Are there Universal Finite or Pushdown Automata?

Manfred Kudlek , Patrick Totzke , Georg Zetzsche
Department Informatik, MIN-Fakultät,
Universität Hamburg, DE
email: {kudlek,3totzke,3zetzsch}@informatik.uni-hamburg.de

Abstract

We investigate the (non)-existence of universal automata for various classes of automata, as finite and pushdown automata, and in particular the influence of the representation and encoding function. An alternative approach, using transition systems, is presented too.

1 Introduction

It is well known that there exist universal Turing machines (UTM). Such a UTM simulates any special Turing machine (TM) M in a certain way. There are several ways of simulation. One is that a UTM U simulating a TM M with input w halts if and only if M halts on input w. Another possibility is that any computation step of M is simulated by U using some number of steps which might be restricted by some complexity function.

Almost all UTM's constructed so far are deterministic, simulating deterministic TM's. In [4] it has been shown that there exist UTM's simulating all special TM's with complexity constraints. These complexity constraints, for space or time, are from a subclass of all primitive recursive functions over one variable. The UTM's have the same complexity constraints.

In both cases, general TM's and those with complexity constraint, the specific TM M and its input $w \in \Sigma(M)^*$, where $\Sigma(M)$ is the alphabet of M, have to be encoded. Such an encoding, and also the decoding, can be achieved by deterministic finite state transducers (DFST), which means that

encoding and decoding is bijective. The input for a UTM U, to simulate M with input w, can then have the form $c_m(M)\#c_i(w)$ where c_M, c_i are the encoding functions for M, w, respectively.

If one intends to construct universal machines for weaker automata classes it should be kept in mind that encoding and decoding for such automata should not exceed the power of deterministic versions of those machines. Otherwise too much power and information could be hidden in the encoding.

In [3] it has been shown, under this condition, that there don't exist universal 1-way finite automata (FA), neither DFA nor NFA. The proof uses arguments on the number of states of such automata.

So the question arises whether there exist universal PDA, and if yes if encoding and decoding can be achieved by DFST's, or if deterministic pushdown transducers (DPDT) are necessary.

2 Transducers

Pushdown transducers (see e.g. [1, 2]) are just the analogon to finite state transducers, i.e. pushdown automata with output.

Formally:

A *(non-deterministic) pushdown transducer (PDT)* is a construct
$(Q, \Sigma_i, \Sigma_o, \Delta, \$, q_0, Q_f, \rho)$ where
$\rho \subseteq Q \times \Sigma_i^* \times \Sigma_o^* \times \Delta^* \times \Delta^* \times Q$

A *deterministic pushdown transducer (DPDT)* is the deterministic version of PDT. That is, ρ is a function.

In the sequel a normal form of PDT will be considered, being quasi lettering in input and pushdown, i.e.
$\rho \subseteq Q \times (\Sigma_i \cup \{\lambda\}) \times (\Sigma_o \cup \{\lambda\}) \times (\Delta \cup \{\lambda\}) \times (\Delta \cup \{\lambda\}) \times Q$.

Contrary to regular languages (**REG**) which are closed under finite state transductions the context-free languages (**CF**) are not closed under pushdown transductions, non-deterministic as well as deterministic. Even linear context-free languages give sets outside **CF** if a deterministic pushdown transduction is applied. This is shown by the following examples of DPDT's.

Let $L_1 = \{0^n 1 \mid n \geq 0\}$ and
$T_1 = (\{q_0, q_1\}, \{0, 1\}, \{0, 1\}, \{q_0\}, \{q_1\}, \rho_1)$ with

$\rho_1 = \{(q_0, 0, 0, \$, 0\$, q_0), (q_0, 0, 0, 0, 00, q_0), (q_0, 1, \lambda, \$, \$, q_2),$
$(q_0, 1, \lambda, 0, 0, q_1), (q_1, \lambda, 1, 0, \lambda, q_1), (q_1, \lambda, \lambda, \$, \$, q_2)\}$.
Then $\tau(L_1) = \{0^n 1^n \mid n \geq 0\} \notin \mathbf{REG}$.

Let $L_2 = \{0^n 1^n 0 \mid n > 0\}$ and
$T_2 = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{0, 1\}, \{q_0\}, \{q_3\}, \rho_2)$ with
$\rho_2 = \{(q_0, 0, 0, \$, 0\$, q_0), (q_0, 0, 0, 0, 00, q_0), (q_0, 1, 1, 0, 10, q_1),$
$(q_1, 1, 1, 1, 11, q_1), (q_1, 0, \lambda, 1, 1, q_2), (q_2, \lambda, 0, 1, \lambda, q_2),$
$(q_2, \lambda, 1, 0, \lambda, q_2), (q_2, \lambda, \lambda, \$, \$, q_3)\}$.
Then $\tau(L_2) = \{0^n 1^n 0^n 1^n \mid n \geq 0\} \notin \mathbf{CF}$.

Let $L_3 = \{wcw^R c \mid w \in \{0, 1\}^+\}$ and
$T_3 = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{0, 1\}, \{q_0\}, \{q_3\}, \rho_3)$ with
$\rho_3 = \{(q_0, 0, 0, \$, 0\$, q_0), (q_0, 1, 1, \$, 1\$, q_0), (q_0, 0, 0, 0, 00, q_0),$
$(q_0, 0, 0, 1, 01, q_0), (q_0, 1, 1, 0, 10, q_0), (q_0, 1, 1, 1, 11, q_0),$
$(q_0, c, c, 0, 0, q_1), (q_0, c, c, 1, 1, q_1), (q_1, 0, 0, \$, 0\$, q_1),$
$(q_1, 1, 1, \$, 1\$, q_1), (q_1, 0, 0, 0, 00, q_1), (q_1, 0, 0, 1, 01, q_1),$
$(q_1, 1, 1, 0, 10, q_1), (q_1, 1, 1, 1, 11, q_1), (q_1, c, c, 0, 0, q_2),$
$(q_1, c, c, 1, 1, q_2), (q_2, \lambda, \lambda, 0, 0, q_2), (q_2, \lambda, \lambda, 1, 1, q_2),$
$(q_2, \lambda, \lambda, c, c, q_2), (q_2, \lambda, \lambda, \$, \$, q_3)\}$.
Then $\tau(L_3) = \{wcw^R cwcw^R c \mid w \in \{0, 1\}^+\} \notin \mathbf{CF}$.

One also might consider 2-way FST (2FST). However, **REG** is not closed under 2-way finite state transductions, as can be seen from the following example of a 2DFST.
Let $L_4 = \{awb \mid w \in \{0, 1\}^*\}$ and
$T_4 = (\{q_0, q_1, q_2, q_3\}, \{0, 1, a, b\}, \{0, 1, a\}, \{q_0\}, \{q_4\}, \rho_2)$ with
$\rho_4 = \{(q_0, a, a, R, q_0), (q_0, 0, 0, R, q_0), (q_0, 1, 1, R, q_0), (q_0, b, \lambda, L, q_1),$
$(q_1, 0, \lambda, L, q_1), (q_1, 1, \lambda, L, q_1), (q_1, a, a, R, q_2),$
$(q_2, 0, 0, R, q_2), (q_2, 1, 1, R, q_2), (q_2, b, \lambda, M, q_3)$.
Then $\tau(L_4) = \{awaw \mid w \in \{0, 1\}^*\} \notin \mathbf{CF}$.

3 Representations

A representation of FA, PDA, TM's etc. has to contain information on the set of states, initial and final states, alphabets, and the set of transitions:
$(Q, \Sigma, \Delta, Q_0, Q_f, R)$ with e.g. $R \subseteq Q \times \Sigma^* \times \Delta^* \times \Delta^* \times Q$ for a PDA.

Usually R is represented by an ordered list of elements from R, together with a lists for Q_0 and Q_f. For non-deterministic machines one might also allow repetitions of list elements. This can give a regular set of representations if Q, Σ, and Δ are fixed.

FA and PDA usually are represented by the list R of their transitions, putting together the tuples (q, x, q') or (q, x, y, y', q') for FA or PDA respectively, where $x \in (\Sigma_i \cup \{\lambda\})$, and $y, y' \in (\Delta \cup \{\lambda\})$.

For FA one has a representation $R \in (Q \cdot (\Sigma_i \cup \{*\}) \cdot Q)^+$, and for PDA $R \in (Q \cdot (\Sigma_i \cup \{*\}) \cdot (\Delta \cup \{*\}) \cdot (\Delta \cup \{*\}) \cdot Q)^+$ where $*$ stands for λ. Together with the input w this gives a representation $R(M)\#w$. But one might think also of a representation $w\#R(M)$ or even $R(M) \sqcup\!\sqcup \# \sqcup\!\sqcup w$ where $\sqcup\!\sqcup$ is the shuffle operation.

If the first version of representation is encoded by a DFST T the result is a word $\tau(R(M))\tau(\#)\tau(w)$ where τ is the function associated to T. This follows from the fact that T is working 1-way. The same conditions hold for the 'inverse', namely for a DPDT T' with associated function τ'. Furthermore,

$\tau'(\tau(R(M))\tau(\#)\tau(w)) = R(M)\#w$.

Considering representations of FA with arbitrary sets of states Q and arbitrary alphabets Σ, states q_k can be by represented qa^k $(0 < k)$ and symbols x_m by xa^m $(0 < m)$ over the finite alphabet $\{q, x, a\}$. Then the set of all representations of finite automata is given by

$F = (\{q\} \cdot \{a\}^+ \cdot (\{x\} \cdot \{a\}^+ \cup \{*\}) \cdot \{q\} \cdot \{a\}^+)^+$.

Clearly, $F \in \mathbf{REG}$. But note that this holds only for the non-deterministic FA. In the case of DFA there is the condition that a state q_k, represented by qa^k, can appear only once as first component in the list of transitions.

An analogous property also holds for PDA.

Since **REG** and **CF** are closed under FST mappings, encoding (and decoding) will not lead out from these classes. However, the examples above show that this does not hold for PDT mappings. Therefore, not to gain too much power it might be reasonable to have the condition that those PDT mappings used for encoding and decoding are not leading out from the class **CF**.

4 Universality?

In the sequel we shall consider PDA M accepting languages $L(M) \subseteq \{0,1\}^*$. Assume that there exists a universal PDA U simulating all specific PDA M over $\{0,1\}$. Denote this class by $\mathbf{CF}_2$. Let Σ_U the alphabet of U. Then, with the first version of representation,

$L(U) = \{\tau(R(M))\tau(\#)\tau(w) \mid L(M) \in \mathbf{CF}_2, w \in L(M)\} \in \mathbf{CF}$.

Consider now special regular (context-free) languages $\{w\} \subset \{0,1\}^*$. A representation of a DFA M, being also a PDA and 2DFST, accepting exactly the language $\{w\}$, e.g. looks like $R(M) = q_0wq_1 = \varphi(w)$. Together with an input $v \in \{0,1\}^*$ the representation has the form $R(M)\#v = q_0wq_1\#v$. A DFST maps this into $\tau(\varphi(w))\tau(\#)\tau(v)$. Clearly,

$S = \{\tau(\varphi(w))\tau(\#)\tau(v) \mid v, w \in \{0,1\}^*\}$

$= \{\tau(\varphi(w)) \mid w \in \{0,1\}^*\} \cdot \{\tau(\#)\} \cdot \tau(\{0,1\}^*) \in \mathbf{REG}$.

Now

$L(U) \cap S = \{\tau(\varphi(w))\tau\#\tau(v) \mid w, v \in \{0,1\}^*, v = w\}$

$= \{\tau(\varphi(w))\tau(\#)\tau(w) \mid w \in \{0,1\}^*\} \in \mathbf{CF}$

since $\mathbf{CF}$ is closed under intersection with regular sets.

Applying the 'inverse' DFST mapping τ' yields

$\tau'(L(U) \cap S) = \{\varphi(w)\#w \mid w \in \{0,1\}^*\} \in \mathbf{CF}$.

Another DFST mapping ψ with $\psi(q_0) = \psi(q_1) = \lambda$, $\psi(0) = 0$, $\psi(1) = 1$ gives

$\psi(\tau'(L(U) \cap S)) = \{w\#w \mid w \in \{0,1\}^*\} \in \mathbf{CF}$,

a contradiction.

These considerations can be summarized as

Theorem 1: If encoding and decoding of specific finite or pushdown automata have to be achieved by DFST then there doesn't exist a universal finite automaton, or 2-way finite automaton or pushdown automaton, simulating all specific finite automata. □

It should be remarked, however, that the proof of this theorem cannot be used to show that the statement also holds for all quasi lettering finite automata. The reason is that the DFST for encoding has to know the length of w for the representation $q_0x_1q_1 \cdots x_kq_k$ where $x_i \in \{0,1\}$ and all q_i are different.

Therefore we give another proof that this theorem also holds for quasi-lettering automata. For non-deterministic (quasi-lettering) FA it can be as-

sumed that there is exactly one initial and exactly one final state, and that the first element in the list has the form (q_1, x, q) where q_1 is the initial state and $x \in \Sigma$, and that the last element has the form (q_2, λ, q_2) where q_2 is the final state.

Let $M_\mu = (Q, \{0,1\}, q_1, \{q_2\}, R_\mu)$ a FA where
$\mu = (i, i', i'', j, j', j'', k, k', k'', \ell, \ell', \ell'')$ and
$2 < i, i', i'', j, j', j'', k, k', k'', \ell, \ell', \ell'' \leq |Q|$ are fixed and
$R(M_\mu) = \{(q_1, 0, q_i), (q_1, 1, q_k), (q_{i'}, 0, q_{j'}), (q_{i''}, 1, q_{\ell''}),$
$(q_{k''}, 0, q_{j''}), (q_{k'}, 1, q_{\ell'}), (q_j, 0, q_2), (q_\ell, 1, q_2), (q_2, \lambda, q_2)\}$.

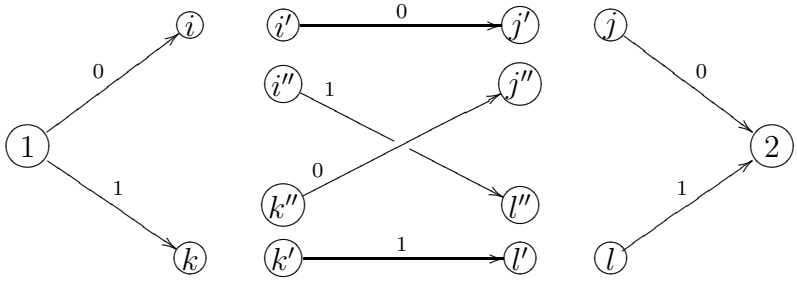

Figure 1: FA

$L(M_\mu) = \{000, 011, 100, 111\}$ implies $i = i' = i''$, $j = j' = j''$, $k = k' = k''$, $\ell = \ell' = \ell''$. Note that M_μ is also a PDA.

An encoding of $R(M_\mu)$ is given e.g. by
$\tau(R(M_\mu)) = qaxbqa^i \cdot qaxcqa^k \cdot qa^{i'}xbqa^{j'} \cdot qa^{i''}xcqa^{\ell''} \cdot$
$qa^{k''}xbqa^{j''} \cdot qa^{k'}xcqa^{\ell'} \cdot qa^jxbqa^2 \cdot qa^\ell xcqa^2 \cdot qa^2 * qa^2$
τ is a DFST mapping. Note that τ actually depends on Q.

Now assume that
$U = \{\tau(R(M))\tau(\#) \mid M \in \text{QLFA}, \{000, 011, 100, 111\} \subseteq L(M)\}$
$\cdot\tau(\{000, 011, 100, 111\})$
is context-free, where QLFA denotes the class of quasi-lettering FA with set of states Q, initial state q_1, final state q_2, and alphabet $\{0,1\}$.

Define $S = \{\tau(R(M_\mu))\tau(\#) \mid 2 < i, i', i'', j, j', j'', k, k', k'', \ell, \ell', \ell'' \leq |Q|\}$
$\cdot\tau(\{000, 011, 100, 111\}$.
Obviously, S is regular, implying that $U \cap S$ is context-free. But
$U \cap S = \{qaxbqa^i \cdot qaxcq^k \cdot qa^ixba^\ell \cdot q^ixcqa^\ell \cdot qa^kxbqa^j \cdot qa^kxcqa^\ell \cdot qa^jxbqa^2$
$\cdot qa^\ell xcqa^2 \cdot qa^2 * qa^2 \mid 2 < i, j, k, \ell \leq |Q|\}$
is not context-free, a contradiction.

If 2-way pushdown transducers are allowed however, and the encoding is not required to have the form $\tau(R(M)\tau(\#)\tau(w)$ it is possible to construct a universal PDA, following an idea by Gh. Păun et. al.. To be more precise, there exists a quasi-lettering universal PDA simulating all lettering PDA.

Let $M = (Q, \Sigma, \Delta, \delta, q_0, \{q_f\})$ with $\Sigma = \{0, 1\}$ be a lettering PDA, and $R(M) \subseteq \{Q \times \Sigma \times \Delta \times \Delta \times Q\}^*$ be its representation. Suppose we allow $\tau(R(A)\#w)$ to be transduced by a 2-way PDT (2PDT). Then one can choose the coding $\tau(R(M)\#x_1x_2 \dots x_n) = x_1\sigma(R(M))x_2\sigma(R(M)) \dots x_n\%\sigma(R(M))$ where $x_i \in \Sigma$, and $\sigma(R(M))$ is a coding of $R(M)$ in a fixed alphabet. Note that σ depends on Q. For simplicities sake let $\sigma = id$ at first. Then τ can be calculated by a 2PDT that works as follows:

1. First go right until the place after # and print the first symbol of w.
2. Go left and push $a \in \Delta$ to the stack until reading #.
3. Go left to the beginning of the input.
4. Print $R(M)$ until reading the #.
5. Go further right and simultaniously pop $a \in \Delta$ from the stack. On empty stack check if the end of the input has been reached.
   (a) If so, print '%', go to the beginning of the input and copy $R(M)$ one last time and halt.
   (b) Otherwise print the current input symbol and repead from step 2.

Consider the following PDA U with $\Sigma_U = \{0, 1\} \cup (Q \times \Sigma \times \Delta \times \Delta \times \Sigma)$ and $\Delta_U = \{a\} \cup Q$ for all possible Q, Σ, Δ. The Σ_U, Δ_U are not finite, but when chosen an appropiate encoding σ for the representation of simulated automata, the following idea for an UPDA works with finite alphabets Σ_U, Δ_U. For now, assume any transition (q, x, a, b, q') in the simulated automaton to be an atomic symbol of our UPDA.

At any time U's stack consists of a word $q\delta_m\delta_{m-1} \dots \delta_0$ where q is the current state of M and $\delta = \delta_m \dots \delta_0$ is the stack content of M during a simulation. In state R, U first reads a symbol x_i from the input (word w) and afterwards checks whether the simulated PDA could have read x_i by travelling through the input $R(M)$ and looking for a transition $(q, x_i, \delta_r, \delta_w, q')$ with

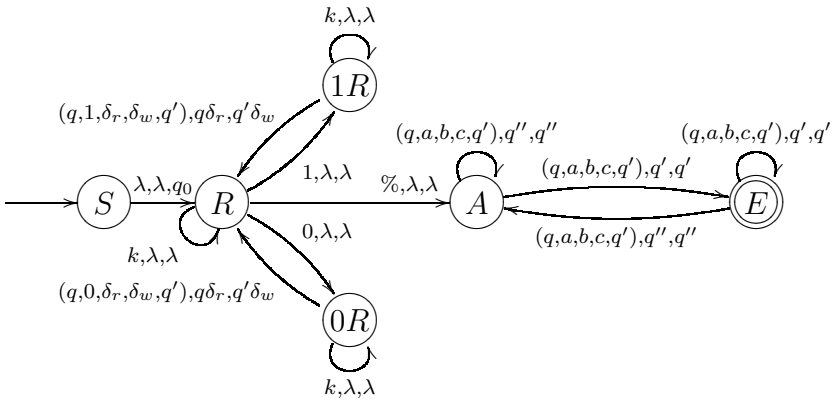

Figure 2: Universal PDA

q being the state M is currently in, and storing the new state and changed stack content of M in its own stack.

In the construction we have used (q, x, a, b, q') etc. as one symbol, but it works also if an encoding σ over a finite alphabet Σ_U is used. Such an encoding for example is given by $\sigma((q_i, x, \delta_j, \delta_k, q_\ell)) = DSA^ixPB^jPB^kSA^\ell$ with $x \in \Sigma = \{0, 1\}$. Then, point 4 in the transduction τ would contain additional steps to encode a transition. E.g. to read symbol $q_i \in Q$ and print Sa^i and so on. Also, the UPDA U must have additional components for decoding in all states that have outgoing transitions reading a tuple (q, x, a, b, q') from the input.

This can be achieved as follows. Let $\Sigma_U = \{A, B, D, P, S, T, 0, 1\}$ where D, P, S, T are markers. D is the deliminator of $DSA^iyPB^jPB^kSA^\ell$ encoding $(q_i, y, \delta_j, \delta_k, q_\ell)$, and T the deliminator of a block $\sigma(R(M))$.

In a step U, after reading $x \in \{0, 1\}$, goes into different states according to x. The stack of U contains $SA^mPB^nPB^r \cdots \$$ where \$ is the bottom symbol. U non-determistically goes to some $DSA^iyPB^jPB^kSA^\ell$ within $\sigma(R(M))$. If $i = m$, $x = y$, and $j = n$ then $SA^mPB^n \cdots \$$ in the stack is replaced by $SA^\ell PB^k \cdots \$$. Otherwise, U goes into a sink. Note that U is a quasi-lettering PDA.

With a slight modification in the encoding it can be shown that the result also holds for PDA M not being lettering for the pushdown alphabet.

5 Transition Systems

In this section we present an alternative approach to investigate universal automata.

In order to specify what we mean by 'universal PDA' or 'universal grammar', we introduce the notion of *Transition Systems*, a generalisation of finite automata that models any kind of computational device having an internal *state* that can be altered by the occurrence of an *action* during the course of a computation.

Definition 1: (Transition System)

A *Transition System* is a quintuple $(S, \Sigma, \delta, S_0, S_F)$ with

S	a set of states,
Σ	an alphabet of transitions
$\delta \subseteq S \times (\Sigma \cup \{\lambda\}) \times Q$	a transition relation
$S_0 \subseteq S$	a set of initial states
$S_F \subseteq S$	a set of final states

We write $s \xrightarrow{t} s'$ for $(s, t, s') \in \delta$. A transition system is called *finite* if $S \cup T$ is. The transition relation in a transition system can be extended to finite sequences of transitions:

- $s \xrightarrow{\lambda} s$ for all states s and the empty sequence λ.
- $s \xrightarrow{wt} s'$ iff a state s'' exists, such that $s \xrightarrow{w} s'' \wedge s'' \xrightarrow{t} s'$

Let $\xrightarrow{*}$ denote the transitive and reflexive closure of $\longrightarrow$.

The *language* of the transition system A is

$$L(A) = \{w \in \Sigma^* | s_0 \xrightarrow{w} s_f, s_0 \in S_0, s_f \in S_F\}.$$

Any finite automaton $A = (Q, \Sigma, \delta, q_0, Q_F)$ is also a transition system by definition. However, in a transition system Q and Q_F are generally not finite. Any PDA $A = (Q, \Sigma, \Delta, \delta, q_0, Q_F)$ defines a transition system whose states are all possible configurations $c \in Q \times \Delta^*$ of the PDA, and transitions are defined by $(q, wx) \xrightarrow{a} (q', wy) \iff (q, a, x, y, q') \in \delta$.

Definition 2: (Universal Transition System)

Let $\mathbf{X} \subseteq \mathbf{RE}$ be a language class below $\mathbf{RE}$. The TS $A = (S, \Sigma, \delta, S_0, S_F)$ is called $\mathbf{X}$-*universal* iff for any language $L \in \Sigma^*$ in $\mathbf{X}$ there is a state $s_L \in S$ such that $L((S, \Sigma, \delta, s_L, S_F)) = L$.

Any computing device that unambiguously defines a transition system will be considered **X**-universal if its transition system is.

Obviously there *is* a **REG**-universal system, namely the disjoint union of all the possible NFA. The interesting question is however, whether or not such a **REG**-universal system can be defined by an NFA. The following lemma recovers a theorem from [3] from a new perspective.

Lemma 1: There is no **REG**-universal NFA.
Proof : For arbitrary $n \in \mathbb{N}, a \in \Sigma$ $L_n = \{a^n\} \in \mathbf{REG}$. Any **REG**-universal system U must have a state s_{L_n} from that on exactly n steps can be made. If the set of states in U was finite, this state could not exist for $n > |S|$, so U must have an infinite set of states. Since any NFA defines a finite transition system, no **REG**-universal system can be defined by an NFA, and therefore no **REG**-universal NFA exists. □

References

[1] Ginsburg, S.: *The Mathematical Theory of Context-free Languages.* McGraw-Hill, 1966.

[2] Gurari, E. M.: *An Introduction to the Theory of Computation.* Computer Science Press, Rockville, 1989.

[3] Kudlek, M.: *On Universal Finite Automata and a-Transducers.* (In: *Grammars and Automata for String Processing: from Mathematics and Computer Science to Biology and back.* Eds. C. Martín Vide, V. Mitrana. *Topics in Computer Mathematics*, pp. 163-170, Taylor and Francis, London, 2003.)

[4] Kudlek M., Margenstern, M.: *Universal Turing Machines with Complexity Constraints.* Proc. Intern. Conf. *Automata and Formal Languages VIII*, Publ. Math. Debrecen **53**, pp. 895-904, 1999.