General Top-Down Parsers Based On Deep Pushdown Expansions

Alexander Meduna *
meduna@fit.vutbr.cz

Zbyněk Křivka*
krivka@fit.vutbr.cz

Abstract: This paper discusses a generalization of the classical general top-down parsers formalized by pushdown automata. This generalization consists in allowing them to make expansions deeper in the pushdown. Based on the expansion depth, this paper establishes an infinite hierarchy of language families between the families of context-free and context-sensitive languages. The discussion about several modifications, their properties and open problems follows.

Keywords: parsing, pushdown automata, state grammars, infinite hierarchy

1 Introduction

Consider the standard conversion of a context-free grammar to a pushdown automaton acting as a general top-down parser for the grammar (see, for instance, page 176 in [1], page 148 in [4], page 113 in [6], or page 444 in [8]). During every move, the parser either pops or expands its pushdown depending on the symbol occurring on the pushdown top. More specifically, if an input symbol occurs on the pushdown top, the parser compares the pushdown top symbol with the current input symbol, and if they coincide, M *pops* the topmost symbol from the pushdown and proceeds to the next input symbol on the input tape. If a nonterminal occurs on the pushdown top, the parser expands its pushdown so it replaces the top nonterminal with a string. M accepts an input string, x, if it makes a sequence of moves so it completely reads x, empties its pushdown, and enters a final state; the latter requirement of entering a final state is dropped in some books (see, for instance, Algorithm 5.3.1.1.1 in [6] or Theorem 5.1 in [2]).

In this paper, we discuss a slight generalization of this parser. Indeed, the generalized top-down parser works exactly as the above parser except that it can make *expansions of depth* m so it replaces the mth topmost pushdown symbol with a string, for some $m \geq 1$. We demonstrate that the top-down parsers that make expansions of depth m or less accept a proper language subfamily of the language family accepted by the top-down parsers that make expansions of depth $m + 1$ or less, for every $m \geq 1$. The resulting infinite hierarchy of language families obtained in this way occurs between the family of context-free and context-sensitive languages. For every positive number n, however, there exist some context-sensitive languages that cannot be accepted by any top-down parsers that make expansions of depth n or less.

2 Preliminaries

This paper assumes that the reader is familiar with the theory of automata, formal languages, and parsing (see [1, 4, 6–8, 10]). For a set, Q, $card(Q)$ denotes the cardinality of Q. I denotes

* Brno University of Technology, Faculty of Information Technology, Department of Information Systems, Božetěchova 2, Brno 61266, Czech Republic

the set of all positive integers. For an alphabet, V, V^* represents the free monoid generated by V under the operation of concatenation. The identity of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$; algebraically, V^+ is thus the free semigroup generated by V under the operation of concatenation. For $w \in V^*$, $|w|$ denotes the length of w and $alph(w)$ denotes the set of symbols occurring in w. For $W \subseteq V$, $occur(w, W)$ denotes the number of occurrences of symbols from W in w. For every $i \geq 0$, **prefix**(w, i) is w's prefix of length i if $|w| \geq i$, and **prefix**$(w, i) = w$ if $i \geq |w| + 1$.

A *state grammar* (see [5]) 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 $P \subseteq (W \times (V - T)) \times (W \times V^+)$ 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 ${}_G states(z) = \{q | (q, B) \rightarrow (p, v) \in P$, where $B \in (V - T) \cap alph(z), v \in V^+, q, p \in W\}$. If $(q, A) \rightarrow (p, v) \in P, x, y \in V^*, {}_G states(x) = \emptyset$, 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 $(q, xAy)\ {}_n\Rightarrow\ (p, xvy)$ $[(q, A) \rightarrow (p, v)]$. Whenever there is no danger of confusion, we simplify $(q, xAy) \Rightarrow (p, xvy)$ $[(q, A) \rightarrow (p, v)]$ and $(q, xAy)\ {}_n\Rightarrow\ (p, xvy)$ $[(q, A) \rightarrow (p, v)]$ to $(q, xAy) \Rightarrow (p, xvy)$ and $(q, xAy)\ {}_n\Rightarrow\ (p, xvy)$, respectively. In the standard manner, we extend $\Rightarrow$ *to* $\Rightarrow^m$, where $m \geq 0$; then, based on $\Rightarrow^m$, we define $\Rightarrow^+$ *and* $\Rightarrow^*$. Let $n \in I$ and $v, \varpi \in (W \times V^+)$. To express that every derivation step in $v \Rightarrow^m \varpi, v \Rightarrow^+ \varpi$, and $v \Rightarrow^* \varpi$ is n-limited, we write $v\ {}_n\Rightarrow^m \varpi, v\ {}_n\Rightarrow^+ \varpi$, and $v\ {}_n\Rightarrow^* \varpi$ instead of $v \Rightarrow^m \varpi, v \Rightarrow^+ \varpi$, and $v \Rightarrow^* \varpi$, respectively. By $strings(v\ {}_n\Rightarrow^* \varpi)$, we denote the set of all strings occurring in the derivation $v\ {}_n\Rightarrow^* \varpi$. The *language of* G, $L(G)$, is defined as $L(G) = \{w \in T^* | (q, S) \Rightarrow^* (p, w), q, p \in W\}$. Furthermore, we define for every $n \geq 1, L(G, n) = \{w \in T^* | (q, S)\ {}_n\Rightarrow^* (p, w), q, p \in W\}$, and $L(G, n)$ is called *n-limited language of* G. 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 state grammar G is of *degree* n for a positive integer n if and only if $L(G, n) = L(G)$. $\boldsymbol{ST}_n$ denotes the family of languages containing (n or less)-limited languages of arbitrary state grammar. More formally, for every $n \geq 1$, set $\boldsymbol{ST}_n = \{L(G, i) |$ G is an arbitrary state grammar, $1 \leq i \leq n\}$. If $L(G, n) \neq L(G)$ for every positive integer n, then G is state grammar of *infinite degree*. Let $\boldsymbol{ST}_\infty = \bigcup_{n=1}^{\infty} \boldsymbol{ST}_n$. Let $\boldsymbol{ST}_\omega$ be the entire family of state languages.

CF and ***CS*** denote the families of context-free and context-sensitive languages, respectively.

Kasai proved in his paper (see [5]) these crucial theorems concerning state grammars (reformulated in the terms of this paper):

Theorem Kasai.2. $\boldsymbol{ST}_\omega = \boldsymbol{CS}$.

Corollary Kasai.1. $\boldsymbol{ST}_\infty \subset \boldsymbol{ST}_\omega$.

Theorem Kasai.5. For every $n \geq 1$, $\boldsymbol{ST}_n \subset \boldsymbol{ST}_{n+1}$.

Observe that for each $n \geq 1$, $\boldsymbol{ST}_n \subseteq \boldsymbol{ST}_{n+1}$ follows from the definition of state languages.

Example 1. Consider a state grammar, $G = (\{A, C, a, b, c\}, \{1, 2, 3, 4\}, \{a, b, c\}, P, S)$ with

$$\begin{aligned} P = \{ \quad & (1, S) \rightarrow (2, AC) \\ & (2, A) \rightarrow (3, aAb), \\ & (2, A) \rightarrow (4, ab), \\ & (3, C) \rightarrow (2, Cc), \\ & (4, C) \rightarrow (4, c) \qquad \}. \end{aligned}$$

The demonstration of generating the string $aabbcc$ in G follows. G makes 2-limited derivation, so G is of degree 2.

$$\begin{aligned} (1, S) &\Rightarrow (2, AC) && [(1, S) \rightarrow (2, AC)] \\ &\Rightarrow (3, aAbC) && [(2, A) \rightarrow (3, aAb)] \\ &\Rightarrow (2, aAbCc) && [(3, C) \rightarrow (2, Cc)] \\ &\Rightarrow (4, aabbCc) && [(2, A) \rightarrow (4, ab)] \\ &\Rightarrow (4, aabbcc) && [(4, C) \rightarrow (4, c)] \end{aligned}$$

Therefore, $(1, S) \Rightarrow^* (4, aabbcc)$. Observe that $L(G) = L(G, 2) = \{a^n b^n c^n | \ n \geq 1\}$, which belongs to $\boldsymbol{CS - CF}$.

3 Definitions

A *deep top-down parser*, a *DTDP* for short, is a 7-tuple, $M = (Q, \Sigma, \Gamma, R, s, S, F)$, where Q is a finite set of *states*, Σ is an *input alphabet*, and Γ is a *pushdown alphabet*, I, Q, Γ are pairwise disjoint (see Section 2 for I), $\Sigma \subseteq \Gamma, \Gamma - \Sigma$ contains a special *bottom* symbol denoted by $\#$, $R \subseteq (I \times Q \times (\Gamma - (\Sigma \cup \{\#\})) \times Q \times (\Gamma - \{\#\})^+) \cup (I \times Q \times \{\#\} \times Q \times (\Gamma - \{\#\})^* \{\#\})$ is a finite relation, $s \in Q$ is the *start state*, $S \in \Gamma$ is the *start pushdown symbol*, $F \subseteq Q$ is a set of *final states*. Instead of $(m, q, A, p, v) \in R$, we write $mqA \rightarrow pv \in R$ and call $mqA \rightarrow pv$ a *rule*; accordingly, R is referred to as the *set of M's rules*. A *configuration of M* is a triple in $Q \times \Sigma^* \times (\Gamma - \{\#\})^* \{\#\}$. Let χ denote the set of all configurations of M. Let $x, y \in \chi$ be two configurations. M *pops* its pushdown from x to y, symbolically written as $x \ {}_p\Rightarrow \ y$, if $x = (q, az, au), y = (q, z, u)$, where $a \in \Sigma, z \in \Sigma^*, u \in \Gamma^*$. M *expands* its pushdown from x to y, symbolically written as $x \ {}_e\Rightarrow \ y$, if $x = (q, w, uAz), y = (p, w, uvz), mqA \rightarrow pv \in R$, where $A \in \Gamma - \Sigma, u, v, z \in \Gamma^*, q, p \in Q, w \in \Sigma^*$, and $occur(u, \Gamma - \Sigma) = m - 1$. To express that M makes $x \ {}_e\Rightarrow \ y$ according to $mqA \rightarrow pv$, we write $x \ {}_e\Rightarrow \ y \ [mqA \rightarrow pv]$. We say that $mqA \rightarrow pv$ is a *rule of depth m*; accordingly, $x \ {}_e\Rightarrow \ y \ [mqA \rightarrow pv]$ is an *expansion of depth m*. M makes a *move* from x to y, symbolically written as $x \Rightarrow y$, if M either $x \ {}_e\Rightarrow \ y$ or $x \ {}_p\Rightarrow \ y$. If $n \in I$ is the minimal positive integer such that each of M's rules is of depth n or less, we say that M *is of depth n*, symbolically written as ${}_nM$. M is *deterministic with respect to the depth of its expansions* if for every $q \in Q, card(\{m | mqA \rightarrow pv \in R, A \in \Gamma - \Sigma, v \in \Gamma^+, p \in Q\}) \leq 1$ because at this point from the same state, all expansions that M can make are of the same depth. In the standard manner, 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 M be of depth n, for some $n \in I$. We define the *language accepted by ${}_nM$*, $L({}_nM)$, as $L({}_nM) = \{w \in \Sigma^* | \ (s, w, S\#) \Rightarrow^* (f, \varepsilon, \#)$ in ${}_nM$ with $f \in F\}$. In addition, we define the *language that M accepts by empty pushdown*, $E({}_nM)$, as $E({}_nM) = \{w \in \Sigma^* | \ (s, w, S\#) \Rightarrow^* (q, \varepsilon, \#)$ in ${}_nM$ with $q \in Q\}$.

For every every $k \geq 1$, set $\boldsymbol{TD}_k = \{L({}_iM) \mid {}_iM$ is a *DTDP*, $1 \leq i \leq k\}$ and $\boldsymbol{ETD}_k = \{E({}_iM) \mid {}_iM$ is a *DTDP*, $1 \leq i \leq k\}$.

Example 2. Consider a *DTDP*, ${}_2M = (\{s,q,p,f\},\{a,b,c\},\{A,B,S,\#\},R,s,S,\{f\})$ with

$$\begin{aligned} R = \{\ & 1sS \to qAB, \\ & 1qA \to paAb, \\ & 1qA \to pab, \\ & 2pB \to qBc, \\ & 2pB \to fc \qquad \}. \end{aligned}$$

Notice that M is deterministic with respect to the depth of its expansion. With $aabbcc$, M makes

$$\begin{aligned} (s, aabbcc, S\#) &\ {}_e\Rightarrow (q, aabbcc, AB\#) && [1sS \to qAB] \\ &\ {}_e\Rightarrow (p, aabbcc, aAbB\#) && [1qA \to paAb] \\ &\ {}_p\Rightarrow (p, abbcc, AbB\#) \\ &\ {}_e\Rightarrow (q, abbcc, AbBc\#) && [2pB \to qBc] \\ &\ {}_e\Rightarrow (p, abbcc, abbBc\#) && [1qA \to pab] \\ &\ {}_p\Rightarrow (p, bbcc, bbBc\#) \\ &\ {}_p\Rightarrow (p, bcc, bBc\#) \\ &\ {}_p\Rightarrow (p, cc, Bc\#) \\ &\ {}_e\Rightarrow (f, cc, cc\#) && [2pB \to fc] \\ &\ {}_p\Rightarrow (f, c, c\#) \\ &\ {}_p\Rightarrow (f, \varepsilon, \#) \end{aligned}$$

We write $(s, aabbcc, S\#) \Rightarrow^* (f, \varepsilon, \#)$, and we say that the string $aabbcc$ is successfully accepted by *DTDP* M. Observe that $L(M) = E(M) = \{a^n b^n c^n | n \geq 1\} \in \boldsymbol{TD}_2$, and $L(M) \in \boldsymbol{CS - CF}$.

4 Results

This section proves that $\boldsymbol{TD}_1 = \boldsymbol{ETD}_1 = \boldsymbol{CF}$ and for every $n \geq 1, \boldsymbol{ETD}_n = \boldsymbol{TD}_n \subset \boldsymbol{ETD}_{n+1} = \boldsymbol{TD}_{n+1} \subset \boldsymbol{CS}$.

Lemma 1. *For every state grammar, G, and for every $n \geq 1$, there exists a DTDP of depth n, ${}_nM$, such that $L(G,n) = L({}_nM)$; in addition, ${}_nM$ is deterministic with respect to the depth of its expansions.*

Proof. Let $G = (V, W, T, P, S)$ be a state grammar and $n \geq 1$. Set $N = V - T$. Define the homomorphism f over $(\{\#\} \cup V)^*$ as $f(A) = A$ for every $A \in \{\#\} \cup N$, and $f(a) = \varepsilon$ for every $a \in T$. Introduce the *DTDP* of depth n,

$${}_nM = (Q, T, \{\#\} \cup V, R, s, S, \{\$\}),$$

where $Q = \{s, \$\} \cup \{\langle p, u\rangle | p \in W, u \in \textbf{prefix}(v, i), v \in N^*\{\#\}^n, 1 \leq i \leq n\}$ and R is constructed by performing the following steps:

1. for every $(p, S) \to (q, x) \in P, p, q \in W, x \in V^+$, add
   $1sS \to \langle p, S\rangle S$ to R;
2. if $(p, A) \to (q, x) \in P, \langle p, uAv\rangle \in Q, p, q \in W, A \in N, x \in V^+, u \in N^*, v \in N^*\{\#\}^*$, $|uAv| = n, p \notin {}_G states(u)$, then add
   $|uA|\langle p, uAv\rangle A \to \langle q, \textbf{prefix}(uf(x)v, n)\rangle x$ to R;

3. if $A \in N, p \in W, u \in N^*, v \in \{\#\}^*, |uv| \leq n-1, p \notin {}_G states(u)$, then add rules
   $|uA|\langle p, uv\rangle A \rightarrow \langle p, uAv\rangle A$ and
   $|u\#|\langle p, uv\rangle\# \rightarrow \langle p, uv\#\rangle\#$ to R;
4. for every $q \in W$, add
   $1\langle q, \#^n\rangle\# \rightarrow \$\#$ to R.

Basic Idea

${}_nM$ simulates G's n-limited derivations so it always records the first n nonterminals occurring in the current sentential form in its state (if there appear fewer than n nonterminals in the sentential form, it completes them to n in the state by $\#s$ from behind). ${}_nM$ simulates a derivation step in the pushdown and, simultaneously, records the newly generated nonterminals in the state. When G successfully completes the generation of a terminal string, ${}_nM$ completes reading the string, empties its pushdown and enters the final state \$.

Notice that ${}_nM$ is deterministic with respect to the depth of its expansions. For rigorous formal proof that $L(G, n) = L({}_nM)$ see [9].

In this theoretically oriented paper, we have not concentrated our attention on the efficiency of the construction above. We will discuss the pragmatically oriented aspects of the construction in the future. Specifically, we plan to ensure that no useless state will be generated and that every second component of each composite state is of the same length. Let us note that the following construction in Lemma 2 is given from a theoretical viewpoint as well.

Lemma 2. *For every $n \geq 1$ and every DTDP, ${}_nM$, there exists a state grammar, G, such that $L(G, n) = L({}_nM)$.*

Proof. Let $n \geq 1$ and ${}_nM = (Q, T, V, R, s, S, F)$ be a *DTDP*. Let Z and \$ be two new symbols that occur in no component of ${}_nM$. Set $N = V - T$. Introduce sets $C = \{\langle q, i, \triangleright\rangle | q \in Q, 1 \leq i \leq n-1\}$, $D = \{\langle q, i, \triangleleft\rangle | q \in Q, 0 \leq i \leq n-1\}$, an alphabet W such that $card(V) = card(W)$, and for all $1 \leq i \leq n$, an alphabet U_i such that $card(U_i) = card(N)$. Without any loss of generality, assume that V, Q, and all these newly introduced sets and alphabets are pairwise disjoint. Set $U = \cup_{i=1}^{n} U_i$. For each $1 \leq i \leq n-1$, set $C_i = \{\langle q, i, \triangleright\rangle | q \in Q\}$ and for each $0 \leq i \leq n-1$, set $D_i = \{\langle q, i, \triangleleft\rangle | q \in Q\}$. Introduce a bijection h from V to W. For each $1 \leq i \leq n$, introduce a bijection ${}_ig$ from N to U_i. Define the state grammar

$$G = (V \cup W \cup U \cup \{Z\}, Q \cup C \cup D \cup \{\$\}, T, P, Z),$$

where P is constructed by performing the following steps:

1. add
   $(s, Z) \rightarrow (\langle s, 1, \triangleright\rangle, h(S))$ to P;
2. for every $q \in Q, A \in N, 1 \leq i \leq n-1, x \in V^+$, add
   $(\langle q, i, \triangleright\rangle, A) \rightarrow (\langle q, i+1, \triangleright\rangle, {}_ig(A))$ and
   $(\langle q, i, \triangleleft\rangle, {}_ig(A)) \rightarrow (\langle q, i-1, \triangleleft\rangle, A)$ to P;
3. if $ipA \rightarrow qxY \in R$, for some $p, q \in Q, A \in N, x \in V^*, Y \in V, i = 1, \ldots, n$, then add
   $(\langle p, i, \triangleright\rangle, A) \rightarrow (\langle q, i-1, \triangleleft\rangle, xY)$ and
   $(\langle p, i, \triangleright\rangle, h(A)) \rightarrow (\langle q, i-1, \triangleleft\rangle, xh(Y))$ to P;

4. for every $q \in Q, A \in N$, add
$(\langle q, 0, \triangleleft\rangle, A) \rightarrow (\langle q, 1, \triangleright\rangle, A)$ and
$(\langle q, 0, \triangleleft\rangle, h(Y)) \rightarrow (\langle q, 1, \triangleright\rangle, h(Y))$ to P;
5. for every $q \in F, a \in T$, add
$(\langle q, 0, \triangleleft\rangle, h(a)) \rightarrow (\$, a)$ to P.

Basic Idea

G simulates the application of $ipA \rightarrow qy \in R$ so it makes a left-to-right scan of the sentential form, counting the occurrences of nonterminals until it reaches the ith occurrence of a nonterminal. If this occurrence equals A, it replaces this A with y and returns to the beginning of the sentential form in order to analogically simulate a move from q. Throughout the simulation of ${}_nM$'s moves by G, the rightmost symbol of every sentential form is from W. G completes the simulation of an acceptance of a string x by ${}_nM$ so it uses a rule introduced in step 5 of the construction of P to change the rightmost symbol of $x, h(a)$, to a and, thereby, to generate x.

To keep this paper as readable as possible and due to insufficient space, we omit the following rigorous part of the proof. The reader can easily fill them in (for details see [9]).

Theorem 3. *For every $n \geq 1$, $\boldsymbol{ST}_n = \boldsymbol{TD}_n = \boldsymbol{ETD}_n$.*

Proof. The equivalence $\boldsymbol{ST}_n = \boldsymbol{TD}_n$ for each $n \geq 1$ follows from Lemmas 1 and 2. By analogy with that demonstration, we can establish $\boldsymbol{ST}_n = \boldsymbol{ETD}_n$ for each $n \geq 1$. This theorem holds thanks to transitiveness of equivalence relation.

The main result of this paper follows next.

Theorem 4. *$\boldsymbol{TD}_n \subset \boldsymbol{TD}_{n+1}$, for every $n \geq 1$.*

Proof. This result follows from Theorem 3 above and from Theorem 5 in [5], which says that the m-limited state grammars generate a proper subfamily of the family that $(m+1)$-limited state grammars generate, for every $m \geq 1$.

Finally, we state two theorems concerning ***CF*** and ***CS***.

Theorem 5. *$\boldsymbol{TD}_1 = \boldsymbol{ETD}_1 = \boldsymbol{CF}$.*

Proof. Let us recall that one-limited state grammars characterize ***CF*** (see [5]). Thus, Theorem 5 follows from Lemmas 1 and 2 for $n = 1$.

Theorem 6. *For every $n \geq 1$, $\boldsymbol{TD}_n = \boldsymbol{ETD}_n \subset \boldsymbol{CS}$.*

Proof. From Theorem 2 and Corollary 1 in [5], for every $n \geq 1$, the n-limited state grammars generate a proper subfamily of ***CS***. Thus, Theorem 6 follows from Lemmas 1, 2 and Theorems 3.

5 Modifications

Every following modification is presented by its difference from *DTDP* just by a gist, so the same or very analogous definitions of notions are omitted. The short discussion about generative power is included (without proofs).

5.1 *DTDP* with erasing rules

Let us note that throughout this paper, we have considered only true pushdown expansions in the sense that the pushdown symbol is replaced with a nonempty string rather than the empty string; at this point, no pushdown expansion can result in shortening the pushdown length. Nevertheless, the discussion of moves that allow *DTDP*s to replace a pushdown symbol with ε and, thereby, shorten its pushdown represent a natural generalization of *DTDP*s discussed in this paper. What is the language family defined by *DTDP*s generalized in this way?

5.2 Symbol-dependent *DTDP*

The definition of symbol-dependent deep top-down parser is identical up to the definition of expansion between two *DTDP*'s configurations:

$$(q, w, uAz) \ {}_e\Rightarrow \ (p, w, uvz) \ [mqA \rightarrow pv], \text{ where } occur(u, \{A\}) = m - 1.$$

For instance, language $a^n b^n c^n$, $n \geq 1$ can be described by symbol-dependent *DTDP* with rules:

$$1sS \rightarrow qAB, \ 1qA \rightarrow paAb, \ 1qA \rightarrow pab, \ 1pB \rightarrow qBc, \ 1pB \rightarrow fc.$$

5.3 Input-reading *DTDP*

We do not distinguish pop and expansion type of moves in this modification. These two kinds of actions (pop, expand) are composed together into the rule of the form

$$(amqA \rightarrow px), a \in \Sigma \cup \{\varepsilon\}, m \geq 1, p, q \in Q, A \in \Gamma - \Sigma, x \in \Gamma^+.$$

So, the selection of applicable rules is based on actual state, actual input symbol, and non-input symbol in depth of m in the pushdown. The power of this input-reading deep top-down parser is the next open question.

5.4 Relatively *DTDP*

Consider changed form of rules

$$\mu qA \rightarrow pv, \mu \in \{1, 0, -1\}, p, q \in Q, A \in \Gamma - \Sigma, v \in \Gamma^+.$$

Each configuration can be written as (q, x, uAz, Π), $\Pi \in I$, and starting configuration as $(s, w, S\#, 1)$, respectively. The expansion move is defined as

$$(q, x, uAz, \Pi) \ {}_e\Rightarrow \ (p, x, uvz, \Pi + \mu) \ [\mu qA \rightarrow pv], \text{ where } occur(uA, \Gamma - \Sigma) = \Pi - \mu.$$

We can also introduce a generalization of the same power, where $\mu \in \{n, n-1, \ldots, 1, 0, -1, \ldots, -(n-1), -n\}$ for some constant $n \in I$. It is easy to prove equivalence between relatively *DTDP* and its generalization. On the other hand comparison with *DTDP* is open problem again.

5.5 *DTDP* with symbol searching

If neither pop nor expansion is possible for any rule in actual state, then take a look in the special set of rules F of *DTDP* and make symbol searching action. F is called *failure set*. Rule from F is applied in the following way:
$x = (q, w, uAz) \ _{ss}\Rightarrow \ (p, w, uvz)\ [f\colon \mu qA \rightarrow pv]$ such that no rule from R can be used in the configuration x and from F is chosen rule f such that m is minimal positive integer greater or equal to μ, where $m = occur(u, \Gamma - \Sigma)$.

Observe that this kind of *DTDP* acts in analogical way to appearance checking mode in regulated grammars (see [3]).

In addition, most of these modifications can be combined into much more complicated automata models.

6 Conclusion

Deep top-down parsers fulfill space between classical pushdown automata and linear bounded automata which characterize context-free and context-sensitive languages, respectively. The power of this new formal model is proved by equivalence to the family of n-limited languages generated by state grammars. Then, several variants of *DTDP* are introduced.

Future research can be divided into two areas: (a) looking for more powerful modifications, exploring their properties and (b) studying determinism and searching for applications in parsing of non-context-free languages.

The authors acknowledge the support of GAČR grant 201/04/0441. They also thank Tomáš Kopeček for his comments concerning this paper.

Bibliography

1. Aho, A. V., Ullman, J. D.: *The Theory of Parsing, Translation and Compiling, Volume I: Parsing* (Prentice Hall, Englewood Cliffs, New Jersey, 1972)
2. Autebert, J., Berstel, J., Boassom, L.: Context-Free Languages and Pushdown Automata. In: *Handbook of Formal Languages*, ed. by Rozenberg, G., Salomaa, A., vol 1 (Springer, 1997)
3. Dassow, J., Păun, G.: *Regulated Rewriting in Formal Language Theory* (Springer, Berlin, 1989)
4. Harrison, M. A.: *Introduction to Formal Language Theory* (Addison-Wesley, Reading, Massachusetts, 1978)
5. Kasai, T.: An Hierarchy Between Context-Free and Context-Sensitive Languages. In: *Journal of Computer and System Sciences*, Vol. 4 (1970) pp 492-508
6. Lewis, H. R., Papadimitriou, C. H.: *Elements of the Theory of Computation* (Prentice-Hall, Englewood Cliffs, 1981)
7. Martin, J. C.: *Introduction to Languages and the Theory of Computation* (McGraw-Hill, New York, 1991)
8. Meduna, A.: *Automata and Languages: Theory and Applications* (Springer, London, 2000)
9. Meduna, A.: Deep Pushdown Automata. In: *Acta Informatica* (2006, in press)
10. Sudkamp, T. A.: *Languages and Machines* (Addison Wesley, Reading, Massachusetts, 1988)