PARALLEL DEEP PUSHDOWN AUTOMATA

Peter Solár
Doctoral Degree Programme (3), FIT BUT
E-mail: xsolar05@stud.fit.vutbr.cz

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

Abstract: This paper presents *parallel deep pushdown automata* as a parallel version of the deep pushdown automata. The main difference is that parallel deep pushdown automaton can expand n topmost noninput pushdown symbols in only one move between two configurations. Like deep pushdown automata, parallel deep pushdown automata represents automaton counterpart to n-limited state grammars.

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

1 INTRODUCTION

Context-free grammars and pushdown automata are core models of the formal language theory. This theory has constantly paid some attention to investigation of pushdown automata but there exist fewer modifications compared to the number of grammatical modifications. For some examples about grammatical modifications see [3]. The most important for this paper are *state grammars* introduced in 1970 by T. Kasai [4]. These grammars describe an infinite hierarchy of languages lying between context-free and context-sensitive languages. New models in automata theory usually describe a proper subset of context-free languages or, conversely, are powerful as Turing machines ([5]). State grammars (or more precisely n-limited state grammars) lacked any automaton counterpart until prof. Meduna defined a *deep pushdown automaton* ([6]) in 2006. Unlike conventional pushdown automata, these automata can expand noninput pushdown symbols deeper in its pushdown, not only on the pushdown top. If we apply restriction to maximum depth where noninput symbols can be expanded to some positive number n, we got the same infinite hierarchy of languages as are defined by state grammars.

This paper presents *parallel deep pushdown automata* as a parallel version of the deep pushdown automata (${}_{deep}M$) introduced in [6]. A parallel deep pushdown automaton, ${}_{deep}M_{par}$ works exactly as ${}_{deep}M$ except that it can simultaneously expand n topmost noninput symbols on the pushdown in one move between configurations. We demonstrate that this modification does not restrict the expressive power of these automata and that the deep pushdown automata are automaton counterpart to state grammars, too. Due to space constraints there are only the main ideas of the proofs presented. In conclusion of this paper there are formulated some open problems related to these automata.

2 PRELIMINARIES

This paper assumes that the reader is familiar with the theory of automata and formal languages (see [1], [2] and [5]).

$\mathbb{N}^+$ 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 *deep pushdown automaton* (see [6]) is a septuple ${}_dM = (Q, \Sigma, \Gamma, R, s, S, F)$, where d is a maximum depth at which can be noninput 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, $\mathbb{N}^+$, Γ and Σ are pairwise disjoint. $R \subseteq (\mathbb{N}^+ \times Q \times (\Gamma - (\Sigma \cup \{\#\})) \times Q \times (\Gamma - \{\#\})^+) \cup (\mathbb{N}^+ \times Q \times \{\#\} \times Q \times (\Gamma - \{\#\})^* \{\#\})$. 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 $r = mpA \rightarrow qw$ and call r a *rule*.

For each rule $r = mpA \rightarrow qw \in R$ we define depth$(r) = m$, lhs$(r) = A$, rhs$(r) = w$, stateFrom$(r) = p$ and stateTo$(r) = q$.

For every $k \geq 1$, ${}_{deep}\mathbf{PD}_k$ denotes the family of languages defined by the deep pushdown automaton of depth i, where $1 \leq i \leq k$. Analogously ${}^{empty}_{deep}\mathbf{PD}_k$ denotes the family of languages defined by the parallel deep automaton of depth i by empty pushdown, where $1 \leq i \leq k$.

A *state grammar* (see [4]) is a quintuple $G = (V, W, T, P, S)$, 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 $(q, A) \rightarrow (p, v) \in P$ throughout.

For every $z \in V^*$, set ${}_G states(z) = \{q \mid (q, B) \rightarrow (p, v) \in P$, where $B \in (V - T) \cap alph(z)$, $v \in V^+$, $q, p \in W\}$ If $(q, A) \rightarrow (p, v) \in P$, $x, y \in V^*$, ${}_G states(x) \cap \{q\} = \varnothing$, then G makes a derivation step from (q, xAy) to (p, xvy), symbolically written as $(q, xAy) \Rightarrow (p, xvy)[(q, A) \rightarrow (p, v)]$ in G; in addition, if n is a positive integer satisfying $occur(xA, V - T) \leq n$, we say that $(q, xAy) \Rightarrow (p, xvy)$ $[(q, A) \rightarrow (p, v)]$ is n-limited, symbolically written as (q, xAy) ${}_n\Rightarrow$ (p, xvy) $[(q, A) \rightarrow (p, v)]$. Usually if there is no possibility of confusion, we simplify $(q, xAy) \Rightarrow (p, xvy)$ $[(q, A) \rightarrow (p, v)]$ to $(q, xAy) \Rightarrow (p, xvy)$ and (q, xAy) ${}_n\Rightarrow$ (p, xvy) $[(q, A) \rightarrow (p, v)]$ 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.

3 DEFINITIONS

In this section both informal and formal definition of parallel deep pushdown automata will be given.

3.1 INFORMAL DEFINITION

The parallel deep pushdown automaton differs from a standard deep pushdown automaton by possibility to expand several topmost noninput pushdown symbols in one move. This is achieved by the form of the rules. Each rule is composition of simpler rules. The first noninput pushdown symbol in the rule corresponds to the topmost noninput symbol on the pushdown, the second noninput pushdown symbol corresponds to the second topmost noninput 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 automaton. The numbering of noninput 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 noninput symbols on the pushdown and their arrangement is identical with arrangement of these symbols in the rule.

3.2 FORMAL DEFINITION

A *parallel deep pushdown automaton* is a septuple ${}_dM_p = (Q, \Sigma, \Gamma, R, s, S, F)$, where d is a maximum depth at which can be noninput symbol expanded, Q is a finite set of the states, Σ is an input alphabet (finite set of input symbols), 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, $\mathbb{N}^+$, Γ and Σ are pairwise disjoint. R is a finite set of rules in a form $p(A_1, \ldots, A_n) \to q(w_1, \ldots, w_n)$, where $p, q \in Q$, $A_i \in \Gamma - \Sigma$, $w_i \in \Gamma^+$, $1 \leq i \leq n$.

A *configuration* of the parallel deep pushdown automaton M is a triple $Q \times \Sigma^* \times (\Gamma - \{\#\})^*\{\#\}$. Let χ be a set of all configurations of automaton M and let $x, y \in \chi$ be two configurations. $x \Rightarrow y$ is a move between these 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 M *pops* its pushdown from x to y, $x_p \Rightarrow y$. M *expands* its pushdown if $x = (p, au, wAz), y = (q, au, wvz), r = p(A) \to q(v) \in R$, accordingly to rule r, symbolically $x_e \Rightarrow y[p(A) \to q(v)]$ or $x_e \Rightarrow y$ if there is only one usable rule. In the standard manner we can extend ${}_p\Rightarrow$, ${}_e\Rightarrow$, and $\Rightarrow$ to ${}_p\Rightarrow^m$, ${}_e\Rightarrow^m$, and $\Rightarrow^m$, respectively, for $m \geq 0$. Then based on ${}_p\Rightarrow^m$, ${}_e\Rightarrow^m$, and $\Rightarrow^m$, define ${}_p\Rightarrow^+$, ${}_p\Rightarrow^*$, ${}_e\Rightarrow^+$, ${}_e\Rightarrow^*$, $\Rightarrow^+$ and $\Rightarrow^*$. Let ${}_nM_p$ be of depth $n \in \mathbb{N}^+$, where n is maximal depth of rules. We define a *language accepted by* ${}_nM_p$, $L({}_nM_p)$ as $L({}_nM_p) = \{w \in \Sigma^* : (s, w, S\#) \Rightarrow^* (f, \varepsilon, \#) \in {}_nM \text{ with } f \in F\}$ and *language accepted by* ${}_nM_p$, $E({}_nM_p)$ *by empty pushdown* as $E({}_nM_p) = \{w \in \Sigma^* : (s, w, S\#) \Rightarrow^* (q, \varepsilon, \#) \in {}_nM \text{ with } q \in Q\}$.

For every $k \geq 1$, ${}_{par.d.}\mathbf{PD}_k$ denotes the family of languages defined by the parallel deep pushdown automaton of depth i, where $1 \leq i \leq k$. Analogously ${}^{empty}_{par.d.}\mathbf{PD}_k$ denotes the family of languages defined by the parallel deep pushdown automaton of depth i by empty pushdown, where $1 \leq i \leq k$. **CF** denote the family of context-free languages. **CS** denote the family of context-sensitive languages.

4 RESULTS

Lemma 1 *For every state grammar G, and for every $n \geq 1$, there exists a parallel deep pushdown automaton of depth n, ${}_nM$, such that $L(G, n) = L({}_nM)$.*
Proof idea Parallel deep pushdown automaton ${}_nM_p$ simulates n-limited derivations in G. It allways record the first n nonterminals occuring in the current sententional form. If there appear less than n nonterminals in the sentential form, it completes them to n with symbols #). ${}_nM_p$ simulates a derivation step in the pushdown and, simultaneously, records the newly generated nonterminals in the state. When grammar G succesfully completes the generation of terminal string, ${}_nM_p$ completes reading, empties its pushdown and enters the final state \$.

Lemma 2 *For every $n \geq 1$ and every parallel deep pushdown automaton, ${}_nM_p$, there exist a state grammar G, which generates language accepted by ${}_nM_p$, $L(G, n) = L({}_nM_p)$*

Proof idea Grammar G simulates application of rule $r = p(A_1, A_2, \ldots, A_i) \to q(w_1Y_1, w_2Y_2, \ldots, w_iY_i)$ in ${}_nM_p$. It makes left-to-right scan of the sentential form, counting the occurences of nonterminals until it reaches the ith ocurence of nonterminal. If this occurence equals A_i it replaces this with w_iY_i. Scan direction changes to right-to-left. It reaches previous, $(i-1)$th, occurence of nonterminal. If this occurence equals to A_{i-1} it replaces with $w_{i-1}Y_{i-1}$. These steps are repeated until it returns to beginning of sentential form. If all i nonterminals are successfully replaced, state is changed from p to q and direction is changed back to left-to-right. Otherwise the sentence form is restored to the state before applying the rule. After successful acceptance of input string in simulated ${}_nM_p$ is used rule form step 5 and grammar G successfully derives string.

Lemma 3 *For every deep pushdown automaton, ${}_dM$, exist parallel deep pushdown automaton, ${}_nM_p$, which accept identical language, $L({}_dM) = L({}_dM_p)$.*

Proof idea Parallel deep pushdown automaton, ${}_nM_p$, simulates application of ${}_nM$ rules. Every rule $r = mpA \to qw$, $r_0 \in R_{nM}$, $p,q \in Q$, $A \in N$, $w \in \Gamma^+$ can be substituted with rules in form $r = p(A_1, A_2, \ldots, A_{m-1}, A) \to q(A_1, A_2, \ldots, A_{m-1}, w) \in R$, $A_i \in N$, $1 \leq i \leq m-1$, or with a analogous composition of several consecutive rules. E.g. rules $r_1 = m_1 q_1 A \to qw_1$, $r_2 = m_2 qB \to q_2 w_2$, $r_1, r_2 \in R_{nM}$, $A, B \in N$, $w_1, w_2 \in \Gamma^+$, can be substituted with rules in form $r = q_1(A_1, A_2, \ldots, A_{m_1-1}, A, A_{m_1+1}, \ldots, A_{m_2-1}, B) \to q_2(A_1, A_2, \ldots, A_{m-1}, w_1, A_{m_1+1}, \ldots, A_{m_2-1}, w_2) \in R$, $A_i \in N$, $1 \leq i \leq m_2 - 1 \wedge i \neq m_1$.

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 $L = L({}_nM_p)$.*
Proof This theorem is based on Lemma 1 and Lemma 2.

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 $L = E({}_nM_p)$.*
Proof This theorem is based on Lemma 1 and Lemma 2.

Theorem 3 ${}_{par.d.}\boldsymbol{PD}_n = {}_{deep}\boldsymbol{PD}_n$.
Proof This theorem follows from Theorem 1 and analogous Theorem 1 in [6].

Theorem 4 ${}^{empty}_{par.d.}\boldsymbol{PD}_n = {}_{par.d.}\boldsymbol{PD}_n \subset {}_{par.d.}\boldsymbol{PD}_{n+1} = {}^{empty}_{par.d.}\boldsymbol{PD}_{n+1}$, *for every $n \geq 1$.*
Proof This theorem is based on Theorems 1 and 2 and on hierarchy of state grammars presented in Theorem 5 in [4].

Theorem 5 ${}_{par.d.}\boldsymbol{PD}_1 = {}^{empty}_{par.d.}\boldsymbol{PD}_1 = \boldsymbol{CF}$.
Proof Follows from Theorem 4 and 1-limited state grammars which characterize **CF**.

Theorem 6 *For every $n \geq 1$,* ${}_{par.d.}\boldsymbol{PD}_n = {}^{empty}_{par.d.}\boldsymbol{PD}_n \subset \boldsymbol{CS}$.
Proof From Theorem 2 and Corollary 1 in [4], for every $n \geq 1$, the n-limited state grammars generate a proper subfamily of **CS**. Thus, this Theorem follows from Lemmas 1, 2 and Theorems 1, 2.

4.1 COMPARISON

Now let's look at comparison the activities of a parallel and the original automaton.
Let's have a deep pushdown automaton ${}_2M = (\{s,p,q,f\}, \{a,b,c\}, \{A,S,\#,a,b,c\}, \{[1]1sS \to qAA, [2]1qA \to paAb, [3]1qA \to fab, [4]2pA \to qAc, [5]1fA \to fc\}, s, S, \{f\})$ and input string $aabbcc \in L = \{a^n b^n c^n, n \geq 1\}$.

Automaton ${}_2M$ makes following sequence of moves: $(s, aabbcc, S\#)\ {}_e \Rightarrow (q, aabbcc, AA\#)[1]\ {}_e \Rightarrow (p, aabbcc, aAbA\#)[2]\ {}_p \Rightarrow (p, abbcc, AbA\#)\ {}_e \Rightarrow (q, abbcc, AbAc\#)[4]\ {}_e \Rightarrow (f, abbcc, abbAc\#)[3]\ {}_p \Rightarrow^3 (f, cc, Ac\#)\ {}_e \Rightarrow (f, cc, cc\#)[5]\ {}_p \Rightarrow^2 (f, \varepsilon, \varepsilon\#)$. The input string is accepted after 11 moves (5 expansions and 6 pops).

Given automaton ${}_2M$ can be converted (Lemma 3) to the parallel deep pushdown automaton ${}_2M_p = (\{s,p,q,f\}, \{a,b,c\}, \{A,S,\#,a,b,c\}, \{[1]s(S) \to q(AA), [2]q(A) \to p(aAb), [3]q(A) \to f(ab), [4]f(A) \to f(c), [5]q(A,A) \to q(aAb,Ac), [6]p(A,A) \to q(A,Ac), [7]p(S,A) \to q(S,Ac)\}, s, S, \{f\})$.

With identical input string *aabbcc* makes ${}_2M_p$ following sequence of moves: $(s, aabbcc, S\#)\ {}_e \Rightarrow (q, aabbcc, AA\#)[1]\ {}_e \Rightarrow (q, aabbcc, aAbAc\#)[6]\ {}_p \Rightarrow (q, abbcc, AbAc\#)\ {}_e \Rightarrow (f, abbcc, abbAc\#)[3]\ {}_p \Rightarrow^3 (f, cc, Ac\#)\ {}_e \Rightarrow (f, cc, cc\#)[5]\ {}_p \Rightarrow^2 (f, \varepsilon, \varepsilon\#)$. The input string is accepted after 10 moves (4 expansions and 6 pops). The parallel deep pushdown automaton made fewer expansions than original deep pushdown automaton.

5 CONCLUSION

In this paper a new extension of deep pushdown automata was presented. The main difference between deep pushdown automata and parallel deep pushdown automata is the fact that presented version can expand more noninput pushdown symbols on the pushdown in one move. Hence follows the main advantage. The parallel deep pushdown automaton can generally make fewer moves to accept the input string.

5.1 OPEN PROBLEMS

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

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

ACKNOWLEDGEMENT

This work was supported by the BUT FIT grant FIT-S-12-3 and by the research plan "Security – Oriented Research in Information Technology", 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] 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

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

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