hat{q_1}) & \vdash_\Gamma (Saaq_1 abbbccc, \hat{S}\hat{q_1}) & \vdash_\Gamma & (Saaaq_1 bbbccc, \hat{S}\hat{q_1}) \\
\vdash_\Gamma & (Saaaq_2 bbccc, \hat{S}\hat{b}\hat{q_2}) & \vdash_\Gamma (Saaaq_2 bccc, \hat{S}\hat{b}\hat{b}\hat{q_2}) & \vdash_\Gamma & (Saaaq_2 ccc, \hat{S}\hat{b}\hat{b}\hat{b}\hat{q_2}) \\
\vdash_\Gamma & (Saaq_3 cc, \hat{S}\hat{b}\hat{b}\hat{q_3}) & \vdash_\Gamma (Saq_3 c, \hat{S}\hat{b}\hat{q_3}) & \vdash_\Gamma & (Sq_3, \hat{S}\hat{q_3}) \\
\vdash_\Gamma & (f, \hat{f}) & & &
\end{array}
$$

3 RESULTS

Theorem 3.1: *For every $L \in$ **RE**, there exists* $\mathrm{SCAS}_{(\mathrm{PDA},\mathrm{PDA})}$, Γ, *such that $L(\Gamma) = L$.*

Proof of Theorem 3.1. From Theorem 10.3.1 in [3] is clear, that every recursively enumerable language, L, can be expressed as $L = h(L(M_A) \cap L(M_B))$, where M_A and M_B are deterministic pushdown automata and h is a homomorphism. Let M_A and M_B be deterministic PDA. Without loss of generality, assume that $\Sigma_A = \Sigma_B$ and $\Gamma_A = \Gamma_B$. Let $\Gamma = (M_{A'}, M_{B'}, \Psi)$ be a $\mathrm{SCAS}_{(\mathrm{PDA},\mathrm{PDA})}$ and h be a homomorphism from Σ_A^* to $\Sigma_{A'}^*$, $a \in \Sigma_A$ implies $h(a) \in \Sigma_{A'} \cup \{\varepsilon\}$, where $M_{A'}$, $M_{B'}$ and Ψ are constructed from M_A and M_B in the following way:

1. $Q_X \subseteq Q_{X'}$, $F_X = F_{X'}$, $\Sigma_{X'} = \{h(a) \mid a \in \Sigma_X \wedge h(a) \neq \varepsilon\}$ and $\Gamma_{X'} = \Gamma_X$, where $X \in \{A, B\}$,
2. $s_{X'} \coloneqq s_X$ and $S_{X'} \coloneqq S_X$, where $X \in \{A, B\}$,
3. $\forall p \in Q_A \forall q \in Q_B : pq \in \Psi$,
4. $\forall p \in Q_X \forall Z \in \Gamma_X : Zp \to Zp \in R_{X'}$, where $X \in \{A, B\}$,

5. $\forall Ap \to xq \in R_X : Ap \to xq \in R_{X'}$, where $X \in \{A,B\}$,
6. $\forall a \in \Sigma_A \forall (Ap_0a \to xp_1, Bq_0a \to yq_1) \in R_A \times R_B$:
   - add new state $\langle xp_1 \rangle$ to $Q_{A'}$, add new state $\langle yq_1 \rangle$ to $Q_{B'}$,
   - add new rules $Ap_0h(a) \to A\langle xp_1 \rangle$ and $A\langle xp_1 \rangle \to xp_1$ to $R_{A'}$,
   - add new rules $Bq_0 \to B\langle yq_1 \rangle$ and $B\langle yq_1 \rangle \to yq_1$ to $R_{B'}$,
   - add new control word $\langle xp_1 \rangle \langle yq_1 \rangle$ to Ψ.

Thus, for every $L \in \mathbf{RE}$, there exists $\mathrm{SCAS}_{(\mathrm{PDA},\mathrm{PDA})}$, Γ, such that $L(\Gamma) = L$. □

Corollary 3.2: $\mathscr{L}(\mathrm{SCAS}_{(\mathrm{PDA},\mathrm{PDA})}) = \mathbf{RE}$.

Proof of Corollary 3.2. $\mathbf{RE} \subseteq \mathscr{L}(\mathrm{SCAS}_{(\mathrm{PDA},\mathrm{PDA})})$ follows from Theorem 3.1, $\mathscr{L}(\mathrm{SCAS}_{(\mathrm{PDA},\mathrm{PDA})}) \subseteq \mathbf{RE}$ follows from Church-Turing thesis. □

Theorem 3.3: *For every* $L \in \mathbf{RE}$*, there exists* $\mathrm{SCAS}_{(\mathrm{PDA},\mathrm{PDA},m_1,m_2,\ldots,m_n)}$, Γ, *where* $m_i \in \{\mathrm{FA},\mathrm{PDA}\}$, *for every* $i \in \{1,2,\ldots,n\}$, $n \geq 1$, *such that* $L(\Gamma) = L$.

Proof of Theorem 3.3. The automata $m_1, m_2, \ldots, m_n$ are constructed each with one state, which is initial and final at the same time, and one epsilon rule, representing a loop over that state. The control language additionally contains such control words that when first two pushdown automata make a computation step, then the rest automata make an ε-move from initial state to initial state. Thus, these automata are redundant and the language acceptance depends only on first two pushdown automata. □

Corollary 3.4: $\mathscr{L}(\mathrm{SCAS}_{(\mathrm{PDA},\mathrm{PDA},m_1,m_2,\ldots,m_n)}) = \mathbf{RE}$, *where* $m_i \in \{\mathrm{FA},\mathrm{PDA}\}$, $1 \leq i \leq n$, $n \geq 1$.

Proof of Corollary 3.4. Direct follows from Theorem 3.3 and Church-Turing thesis. □

4 CONCLUSION

In this paper a new type of automata system was presented. It was shown that the presented automata system with at least two pushdown automata as its components can accept any recursively enumerable language. The rigorous proofs are omitted due to the requirements on the length of this paper. The future investigation of presented automata system is to study if the deterministic SCAS has the same computation power as a nondeterministic SCAS. Also is planned to study other ways of communication, with possibly restrictions. The presented automata system can be used for example in parallel compiling, system modeling or linguistics.

REFERENCES

[1] ČERMÁK, Martin. *Formal systems based upon automata and grammars.* Brno, 2012. PhD thesis. Brno University of Technology, Faculty of Information Technology, Department of Information Systems.

[2] CSUHAJ-VARJÚ, E., DASSOW, J., KELEMEN, J. and PĂUN, G. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation.* Singapore: Gordon and Breach Science Publishers, 1994. ISBN 2-88124-957-4.

[3] HARRISON, Michael A. *Introduction to Formal Language Theory.* Boston (MA, USA): Addison-Wesley Longman Publishing, 1978. ISBN 0201029553.