DEEP PUSHDOWN TRANSDUCERS AND PARALLEL DEEP PUSHDOWN TRANSDUCERS

Peter Solár
Doctoral Degree Programme (4), FIT BUT
E-mail: xsolar05@stud.fit.vutbr.cz

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

Abstract: This paper presents two variants of deep pushdown transducers as extended versions of pushdown transducers. The first - *deep pushdown transducers* are based on *deep pushdown automata*. These transducers can expand non-input pushdown symbols deeper in a pushdown.

The second variant - *parallel deep pushdown transducers* are based on my previous work - *parallel deep pushdown automata*. The main difference is that parallel deep pushdown transducer can expand n topmost non-input pushdown symbols in only one move between two configurations.

Keywords: parsing, pushdown automata, deep pushdown automata, parallel deep pushdown automata, state grammars, pushdown transducers, deep pushdown transducers, parallel deep pushdown transducers

1 INTRODUCTION

If we talk about formal language theory, we usually focus on two core models - grammar and automaton. In a grammar we aim to generate a string belonging to the language using production rules. On the other hand, an automaton is supposed to run on some given input sequence and tries to decide if this input sequence is a string belonging to the language recognized by the automaton. But there exists one another formal model called transducer. Simply said, transducers are automata which are enhanced by the possibility to produce some output string while trying to accept input string.

This paper presents two variants of deep pushdown transducers as extended versions of pushdown transducers - *deep pushdown transducers* based on deep pushdown automata (Meduna, 2006, see [7]) and their parallel version *parallel deep pushdown transducers* based on parallel deep pushdown automata (Solár, 2012, see [8]). Deep pushdown automata has the possibility to expand a non-input pushdown symbol deeper in the pushdown, not only on the pushdown top. Parallel deep pushdown automata are able to expand at most n topmost non-input pushdown symbols in a one move.

2 PRELIMINARIES

This paper assumes that the reader is familiar with the theory of automata and formal languages (see [1], [2], [3] and [6]).

$\mathbb{N}^+$ denotes the set of all positive integers. For an alphabet, Σ, Σ^* represents the free monoid generated by Σ under the operation of concatenation. The identity of Σ^* is denoted by ε. Set $\Sigma^+ = \Sigma^* - \{\varepsilon\}$; algebraically, Σ^+ is thus the free semigroup generated by Σ under the operation of concatenation. For a string $w \in \Sigma^*$, $|w|$ denotes the length of string w. For $W \subseteq \Sigma$, $occur(w,W)$ denotes the number of occurrences of symbols from W in w. $alph(w)$ denotes the set of symbols occurring in string w.

A *pushdown transducer* (see [4]) is a 8-tuple $M^T = (Q, \Sigma_I, \Gamma, \Sigma_O, R, s, S, F)$, where Q is a finite set of the states, Σ_I is an input alphabet, Γ is a pushdown alphabet, Σ_O is an output alphabet, $\Sigma_I \subseteq \Gamma$, $s \in Q$ is

the start state, $S \in \Gamma$ is the start pushdown symbol, $F \subseteq Q$ is a set of finite states, Γ and Σ_O are pairwise disjoint. $R \subseteq Q \times (\Sigma_I \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q \times \Gamma^* \times (\Sigma_O \cup \{\varepsilon\})$. Instead of $(p,a,A,q,w,a') \in R$, where $p,q \in Q$, $a \in \Sigma_I$, $A \in \Gamma - \Sigma_I$, $w \in \Gamma^*$, $a' \in \Sigma_O$, we usually write $r = paA \to qwa'$ $(r \in R)$ and call r a *rule*.

A *state grammar* (see [5]) is a 4-tuple $G = (V,W,T,P,S)$, where V is a total alphabet, W is a finite set of states, $T \subseteq V$ is an alphabet of terminals, $S \in (V-T)$ is the start symbol, and $P \in (W \times (V-T)) \times (W \times V+)$ is a finite relation. Instead of $(q,A,p,v) \in P$, we write $(q,A) \to (p,v) \in P$ throughout.

For every $z \in V^*$, set ${}_G states(z) = \{q \mid (q,B) \to (p,v) \in P$, where $B \in (V-T) \cap alph(z)$, $v \in V^+$, $q,p \in W\}$ If $(q,A) \to (p,v) \in P$, $x,y \in V^*$, ${}_G states(x) \cap \{q\} = \varnothing$, then G makes a derivation step from (q,xAy) to (p,xvy), symbolically written as $(q,xAy) \Rightarrow (p,xvy)[(q,A) \to (p,v)]$ in G; in addition, if n is a positive integer satisfying $occur(xA,V-T) \leq n$, we say that $(q,xAy) \Rightarrow (p,xvy)$ $[(q,A) \to (p,v)]$ is n-limited, symbolically written as (q,xAy) ${}_n\Rightarrow$ (p,xvy) $[(q,A) \to (p,v)]$. Usually if there is no possibility of confusion, we simplify $(q,xAy) \Rightarrow (p,xvy)$ $[(q,A) \to (p,v)]$ to $(q,xAy) \Rightarrow (p,xvy)$ and (q,xAy) ${}_n\Rightarrow$ (p,xvy) $[(q,A) \to (p,v)]$ to (q,xAy) ${}_n\Rightarrow$ (p,xvy). In the standard manner, we extend $\Rightarrow$ to $\Rightarrow^m$, $m \geq 0$. Based on $\Rightarrow^m$ we can define $\Rightarrow^+$ and $\Rightarrow^*$. Let $n \in \mathbb{N}^+$ and $\alpha,\beta \in (W \times V)$. To express that every derivation step in $\alpha \Rightarrow^m \beta$, $\alpha \Rightarrow^+ \beta$ and $\alpha \Rightarrow^* \beta$ is n-limited, we write $\alpha_n\Rightarrow^m \beta$, $\alpha_n\Rightarrow^+ \beta$ and $\alpha_n\Rightarrow^* \beta$. The language of G, $L(G)$, is defined as $L(G) = \{w \in T^* \mid (q,S) \Rightarrow^* (p,w)$, $q,p \in W\}$. Also, we define for every $n \geq 1$, $L(G,n) = \{w \in T^* \mid (q,S)$ ${}_n\Rightarrow^*$ (p,w), $q,p \in W\}$. A derivation of the form (q,S) ${}_n\Rightarrow^*$ (p,w), where $q,p \in W$ and $w \in T^*$, represents a *successful n-limited generation* of w in G.

A *deep pushdown automaton* (see [7]) is a 7-tuple ${}_dM = (Q, \Sigma, \Gamma, R, s, S, F)$, where d is a maximum depth at which can be non-input symbol expanded, Q is a finite set of the states, Σ is an input alphabet, Γ is a pushdown alphabet, $\Sigma \subseteq \Gamma$, $\Gamma - \Sigma$ contains a special bottom symbol denoted by #, $s \in Q$ is the start state, $S \in \Gamma$ is the start pushdown state, $F \subseteq Q$ is a set of finite states. Sets $\mathbb{N}$, Q and Γ are pairwise disjoint. $R \subseteq (\mathbb{N}^+ \times Q \times (\Gamma - (\Sigma \cup \{\#\})) \times Q \times (\Gamma - \{\#\})^+) \bigcup (\mathbb{N}^+ \times Q \times \{\#\} \times Q \times (\Gamma - \{\#\})^* \{\#\})$. Instead of $(m,p,A,q,w) \in R$, where $m \leq d$, $p,q \in Q$, $A \in \Gamma - \Sigma$, $w \in \Gamma^+$, we usually write $r = mpA \to qw$ and call r a *rule*.

A *configuration* of the deep pushdown automaton ${}_dM$ is a triple $Q \times \Sigma^* \times (\Gamma - \{\#\})^* \{\#\}$. Let χ be a set of all configurations of automaton ${}_dM$ and let $x,y \in \chi$ be two configurations. $x \vdash y$ is a move between these two configuration. If $x = (p,au,az), y = (q,u,z)$, where $p,q \in Q, a \in \Sigma, u \in \Sigma^*, z \in \Gamma^*$, then ${}_dM$ *pops* its pushdown from x to y, $x\,{}_p\vdash y$. ${}_dM$ *expands* its pushdown if $x = (p,au,wAz), y = (q,au,wvz), r = p(A) \vdash q(v) \in R$, accordingly to the rule r, symbolically $x\,{}_e\vdash y\,[p(A) \to q(v)]$ or $x\,{}_e\vdash y$ if there is only one usable rule. In the standard manner we can extend ${}_p\vdash$, ${}_e\vdash$ and $\vdash$ to ${}_p\vdash^m$, ${}_e\vdash^m$ and $\vdash^m$, respectively, for $m \geq 0$. Then based on ${}_p\vdash^m$, ${}_e\vdash^m$ and $\vdash^m$, define ${}_p\vdash^+$, ${}_p\vdash^*$, ${}_e\vdash^+$, ${}_e\vdash^*$, $\vdash^+$ and $\vdash^*$.

Let ${}_dM$ be of maximal depth $d \in \mathbb{N}$. We define a *language accepted by* ${}_dM$ as $L({}_dM) = \{w \in \Sigma^* : (s,w,S\#) \vdash^* (f,\varepsilon,\#) \in {}_dM$ with $f \in F\}$, *language accepted by* ${}_dM$ *by empty pushdown* as $L_{emptyPD}({}_dM) = \{w \in \Sigma^* : (s,w,S\#) \vdash^* (q,\varepsilon,\#) \in {}_dM$ with $q \in Q\}$ and *language accepted by* ${}_dM$ *by entering final state* as $L_{final}({}_dM) = \{w \in \Sigma^* : (s,w,S\#) \vdash^* (f,\varepsilon,v\#) \in {}_dM$ with $f \in F, v \in \Gamma^*\}$.

For every state grammar G, and for every $n \geq 1$, there exists a deep pushdown automaton of depth n, ${}_nM$, such that $L(G,n) = L({}_nM)$.

For every $n \geq 1$ and every parallel deep pushdown automaton, ${}_nM$, there exist a state grammar G, which generates language accepted by ${}_nM$, $L(G,n) = L({}_nM)$.

A *parallel deep pushdown automaton* (see [8]) is a 7-tuple ${}_dM_p = (Q,\Sigma,\Gamma,R,s,S,F)$, where d is a maximum depth at which can be non-input symbol expanded, Q is a finite set of the states, Σ is an input alphabet, Γ is an pushdown alphabet, $\Sigma \subseteq \Gamma$, $\Gamma - \Sigma$ contains a special bottom symbol denoted by #, $s \in Q$ is the start state, $S \in \Gamma$ is the start pushdown state, $F \subseteq Q$ is a set of finite states. Sets $\mathbb{N}$,

Q and Γ are pairwise disjoint.

$R \subseteq Q \times \big((\Gamma - (\Sigma \cup \{\#\}))_1 \times (\Gamma - (\Sigma \cup \{\#\}))_2 \times \ldots \times (\Gamma - (\Sigma \cup \{\#\}))_n\big) \times Q \times \big(((\Gamma - \{\#\})^+)_1 \times ((\Gamma - \{\#\})^+)_2 \times \ldots \times ((\Gamma - \{\#\})^+)_n\big) \bigcup Q \times \big((\Gamma - (\Sigma \cup \{\#\}))_1 \times (\Gamma - (\Sigma \cup \{\#\}))_2 \times \ldots \times (\Gamma - (\Sigma \cup \{\#\}))_{n-1} \times (\{\#\})_n\big) \times Q \times \big(((\Gamma - \{\#\})^+)_1 \times ((\Gamma - \{\#\})^+)_2 \times \ldots \times ((\Gamma - \{\#\})^+)_{n-1} \times (\{\#\})_n\big)$

Instead of $\big(p,(A_1,\ldots,A_n),q,(w_1,\ldots,w_n)\big) \in R$, where $n \leq d$, $p,q \in Q$, $A_i \in \Gamma - (\Sigma \cup \{\#\})$, $w_i \in (\Gamma - \{\#\})^+$, $1 \leq i < n$, $\big(A_n \in \Gamma - (\Sigma \cup \{\#\}) \wedge w_n \in (\Gamma - \{\#\})^+\big) \vee \big(A_n = \{\#\} \wedge w_n = \{\#\}\big)$, we usually write $r = p(A_1,\ldots,A_n) \to q(w_1,\ldots,w_n)$ and call r a *rule*.

For every state grammar G, and for every $n \geq 1$, there exists a deep pushdown automaton of depth n, ${}_nM_p$, such that $L(G,n) = L({}_nM_p)$.

For every $n \geq 1$ and every parallel deep pushdown automaton, ${}_nM_p$, there exist a state grammar G, which generates language accepted by ${}_nM_p$, $L(G,n) = L({}_nM_p)$.

3 DEFINITIONS

In this section we look at the definitions of presented deep pushdown transducers.

3.1 DEEP PUSHDOWN TRANSDUCER

The deep pushdown transducer differs from the standard pushdown transducer by possibility to expand non-input pushdown symbols deeper in the pushdown.

A *deep pushdown transducer* is a 8-tuple ${}_dM^T = (Q, \Sigma_I, \Gamma, \Sigma_O, R, s, S, F)$, where d is a maximum depth at which can be non-input symbol expanded, Q is a finite set of the states, Σ_I is an input alphabet, Γ is an pushdown alphabet, $\Sigma_I \subseteq \Gamma$, Σ_O is an output alphabet, $\Sigma_O \not\subseteq \Gamma$, $\Gamma - \Sigma_I$ contains a special bottom symbol denoted by #, $s \in Q$ is the start state, $S \in \Gamma$ is the start pushdown symbol, $F \subseteq Q$ is a set of finite states. Sets $\mathbb{N}$, Q, Γ and Σ_O are pairwise disjoint.

$R \subseteq (\mathbb{N}^+ \times Q \times (\Gamma - (\Sigma_I \cup \{\#\})) \times Q \times (\Gamma - \{\#\})^+ \times (\Sigma_O \cup \{\varepsilon\})) \bigcup (\mathbb{N}^+ \times Q \times (\Gamma - (\Sigma_I \cup \{\#\})) \times Q \times (\Gamma - \{\#\})^* \times \Sigma_O) \bigcup (\mathbb{N}^+ \times Q \times \{\#\} \times Q \times ((\Gamma - \{\#\})^*\{\#\}) \times (\Sigma_O \cup \{\varepsilon\})) \bigcup (\mathbb{N}^+ \times Q \times \{\#\} \times Q \times (((\Gamma - \{\#\})^*\{\#\}) \cup \{\varepsilon\}) \times \Sigma_O)$. Instead of $(m,p,A,q,w,a') \in R$, where $m \leq d$, $p,q \in Q$, $A \in \Gamma - \Sigma_I$, $w \in \Gamma^+$, $a' \in \Sigma_O \cup \{\varepsilon\}$, we usually write $r = mpA \to qwa'$ and call r a *rule*.

A *configuration* of the deep pushdown transducer ${}_dM^T$ is a 4-tuple $Q \times (\Sigma_I)^* \times (\Gamma - \{\#\})^*\{\#\} \times (\Sigma_O)^*$. Let χ be a set of all configurations of deep pushdown transducer ${}_dM^T$ and let $x,y \in \chi$ be two configurations. $x \vdash y$ is a move between these configuration. If $x = (p,au,az,b), y = (q,u,z,bc)$, where $p,q \in Q, a \in \Sigma_I, u \in (\Sigma_I)^*, z \in \Gamma^*, b,c \in (\Sigma_O)^*$, then ${}_dM^T$ *pops* its pushdown from x to y, $x\ {}_p\vdash y$. ${}_dM^T$ *expands* its pushdown if $x = (p,au,wAz,b), y = (q,au,wvz,bc), r = p(A) \to q(v)c \in R$, accordingly to the rule r, symbolically written $x\ {}_e\vdash y[p(A) \to q(v)c]$ or $x\ {}_e\vdash y$ if there is only one usable rule. In the standard manner we can extend ${}_p\vdash$, ${}_e\vdash$ and $\vdash$ to ${}_p\vdash^m$, ${}_e\vdash^m$ and $\vdash^m$, respectively, for $m \geq 0$. Then based on ${}_p\vdash^m$, ${}_e\vdash^m$ and $\vdash^m$, define ${}_p\vdash^+$, ${}_p\vdash^*$, ${}_e\vdash^+$, ${}_e\vdash^*$, $\vdash^+$ and $\vdash^*$.

We can define a *language accepted by* ${}_dM^T$ as $L^{Acc}({}_dM^T) = \{w \in (\Sigma_I)^* : (s,w,S\#,\varepsilon) \vdash^* (f,\varepsilon,\#,u) \in {}_dM^T \text{ with } f \in F, u \in (\Sigma_O)^*\}$ and a *language generated by* ${}_dM^T$ as $L^{Gen}({}_dM^T) = \{u \in (\Sigma_O)^* : (s,w,S\#,\varepsilon) \vdash^* (f,\varepsilon,\#,u) \in {}_dM^T \text{ with } f \in F, w \in (\Sigma_I)^*\}$. In the same manner we can define a *language accepted by* ${}_dM^T$ *by empty pushdown* as $L^{Acc}_{emptyPD}({}_dM^T) = \{w \in (\Sigma_I)^* : (s,w,S\#,\varepsilon) \vdash^* (q,\varepsilon,\#,u) \in {}_dM^T \text{ with } u \in (\Sigma_O)^*\}$, a *language generated by* ${}_dM^T$ *by empty pushdown* as $L^{Gen}_{emptyPD}({}_dM^T) = \{u \in (\Sigma_O)^* : (s,w,S\#,\varepsilon) \vdash^* (q,\varepsilon,\#,u) \in {}_dM^T \text{ with } w \in (\Sigma_I)^*\}$, a *language accepted by* ${}_dM^T$ *by entering the final state* as $L^{Acc}_{final}({}_dM^T) = \{w \in (\Sigma_I)^* : (s,w,S\#,\varepsilon) \vdash^* (f,\varepsilon,v\#,u) \in {}_dM^T \text{ with } f \in F, v \in \Gamma^*, u \in (\Sigma_O)^*\}$ and a *language generated by* ${}_dM^T$ *by entering the final state* as $L^{Gen}_{final}({}_dM^T) = \{u \in (\Sigma_O)^* : (s,w,S\#,\varepsilon) \vdash^* (f,\varepsilon,v\#,u) \in {}_dM^T \text{ with } f \in F, v \in \Gamma^*, w \in (\Sigma_I)^*\}$.

3.2 PARALLEL DEEP PUSHDOWN TRANSDUCER

The parallel deep pushdown transducer differs from a deep pushdown transducer by possibility to expand several topmost non-input pushdown symbols in one move. This is achieved by the form of rules. Each rule is a composition of simpler rules. The first non-input pushdown symbol in the rule corresponds to the topmost non-input symbol on the pushdown, the second non-input pushdown symbol corresponds to the second topmost non-input symbol on the pushdown, etc. up to the maximum depth of this rule which is less or equal to the maximum possible depth specific to this transducer. The numbering of non-input symbols on the pushdown coincides with the state before applying of the rule. The rule can be used only if there is a sufficient count of non-input symbols on the pushdown and their arrangement is identical with arrangement of these symbols in the rule.

A *parallel deep pushdown transducer* is a 8-tuple ${}_dM_p^T = (Q, \Sigma_I, \Gamma, \Sigma_O, R, s, S, F)$, where d is a maximum depth at which can be non-input symbol expanded, Q is a finite set of the states, Σ_I is an input alphabet, Γ is an pushdown alphabet, $\Sigma_I \subseteq \Gamma$, Σ_O is an output alphabet, $\Sigma_O \not\subseteq \Gamma$, $\Gamma - \Sigma_I$ contains a special bottom symbol denoted by #, $s \in Q$ is the start state, $S \in \Gamma$ is the start pushdown symbol, $F \subseteq Q$ is a set of finite states. Sets $\mathbb{N}$, Q, Γ and Σ_O are pairwise disjoint.

$R \subseteq Q \times \big((\Gamma - (\Sigma_I \cup \{\#\}))_1 \times (\Gamma - (\Sigma_I \cup \{\#\}))_2 \times \ldots \times (\Gamma - (\Sigma_I \cup \{\#\}))_n\big) \times Q \times \big(((\Gamma - \{\#\})^+)_1 \times ((\Gamma - \{\#\})^+)_2 \times \ldots \times ((\Gamma - \{\#\})^+)_n\big) \times (\Sigma_O)^* \bigcup Q \times \big((\Gamma - (\Sigma_I \cup \{\#\}))_1 \times (\Gamma - (\Sigma_I \cup \{\#\}))_2 \times \ldots \times (\Gamma - (\Sigma_I \cup \{\#\}))_{n-1} \times (\{\#\})_n\big) \times Q \times \big(((\Gamma - \{\#\})^+)_1 \times ((\Gamma - \{\#\})^+)_2 \times \ldots \times ((\Gamma - \{\#\})^+)_{n-1} \times (\{\#\})_n\big) \times (\Sigma_O)^*$

Instead of $(p,(A_1,\ldots,A_n),q,(w_1,\ldots,w_n),a') \in R$, where $n \leq d$, $p,q \in Q$, $A_i \in \Gamma - (\Sigma_I \cup \{\#\})$, $w_i \in (\Gamma - \{\#\})^+$, $1 \leq i < n$, $\big(A_n \in \Gamma - (\Sigma_I \cup \{\#\}) \wedge w_n \in (\Gamma - \{\#\})^+\big) \vee \big(A_n = \{\#\} \wedge w_n = \{\#\}\big)$, $a' \in (\Sigma_O)^*$, we usually write $r = p(A_1,\ldots,A_n) \rightarrow q(w_1,\ldots,w_n)a'$ and call r a *rule*.

A *configuration* of the parallel deep pushdown transducer ${}_dM_p^T$ is a 4-tuple $Q \times (\Sigma_I)^* \times (\Gamma - \{\#\})^*\{\#\} \times (\Sigma_O)^*$. A *move* between two configurations, *sequence of moves* and definitions of *languages accepted* or *generated by parallel deep pushdown transducer* are the same as at deep pushdown transducers - see section 3.1.

4 RESULTS

4.1 ACCEPTED LANGUAGE

Theorem 1 *For every $n \leq 1$ and for every language L, $L = L(G,n)$ for a state grammar, G, if and only if $L = L^{Acc}({}_nM^T)$.*

Theorem 2 *For every $n \leq 1$ and for every language L, $L = L(G,n)$ for a state grammar, G, if and only if $L = L^{Acc}_{emptyPD}({}_dM^T)$.*

Theorem 3 *For every $n \leq 1$ and for every language L, $L = L(G,n)$ for a state grammar, G, if and only if $L = L^{Acc}({}_nM_p^T)$.*

Theorem 4 *For every $n \leq 1$ and for every language L, $L = L(G,n)$ for a state grammar, G, if and only if $L = L^{Acc}_{emptyPD}({}_dM_p^T)$.*
Proof These four Theorems follows from Theorems 1 and 2 in [7], Theorems 1 and 2 in [8] and from the fact that automata are transducers whose output components are ignored.

5 CONCLUSION

In this paper two new extensions of pushdown transducers were presented. These new transducers work exactly as automata they are based on. They can expand non-input pushdown symbols deeper

in the pushdown. According to the maximal depth used in rules we can get the same infinite hierarchy of languages between context-free and context-sensitive languages as is defined by state grammars.

5.1 OPEN PROBLEMS

5.1.1 DETERMINISM

This paper has discussed two versions of deep pushdown transducers and parallel deep pushdown transducers which work nondeterministically. The future investigation of these transducers should pay a special attention to their deterministic versions, which fulfill a crucial role in practice.

5.1.2 GENERATED LANGUAGE

Another future investigation should be paid to the classification of languages generated by deep pushdown transducer and parallel deep pushdown transducer.

5.1.3 GENERALIZATION

Throughout this paper, we considered only true pushdown expansions when non-input pushdown symbol is replaced with nonempty string. What is the language family defined by deep pushdown transducer or parallel deep pushdown transducer generalized in the sense of replacing pushdown symbols with an empty string?

ACKNOWLEDGEMENT

This work was partially supported by the BUT FIT grant FIT-S-11-2 and the research plan MSM 0021630528.

REFERENCES

[1] Aho, A. V., Ullman, J. D.: *The Theory of Parsing, Translation and Compiling, Volume I: Parsing*, Prentice Hall, Englewood Clifs, New Jersey, 1972, ISBN 0139145567

[2] Autebert, J., Berstel, J., Boasson, L.: Context-free languages and pushdown automata. In: Rozenberg, G., Salomaa, A., (eds.) *Handbook of Formal Languages*, vol. 1. Springer, 1997, ISBN 978-3540604204

[3] Dassow, J., Paŭn, G.: *Regulated Rewriting in Formal Language Theory.* AkademieVerlag, Berlin, 1989, ISBN 978-0387514147

[4] Gurari, E.: *An Introduction to the Theory of Computation*, Computer Science Press, 1989, ISBN 0-7167-8182-4

[5] Kasai, T.: An hierarchy between context-free and context-sensitive languages. In: *Journal of Computer and System Sciences* vol. 4, pp. 492–508, 1970, ISSN 0022-0000

[6] Meduna, A.: *Automata and Languages: Theory and Applications.* Springer, London, 2000, ISBN 978-1852330743

[7] Meduna, A.: Deep Pushdown Automata. In: *Acta informatica*, vol. 98, pp. 114–124, 2006, ISSN 0001-5903

[8] Solár, P.: Parallel Deep Pushdown Automata. In: *Proceedings of the 18th Conference STUDENT EEICT 2012*, pp. 410-414, VUT v Brně, Brno, 2012, ISBN 978-80-214-4462-1