Theoretical Computer Science 449 (2012) 23–36

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

The size-cost of Boolean operations on constant height deterministic pushdown automata

Zuzana Bednárová [a], Viliam Geffert [a], Carlo Mereghetti [b,*], Beatrice Palano [b]

[a] Department of Computer Science, P. J. Šafárik University, Jesenná 5, 04154 Košice, Slovakia

[b] Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, via Comelico 39, 20135 Milano, Italy

ARTICLE INFO

Keywords:
Finite state automata
Regular languages
Pushdown automata

ABSTRACT

We study the size-cost of Boolean operations on *constant height deterministic pushdown automata*, i.e., on the traditional pushdown automata with a built-in constant limit on the height of the pushdown. We design a simulation showing that a complement can be obtained with a polynomial tradeoff. For intersection and union, we show an exponential simulation, and prove that the exponential blow-up cannot be avoided.

1. Introduction

The model of constant height pushdown automata is introduced in [1], where it is studied from a descriptional complexity viewpoint. Roughly speaking, this device is a traditional pushdown automaton (see, e.g., [2]) with a built-in constant limit (i.e., not depending on the input length) on the height of the pushdown. It is a routine exercise to show that such devices accept exactly regular languages. Nevertheless, a representation of regular languages by constant height pushdown automata can potentially be more succinct than by standard formalisms. In fact, in [1], optimal exponential and double-exponential gaps are proved between the size of constant height deterministic and nondeterministic pushdown automata (DPDAs and NPDAs, respectively), deterministic and nondeterministic finite state automata (DFAs and NFAs), and that of classical regular expressions.

In this paper, we continue the investigation on the descriptional power of constant height DPDAs by tackling a truly classical problem in descriptional complexity, namely, the *size-cost of Boolean operations*. There exists a wide literature on this research issue with respect to, e.g., finite state automata [3,4], regular expressions [5,6], grammars [7,8], etc. In our context, the problem can be expressed as: given two constant height DPDAs A and B accepting the languages $L(A)$ and $L(B)$, respectively, determine the size — as a function of the sizes of A and B — of constant height DPDAs accepting $L(A)^c$, $L(A) \cap L(B)$, and $L(A) \cup L(B)$.

After introducing some basic notation, we analyze the size-cost of complementing a constant height DPDA A working with a finite set of states Q, a finite pushdown alphabet Γ, and a pushdown of constant height h. A DPDA for $L(A)^c$ is proposed, working with $\|Q'\| \leq \|Q\|(h+1)+1$ states and using the same pushdown alphabet and the same constant height. As an additional bonus, we can replace each DPDA by an equivalent constant height DPDA with $\|Q'\|$ states in a "normal form" such that, on any input of length $\ell \geq 0$, the number of reversals in the pushdown movement does not exceed 2ℓ and the computation always halts after reading the entire input. Basically, the size increase directly reflects the cost of detecting and eliminating pushdown overflow/underflow, but there are also other "non-conventional" rejection policies such as blocking by undefined transitions or infinite loops manipulating with pushdown but not consuming any input.

* Corresponding author. Tel.: +39 02 503 16261; fax: +39 02 503 16261.
E-mail addresses: ivazuzu@eriv.sk (Z. Bednárová), viliam.geffert@upjs.sk (V. Geffert), mereghetti@dsi.unimi.it (C. Mereghetti), palano@dsi.unimi.it (B. Palano).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.05.009

Afterward, we study the costs of intersection and union. Given two constant height DPDAs A and B working with respective sets of states Q_A, Q_B, pushdown alphabets Γ_A, Γ_B, and constant heights h_A, h_B, we design a DPDA for $L(A) \cup L(B)$ with a number of states bounded by $O(\|Q_A\| \cdot h_A \cdot \|Q_B\| \cdot \|\Gamma_B\|^{h_B})$. This reflects the fact that we keep the current states for both A and B in the finite control state, together with the entire content of the pushdown store for B, but only the pushdown height for A. The number of states in a DPDA for $L(A) \cap L(B)$ can be slightly reduced, to $O(\|Q_A\| \cdot \|Q_B\| \cdot \|\Gamma_B\|^{h_B})$. Both these machines use the pushdown alphabet Γ_A and the constant height h_A. The roles of A and B can be swapped here, hence, the cost is exponential in $h = \min\{h_A, h_B\}$. Consequently, if one of these two DPDAs is actually a finite state automaton with $h = 0$ (or even a machine with a constant height counter, using $\|\Gamma\| = 1$), the cost of intersection and union becomes polynomial.

Nevertheless, for the general case, we prove that the exponential blow-up cannot be avoided: for each given alphabet Σ, we exhibit $\{L'_n\}_{n\geq 1}$ and $\{L''_n\}_{n\geq 1}$, two families of languages such that: (i) both L'_n and L''_n can be accepted by DPDAs using $2 \cdot \|\Sigma\| + 2$ states and $\|\Sigma\| - 1$ pushdown symbols, with pushdown height $h = n$, but (ii) their intersection cannot be accepted by any DPDA in which both the number of states and the pushdown height are below $(\|\Sigma\| - 1)^{n/9 - O(\log n)}$. In the case of union, the corresponding lower bound we can establish is $(\|\Sigma\| - 1)^{n/15 - O(\log n)}$. (This required, among others, to introduce some completely new proof techniques, since already a machine with a polynomial pushdown height can use exponentially many different pushdown contents, and hence the standard pigeonhole arguments could not be applied directly, to obtain exponential lower bounds.)

2. Preliminaries

The set of words on an alphabet Σ, including the empty word ε, is denoted here by Σ^*. By $|\sigma|$, we denote the length of a word $\sigma \in \Sigma^*$ and by Σ^i the set of words of length i, with $\Sigma^0 = \{\varepsilon\}$ and $\Sigma^{\leq m} = \bigcup_{i=0}^{m} \Sigma^i$. For a word $\sigma = s_1 \dots s_\ell$, let $\sigma^R = s_\ell \dots s_1$ denote its reversal. By $\|S\|$, we denote the cardinality of a set S, and by S^c its complement. We assume the reader is familiar with the models of deterministic and nondeterministic *finite state automata* (DFA and NFA, for short) and *pushdown automata* (DPDA and NPDA, see, e.g., [2]).

For technical reasons, we introduce the NPDAs in the following form [1], where instructions manipulating the pushdown store are clearly distinguished from those reading the input tape: an NPDA is a sextuplet $A = \langle Q, \Sigma, \Gamma, H, q_1, F\rangle$, where Q is the finite set of states, Σ the input alphabet, Γ the pushdown alphabet, $q_1 \in Q$ the initial state, $F \subseteq Q$ and $Q \setminus F$ the sets of accepting and rejecting states, respectively, and $H \subseteq Q \times (\{\varepsilon\} \cup \Sigma \cup \{+, -\} \cdot \Gamma) \times Q$ the *transition relation*, with the following meaning.

(i) $(p, a, q) \in H$: if the next input symbol is a, A gets from the state p to the state q by reading the symbol a, not using the pushdown store. The set of states starting from which some transitions read from the input is denoted by Q_Σ. (That is, $Q_\Sigma = \{p \in Q : (p, a, q) \in H, \text{ for some } a \in \Sigma \text{ and } q \in Q\}$.)

(ii) $(p, -X, q) \in H$: if the symbol on top of the pushdown is X, A gets from p to q by popping X, not using the input tape. The set of states starting from which some transitions pop from the pushdown is denoted by $Q_{-\Gamma}$.

(iii) $(p, \varepsilon, q) \in H$: A gets from p to q without using the input tape or the pushdown store. The set of states from which some ε-transitions start will be denoted by Q_ε.

(iv) $(p, +X, q) \in H$: A gets from p to q by pushing the symbol X onto the pushdown, not using the input tape. The set of states starting from which some transitions push symbols onto the pushdown is denoted by $Q_{+\Gamma}$.

(v) For completeness, $Q_\emptyset = Q \setminus (Q_\Sigma \cup Q_{-\Gamma} \cup Q_\varepsilon \cup Q_{+\Gamma})$ denotes the set of all remaining states, from which no transitions can start. (In such states, A is forced to halt.)

Such a machine does not need any initial pushdown symbol. An *accepting computation* begins in the state q_1 with the empty pushdown store, and ends in an accepting state $p \in F$ after reading the entire input. As usual, $L(A)$ denotes the language accepted by A.

In most of the standard textbooks, a transition function is defined by a function $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$, combining input and pushdown operations into a single step [2]. However, it is not hard to see that any NPDA in the classical form can be transformed into the new form and vice versa, preserving determinism in the case of DPDAs. A classical transition $\delta(p, a, X) \ni (q, X_1 \dots X_k)$ can be simulated by $(p, -X, p_X)$, (p_X, a, q_0), $(q_0, +X_1, q_1)$, $\dots$, $(q_{k-2}, +X_{k-1}, q_{k-1})$, $(q_{k-1}, +X_k, q)$, where p_X and $q_0, \dots, q_{k-1}$ are some new states, with obvious modifications for $k < 2$. This also requires to store some initial pushdown symbol X_1 at the very beginning, using $(q'_1, +X_1, q_1)$, where q'_1 is a new initial state. The converse transformation is also quite straightforward. (For a more detailed reasoning for introducing our new form, see also Footnote 4 in Section 4 below.)

Given a constant $h \geq 0$, we say that the NPDA A is of pushdown height h if, for any word in $L(A)$, there exists an accepting computation along which the pushdown store never contains more than h symbols. From now on, we shall consider *constant height* pushdown automata only. Such a machine will be denoted by a septuplet $A = \langle Q, \Sigma, \Gamma, H, q_1, F, h\rangle$, where $h \geq 0$ is a constant denoting the pushdown height, and all other elements are defined as above. By definition, the meaning of the transitions in the form (iv) is modified as follows:

(iv′) $(p, +X, q) \in H$: *if the current pushdown store height is smaller than* h, then A gets from the state p to the state q by pushing the symbol X onto the pushdown, not using the input tape; *otherwise* A aborts and rejects.

For a constant height NPDA, a fair definition of size must account for the size of all the components the device consists of, i.e., the finite state control plus the pushdown store and its alphabet.[1] So, according to [1]:

Definition 2.1. The *size of a constant height* NPDA $A = \langle Q, \Sigma, \Gamma, H, q_1, F, h\rangle$ is the ordered triple $\text{size}(A) = (\|Q\|, \|\Gamma\|, h)$.

A *deterministic pushdown automaton* (DPDA) is, roughly speaking, an NPDA for which there is at most one possible move for any configuration. By definition, it must satisfy the following conditions:

- The sets $Q_\Sigma, Q_{-\Gamma}, Q_\varepsilon, Q_{+\Gamma}, Q_\emptyset$ are pairwise disjoint. (The set $Q_\emptyset$ is listed here just for completeness, satisfying this condition automatically.)
- For each $p \in Q$ and each $\alpha \in \Sigma \cup \{-\}\cdot\Gamma \cup \{\varepsilon\}$, there can exist at most one state $q \in Q$ with a transition $(p, \alpha, q) \in H$.
- For each $p \in Q_{+\Gamma}$, there can exist at most one pushdown symbol $X \in \Gamma$ and at most one state $q \in Q$ with a transition $(p, +X, q) \in H$.
- $F \subseteq Q_\Sigma$.

The last condition says that an accepting computation, after reading the last symbol from the input and, optionally, after passing through a finite number of non-accepting states, must halt in an accepting state $p \in Q_\Sigma$, i.e., in a state in which the machine waits for a next input symbol. This resolves a potential ambiguity in interpreting the outcome of the computation, if the machine does not stop immediately after reading the last input symbol.[2]

The reader may easily verify that, for $h = 0$, the definition of constant height NPDA exactly coincides with that of an NFA. (In the deterministic case, $h = 0$ yields a DFA in which ε-transitions are allowed, under the condition that $(p, \varepsilon, q) \in H$ forbids any other transitions starting from p and that $p \notin F$.)

Second, the class of languages accepted by constant height NPDAs coincides with regular languages. Here we provide the corresponding construction for the deterministic case, since it will be required later.

Lemma 2.2. *For each constant height* DPDA $A = \langle Q, \Sigma, \Gamma, H, q_1, F, h\rangle$, *there exists an equivalent* DFA $A' = \langle Q', \Sigma, H', q_1', F'\rangle$ *with* $\|Q'\| \leq \|Q\| \cdot \|\Gamma^{\leq h}\| + 1$.

Proof. The basic idea is to keep the current pushdown content in the finite control state (see [1, Prop. 3]). In addition, if the next-step transition is undefined, or if the height of the pushdown overflows or underflows, we switch to q_R, a new error state and ignore the rest of the input. This gives an intermediate machine $\hat{A} = \langle Q', \Sigma, \hat{H}, q_1', F'\rangle$, where $Q' = Q \times \Gamma^{\leq h} \cup \{q_R\}$, $q_1' = [q_1, \varepsilon]$, and $F' = F \times \Gamma^{\leq h}$.

For each state $p \in Q$ and each pushdown content $\gamma \in \Gamma^{\leq h}$, we define the transitions in $\hat{H}$ as follows.

- If $p \in Q_\Sigma$, we add $([p, \gamma], a, [p_a', \gamma]) \in \hat{H}$, for each transition $(p, a, p_a') \in H$. Moreover, for each symbol $a \in \Sigma$ without a corresponding transition (p, a, p_a') in H, we add $([p, \gamma], a, q_R) \in \hat{H}$.
- If $p \in Q_{-\Gamma}$ and $|\gamma| < h$, we add $([p, \gamma X], \varepsilon, [p_X', \gamma]) \in \hat{H}$, for each transition $(p, -X, p_X') \in H$. Moreover, for each symbol $X \in \Gamma$ without a corresponding transition $(p, -X, p_X')$ in H, we add $([p, \gamma X], \varepsilon, q_R) \in \hat{H}$.
- If $p \in Q_{-\Gamma}$, we add the transition $([p, \varepsilon], \varepsilon, q_R) \in \hat{H}$.
- If $p \in Q_\varepsilon$ and $(p, \varepsilon, p') \in H$, we add $([p, \gamma], \varepsilon, [p', \gamma]) \in \hat{H}$.
- If $p \in Q_{+\Gamma}$, $|\gamma| < h$, and $(p, +X, p') \in H$, we add $([p, \gamma], \varepsilon, [p', \gamma X]) \in \hat{H}$.
- If $p \in Q_{+\Gamma}$ and $|\gamma| = h$, we add $([p, \gamma], \varepsilon, q_R) \in \hat{H}$.
- If $p \in Q_\emptyset$, we add $([p, \gamma], \varepsilon, q_R) \in \hat{H}$.
- Finally, $(q_R, a, q_R) \in \hat{H}$, for each $a \in \Sigma$.

Note that $\hat{A}$ is a deterministic finite state automaton with ε-transitions. That is, an ε-transition $([p, \gamma], \varepsilon, [p', \gamma']) \in \hat{H}$ forbids any other transitions starting from $[p, \gamma]$, and implies that $[p, \gamma] \notin F'$. Therefore, no cycle consisting of ε-transitions contains an accepting state, and hence all transitions going to any state belonging to such cycle can be redirected into q_R. This allows us to remove all ε-cycles in $\hat{A}$ without increasing the number of states. Now, the remaining loop-free paths of ε-transitions can be eliminated by the use of standard tools (see, e.g., [2]), in such a way that both the determinism and the states are preserved. This gives us $A' = \langle Q', \Sigma, H', q_1', F'\rangle$, the desired deterministic finite state automaton without ε-transitions. □

If we do not protest against rejection by aborting in the middle of the input, the state q_R could be removed. This just leaves some transitions undefined, in the same way as in the corresponding conversion for nondeterministic machines [1].

[1] The reader may easily derive several dependences among these three quantities, e.g., by the well-known worktape compression, using the new pushdown alphabet $\Gamma' = \Gamma \times \Gamma$, we could reduce h to $h' = \lfloor h/2 \rfloor$. Conversely, by the use of a binary encoding, we can get $\Gamma' = \{0, 1\}$, to be paid by $h' = h \cdot (1 + \lfloor \log \|\Gamma\| \rfloor)$.

[2] This excludes several peculiar cases of acceptance, among others, a post-processing computation passing through an accepting state in the course of an infinite loop manipulating with pushdown but not consuming any input, or a pushdown overflow aborting the computation in an accepting state. Nevertheless, infinite loops after the last symbol is read, as well as pushdown overflows/underflows, are not excluded along *rejecting* computation paths.

3. Complementing constant height DPDAs

In this section, we study the size-increase when complementing constant height DPDAs. More precisely, given a constant height DPDA $A = \langle Q, \Sigma, \Gamma, H, q_1, F, h\rangle$, determine the size of a constant height DPDA accepting the complement of $L(A)$. Clearly, as a first direct approach, one may turn A into an equivalent DFA A' by Lemma 2.2, storing possible pushdown contents of A in the finite control states. Then, by switching accepting with rejecting states in A', the desired machine A'' would be obtained, using $\|Q''\| = \|Q\| \cdot \|\Gamma^{\leq h}\| + 1$ states, with $\Gamma'' = \emptyset$ and $h'' = 0$. Thus, if an exponential blow-up is tolerable, the use of a fixed pushdown store can be eliminated. However, to keep the cost of complementation polynomial, we need to use a full-featured constant height pushdown store.

We start with a few considerations concerning acceptance and rejection on DPDAs. By Section 2, our constant height DPDA A accepts an input string $\sigma \in \Sigma^*$ by completely sweeping it, and entering an accepting state $p \in Q_\Sigma$. On the other hand, we may encounter several different situations along a rejecting computation path. (See also Fig. 4.)

(i) NORMAL REJECTION: A completely consumes σ and enters a rejecting state $p \in Q_\Sigma$;
(ii) BLOCKING SITUATION: A gets into a state $p \in Q_\Sigma$ with the input head scanning a symbol $a \in \Sigma$, but there is no transition $(p, a, q) \in H$, for any state $q \in Q$, or A gets to $p \in Q_{-\Gamma}$ with the topmost pushdown symbol $X \in \Gamma$, but there is no transition $(p, -X, q) \in H$, or A gets into some $p \in Q_\emptyset$, from which the computation cannot carry on whatsoever;
(iii) PUSHDOWN OVERFLOW OR UNDERFLOW: A pushes a symbol onto the pushdown but h symbols are already piled up (overflow), or A tries to pop an empty pushdown (underflow);
(iv) INFINITE LOOP: A enters an infinite sequence of transitions performing push operations, pop operations, and ε-transitions; in this computation, the input symbols are not consumed and the pushdown boundaries are never violated.

If rejection took place according to possibility (i) only, then we could try to complement A by simply switching accepting with rejecting states in Q_Σ, as for DFAs. However, the presence of rejection possibilities (ii)–(iv) immediately implies that such a simple approach would not work. In what follows, we are going to eliminate possibilities (ii), (iii), and (iv). That is, we shall replace the given constant height DPDA $A = \langle Q, \Sigma, \Gamma, H, q_1, F, h\rangle$ by an equivalent DPDA $A' = \langle Q', \Sigma, \Gamma', H', q_1', F', h'\rangle$, so that A' rejects by (i) only. Compared with the original machine A, there are the following major differences.

First, besides the current state $p \in Q$, the machine A' keeps also track of the current pushdown height in its finite state control, i.e., a value $k \in \{0, \ldots, h\}$. The current content stored in the pushdown of A' mirrors the pushdown store of the original machine. That is, for any given input $\sigma = s_1 \ldots s_\ell$, the machine A' can reach a *configuration* $([p, k], s_i \ldots s_\ell, \gamma)$, where $p \in Q$, $k \in \{0, \ldots, h\}$, $s_i \ldots s_\ell$ denotes the rest of the input not processed yet, and $\gamma \in \Gamma^*$ is the pushdown content, *only if* $k = |\gamma|$ and A can reach the corresponding configuration $(p, s_i \ldots s_\ell, \gamma)$.

Second, if any of the exceptions (ii)–(iv) is detected during the simulation, A' switches to a new rejecting state q_R, in which it scans the remaining part of the input, ignoring the rest of simulation, by the use of the following transitions:

- $(q_R, a, q_R) \in H'$, for each $a \in \Sigma$, with
- $q_R \notin F'$.

This gives

- $Q' = Q \times \{0, \ldots, h\} \cup \{q_R\}$, $q_1' = [q_1, 0]$, $\Gamma' = \Gamma$, and $h' = h$.

However, partition of Q' into Q'_ε, Q'_Σ, $Q'_{+\Gamma}$, $Q'_{-\Gamma}$, $Q'_\emptyset$ will not necessarily agree with that in the original machine. Also F' is defined in a different way.

Third, observe that if we need to resume the simulation starting from any given configuration $([p, k], s_i \ldots s_\ell, \gamma)$, some leading sequence of transitions depends only on p and k, regardless of the input or the pushdown content. More precisely, such sequence can consist of unrestricted number of push operations, pop operations, or ε-transitions (provided that the pushdown boundaries are not violated), but it must end at the moment when A is either going to read the next symbol from the input, or going, for the first time, to pop the pushdown symbol stored at the level k. Until that moment, neither the input symbols nor the deepest k symbols in the pushdown can come into play. (Here we do not exclude the possibility that the length of such leading sequence is zero.) Note also that, for each given $p \in Q$ and each given $k \in \{0, \ldots, h\}$, all details about such leading sequence are completely determined by inspecting the transition table for H, which we can do in advance.

This allows us to define the transitions for A' as follows. For each given $p \in Q$ and each given $k \in \{0, \ldots, h\}$, we precompute the leading sequence that depends only on p and k. Depending on the type of state in which the leading sequence ends and, optionally, on the pushdown height at this moment, we have the following mutually exclusive cases to consider.

CASE 1: *The leading sequence ends in a state $r \in Q_\Sigma$, going to read from the input, when the pushdown height is larger than k. (Consequently, this can happen only if $k < h$.)* The situation is depicted in Fig. 1. Let $(q, +X, q') \in H$ be the last transition increasing the pushdown height from k to $k+1$, along the path connecting p with r. That is, in between q' and r, the pushdown height never gets below $k+1$. Then the outcome of the path connecting p with q' can be obtained by a single transition, namely,

- $([p, k], +X, [q', k+1]) \in H'$.

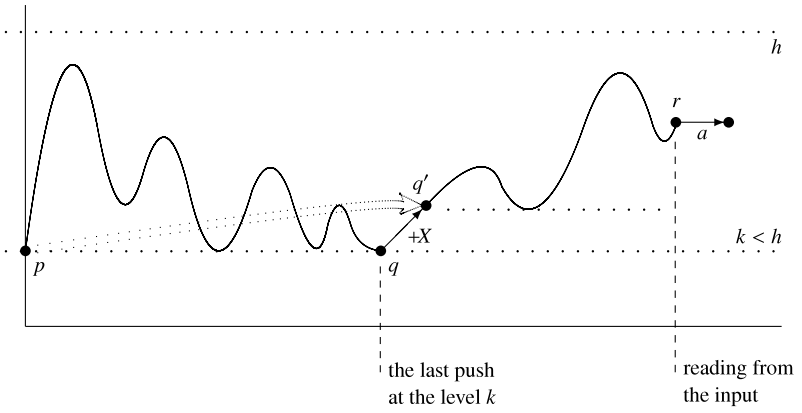

Fig. 1. After storing some more symbols in the pushdown, A reads the next input symbol.

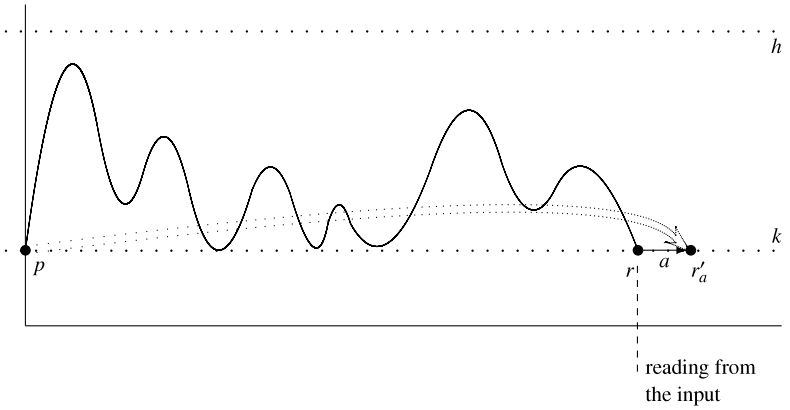

Fig. 2. A reads the next input symbol without changes in the pushdown memory.

We do not exclude $p = q$ or/and $q' = r$, with the corresponding computational segments of length zero. (The same rule applies for all cases that will follow.) Note also that $[p, k]$ qualifies for membership in $Q'_{+\Gamma}$, regardless of the original qualification of states in A.

Case II: *The leading sequence ends in a state $r \in Q_\Sigma$, going to read from the input, when the pushdown height is equal to k.* This is depicted in Fig. 2. But then the path connecting p with r does not modify the pushdown content at all, and hence the next input symbol can be read directly by a transition starting from $[p, k]$. This is obtained as follows:

- For each $(r, a, r'_a) \in H$, we add the transition $([p, k], a, [r'_a, k]) \in H'$.
- If, for some symbol $a \in \Sigma$, there does not exist any state $r'_a \in Q$ with a corresponding transition (r, a, r'_a) in H, we add $([p, k], a, q_R) \in H'$.
- Finally, if $r \in F$, we add $[p, k] \in F'$.

Blocking by undefined input-reading is resolved by entering the rejecting cycle executed in q_R; A' waits for the next input symbol in an accepting state if and only if A does.[3]

Case III: *The leading sequence ends in a state $q \in Q_{-\Gamma}$, going to pop from the pushdown, when the pushdown height is equal to k, for $k > 0$.* This situation, depicted in Fig. 3, is resolved in a similar way as in the previous case. That is, the path connecting p with q does not modify the pushdown content, and hence the topmost symbol can be popped directly, by a transition starting from $[p, k]$, as follows:

- For each $(q, -X, q'_X) \in H$, we add the transition $([p, k], -X, [q'_X, k-1]) \in H'$.
- If, for some symbol $X \in \Gamma$, there is no state $q'_X \in Q$ with a corresponding transition $(q, -X, q'_X)$ in H, we add $([p, k], -X, q_R) \in H'$.

Blocking by undefined pushdown-pop is resolved by a pop operation that enters the rejecting cycle executed in q_R. However, we do not consider whether $[p, k]$ should be included in F', since $[p, k]$ is not in Q'_Σ, but in $Q'_{-\Gamma}$.

[3] In a more sophisticated implementation, we could actually make some additional reduction, by merging $[p, k]$ with $[r, k]$ into a single state. The same trick could be used in the subsequent Case III.

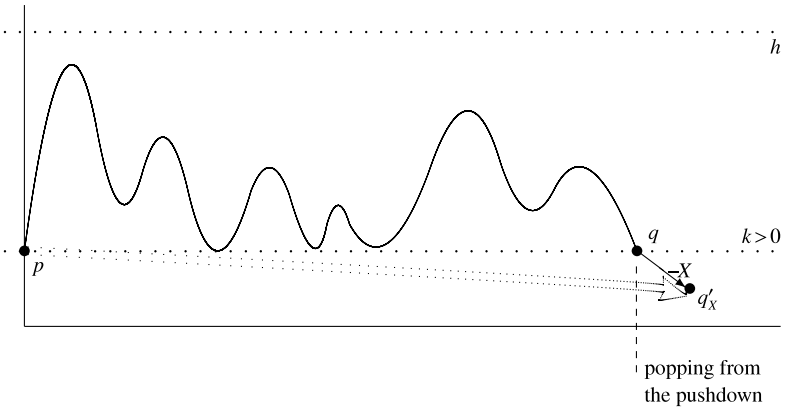

Fig. 3. *A* pops a symbol from the top of the pushdown.

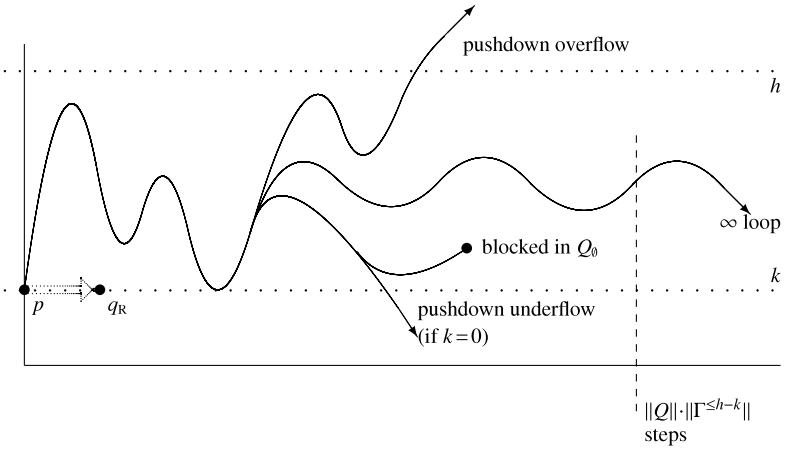

Fig. 4. Exceptional situations, after which *A* does not read the rest of the input.

Case iv: *The leading sequence ends in a state $q \in Q_{-\Gamma}$, going to pop from the pushdown, when the pushdown height is equal to k, for $k = 0$.* Since $k = 0$, the computation is now blocked by pushdown underflow, and hence the input will not be accepted. Therefore, we switch directly to the rejecting cycle executed in q_R, by the following transitions:

- $([p, k], a, q_R) \in H'$, for each $a \in \Sigma$, with
- $[p, k] \notin F'$.

This situation, together with other exceptional cases, is depicted in Fig. 4. Note also that we may obtain $[p, 0] \in Q'_\Sigma$ together with $[p, 1] \in Q'_{-\Gamma}$.

Case v: *The leading sequence ends in a state in $Q_{+\Gamma}$, going to push a symbol onto the pushdown, but the pushdown height has already reached the value h.* Also this situation is depicted in Fig. 4. Now the computation is blocked by pushdown overflow, and hence we switch to the rejecting cycle in q_R, by the use of the same transitions as presented in Case iv for pushdown underflow.

Case vi: *The leading sequence ends in a state in $Q_\emptyset$.* This time the computation is blocked in a state with no possible transitions. This is resolved in the same way as in Case iv. (See Fig. 4.)

Case vii: *The leading sequence has executed more than $\|Q\| \cdot \|\Gamma^{\leq h-k}\|$ steps, but none of the cases above has happened so far.* That is, the computation has been trapped in an infinite loop not consuming any input, and hence the input will never be accepted. (See also Fig. 4.) This follows from the fact that, by fixing the deepest k symbols in the pushdown and the input part not yet processed, the number of different configurations becomes bounded by $\|Q\| \cdot \|\Gamma^{\leq h-k}\|$, and hence some configuration must have been repeated. Therefore, we switch directly to the rejecting cycle executed in q_R, by the same transitions as presented in Case iv.

Let us now present some properties of A'. It is easy to see that A' never gets to any blocking situation, presented in item (ii) at the beginning of this section, because we have introduced the complete set of transitions for each state in Q'.

It is also easy to see that A' never violates the pushdown boundaries: by inspecting Cases i–vii, the reader may easily verify that the push operations are performed only by transitions introduced in Case i, and such transitions are executed

only if k, the current pushdown height, satisfies $k < h$. Similarly, the pop operations, introduced only by Case III, are executed only for $k > 0$. All remaining transitions, including those for the rejecting loop in q_R, are input-consuming, not changing the pushdown height, which completes the argument. Note also that A' does not use any ε-transitions.

To see that A' never gets into an infinite loop, let us examine the conditions that allow the execution of a push operation, in a given state $[p, k]$. By Case I, this is possible only if the leading sequence, starting in p with the pushdown height equal to k, ends in a state $r \in Q_\Sigma$, when the pushdown height is larger than k. (Cf. Fig. 1.) Then a transition $([p, k], +X, [q', k+1])$ is executed, which corresponds, in A, to a path from p to some q not modifying the pushdown content, followed by $(q, +X, q')$, the last transition increasing the pushdown height from k to $k+1$ along the path from p to r. This ensures that the pushdown height never gets below $k+1$ along the path in between q' and r. Consider now the next step in A', for the subsequent state $[q', k+1]$. This step depends on the outcome of the leading sequence starting in q' with the pushdown height equal to $k+1$, as presented by Cases I–VII. But, by the above observation, this leading sequence must also end in a state belonging to Q_Σ, namely, in the same state r with the same pushdown height. (However, this pushdown height is not necessarily larger than $k+1$.) This excludes Cases III–VII and leaves us the following two possibilities: either the next move in A' is again a push operation, by Case I, or it is a read operation, by Case II. Summing up, in A', a push operation must be followed by another push operation, or by an input-consuming operation. Consequently, it cannot be followed by a pop operation.

Taking also into account that A' does not violate the pushdown boundaries, we see that A' cannot execute more that h push transitions in a row, the same limit applies for the number of consecutive pop transitions. This gives us the following structure in between two input-consuming transitions:

- a sequence of consecutive pop transitions, the number of which is between 0 and h, followed by
- a sequence of consecutive push transitions, the number of which is also between 0 and h.

This structure covers also pre-processing and post-processing, however, a computation cannot start with pop transitions.

It should also be obvious that A' is equivalent to the original DPDA A. This follows from the fact that, along each computation path, every single transition (introduced in Cases I–VII) implements an outcome of changes in configuration that were made by a legal segment of transitions in the original computation path, except for transitions switching to the rejecting loop in q_R. However, such switch is activated only if the original machine rejects without consuming the rest of the input.

In conclusion, the above construction enables us to state:

Theorem 3.1. *Each constant height* DPDA $A = \langle Q, \Sigma, \Gamma, H, q_I, F, h\rangle$ *can be replaced by an equivalent constant height* DPDA $A' = \langle Q', \Sigma, \Gamma, H', q'_I, F', h\rangle$ *with the number of states bounded by* $\|Q'\| \leq \|Q\|(h+1)+1$, *using the same pushdown alphabet and pushdown height, such that (i) its computations are never blocked, violate pushdown boundaries, or get into loops, (ii) on any input string, the computation halts after reading the entire input, in a state belonging to* Q'_Σ, *in which it waits for a possible next input symbol.*

In addition, in the course of the computation, the sequence of labels in transitions follows the trajectory given by $(+\Gamma)^{\leq h}[\,\Sigma(-\Gamma)^{\leq h}(+\Gamma)^{\leq h}\,]^\ell$, *for any input of length* $\ell \geq 0$, *and hence* A' *does not make more than* 2ℓ *reversals in the pushdown movement.*

It is clear that the above theorem directly determines the cost of complementing constant height DPDAs:

Theorem 3.2. *Each constant height* DPDA $A = \langle Q, \Sigma, \Gamma, H, q_I, F, h\rangle$ *can be replaced by a constant height* DPDA A' *accepting the complement of* $L(A)$ *with the number of states bounded by* $\|Q'\| \leq \|Q\|(h+1)+1$, *using the same pushdown alphabet and pushdown height.*

Proof. For the given A, we first construct an equivalent constant height DPDA $\hat{A}$ in normal form presented by Theorem 3.1. After that, the constant height DPDA A' is obtained from $\hat{A}$ by swapping the roles of accepting and rejecting states. More precisely, $\hat{F}$ is replaced by $\hat{Q}_\Sigma \setminus \hat{F}$. □

4. Intersection and union of constant height DPDAs

Let us now deal with the other two Boolean operations, and tackle the following problem: given constant height DPDAs $A = \langle Q_A, \Sigma, \Gamma_A, H_A, q_A, F_A, h_A\rangle$ and $B = \langle Q_B, \Sigma, \Gamma_B, H_B, q_B, F_B, h_B\rangle$, determine the size of constant height DPDAs accepting $L(A) \cap L(B)$ and $L(A) \cup L(B)$.

Clearly, also here one may transform A and B into equivalent DFAs, by Lemma 2.2, and operate as usual in order to obtain DFAs with at most $(\|Q_A\| \cdot \|\Gamma_A^{\leq h_A}\|+1) \cdot (\|Q_B\| \cdot \|\Gamma_B^{\leq h_B}\|+1)$ states performing the desired tasks. However, by actually exploiting the power of pushdown storage, we can provide a slightly improved construction. Before doing so, we need to show costs for simpler operations, namely, intersection and union of DPDAs with DFAs.

Lemma 4.1. *For each constant height* DPDA $A = \langle Q_A, \Sigma, \Gamma_A, H_A, q_A, F_A, h_A\rangle$ *and each* DFA $B = \langle Q_B, \Sigma, H_B, q_B, F_B\rangle$, *there exist two constant height* DPDAs C *and* D *such that* $L(C) = L(A) \cap L(B)$ *and* $L(D) = L(A) \cup L(B)$, *both using the pushdown alphabet* Γ_A *and the pushdown height* h_A, *with the number of states in* C *bounded by* $\|Q_C\| \leq \|Q_A\| \cdot \|Q_B\|$, *while in* D *bounded by* $\|Q_D\| \leq (\|Q_A\|(h_A+1)+1) \cdot \|Q_B\|$.

Proof. First, we can easily present the constant height DPDA C for $L(A) \cap L(B)$. The basic idea is quite simple, to keep the current states for both A and B in the finite control state, and to manipulate with the pushdown in the same way as A does. If the computation in A is blocked, violates the pushdown boundaries, or gets into an infinite loop (see (ii)–(iv) listed at the beginning of Section 3), the computation in C fails for the same reasons and the input is rejected. Thus, let $C = \langle Q_C, \Sigma, \Gamma_C, H_C, q_C, F_C, h_C\rangle$, where $Q_C = Q_A \times Q_B$, $q_C = [q_A, q_B]$, $F_C = F_A \times F_B$, $\Gamma_C = \Gamma_A$, and $h_C = h_A$. Now, for each state $[p, q] \in Q_A \times Q_B$, we define the transitions in H_C as follows:

- If, for some symbol $a \in \Sigma$, $(p, a, p'_a) \in H_A$ and $(q, a, q'_a) \in H_B$, then $([p, q], a, [p'_a, q'_a]) \in H_C$.
- If, for some $X \in \Gamma$, $(p, -X, p'_X) \in H_A$, then $([p, q], -X, [p'_X, q]) \in H_C$.
- If $(p, \varepsilon, p') \in H_A$, then $([p, q], \varepsilon, [p', q]) \in H_C$.
- If $(p, +X, p') \in H_A$, then $([p, q], +X, [p', q]) \in H_C$.

(Without loss of generality, we assume that B is ε-free, keeping the same number of states.) It is clear that, on any given input, the DPDA C runs A and B in parallel; the choice of the accepting states ensures that C accepts if and only if both A and B accept.

Let us now consider a constant height DPDA D for $L(A) \cup L(B)$. Note that even if the computation in A fails, e.g., because of a pushdown overflow or an infinite loop, we still have to verify whether the given input is not accepted by B. Therefore, we first replace A by an equivalent DPDA $\hat{A} = \langle \hat{Q}_A, \Sigma, \Gamma_A, \hat{H}_A, \hat{q}_A, \hat{F}_A, h_A\rangle$ in normal form presented in Theorem 3.1, using the same pushdown, but with $\|\hat{Q}_A\| = \|Q_A\|(h_A+1)+1$ states. This machine, independent of whether the input is accepted, always halts after reading the entire input, in a state belonging to $\hat{Q}_{\Sigma,A}$, in which it waits for a possible next input symbol.

After that, the constant height DPDA D accepting $L(\hat{A}) \cup L(B)$ can be obtained in the same way as the machine for the intersection, but now we take $F_D = (\hat{F}_A \times Q_B) \cup (\hat{Q}_{\Sigma,A} \times F_B)$. That is, D accepts if and only if at least one of the original machines accept. Note also that D fulfills the criteria of the normal form presented in Theorem 3.1.

Clearly, the number of states in these two machines is bounded by $\|Q_C\| \leq \|Q_A\| \cdot \|Q_B\|$ and by $\|Q_D\| \leq (\|Q_A\|(h_A+1)+1) \cdot \|Q_B\|$, respectively. □

Now we are ready to present the costs for intersection and union of two DPDAs, beginning with upper bounds.

Theorem 4.2. *For each two constant height* DPDAs $A = \langle Q_A, \Sigma, \Gamma_A, H_A, q_A, F_A, h_A\rangle$ *and* $B = \langle Q_B, \Sigma, \Gamma_B, H_B, q_B, F_B, h_B\rangle$, *there exist two constant height* DPDAs C *and* D *such that* $L(C) = L(A) \cap L(B)$ *and* $L(D) = L(A) \cup L(B)$, *both using the pushdown alphabet* Γ_A *and the pushdown height* h_A, *with the number of states in* C *bounded by* $\|Q_C\| \leq \|Q_A\| \cdot \|Q_B\| \cdot \|\Gamma_B^{\leq h_B}\|$, *while in* D *bounded by* $\|Q_D\| \leq (\|Q_A\|(h_A+1)+1) \cdot (\|Q_B\| \cdot \|\Gamma_B^{\leq h_B}\|+1)-1$.

Proof. The key idea is first to turn one of the two initial DPDAs, say B, into an equivalent DFA $\hat{B}$ using $\|\hat{Q}_B\| = \|Q_B\| \cdot \|\Gamma_B^{\leq h_B}\|+1$ states, by Lemma 2.2. Then we combine A with $\hat{B}$ into a single machine by the use of Lemma 4.1, simulating A and $\hat{B}$ in parallel, with suitably fixed accepting states, and using the pushdown storage in the same way as A does.

However, there are some subtle differences between the DPDAs C and D. In the case of C accepting $L(A) \cap L(\hat{B})$, we do not mind if C terminates in the middle of a rejected input because the simulation of $\hat{B}$ is aborted. Therefore, before using the construction of Lemma 4.1, we can remove, from $\hat{B}$, the rejecting cycle executed in the state $\hat{q}_{R,B}$ (introduced to $\hat{B}$ by the construction of Lemma 2.2). After that, we apply Lemma 4.1 and get the DPDA C using $\|Q_A\| \cdot (\|\hat{Q}_B\|-1) = \|Q_A\| \cdot \|Q_B\| \cdot \|\Gamma_B^{\leq h_B}\|$ states.

On the other hand, in the case of D accepting $L(A) \cup L(\hat{B})$, we have to verify whether the input is not accepted by A even if $\hat{B}$ has already entered the cycle executed in $\hat{q}_{R,B}$. Therefore, we apply Lemma 4.1 directly. This gives a machine with $(\|Q_A\|(h_A+1)+1) \cdot \|\hat{Q}_B\| = (\|Q_A\|(h_A+1)+1) \cdot (\|Q_B\| \cdot \|\Gamma_B^{\leq h_B}\|+1)$ states. A finer analysis reveals that actually $Q_D = \hat{Q}_A \times \hat{Q}_B$, where $\hat{A}$ is obtained by transformation of A into the normal form presented in Theorem 3.1. Also in this machine we have a rejecting cycle, executed in a state $\hat{q}_{R,A}$. Thus, among others, D uses states $[p, \hat{q}_{R,B}]$, for each $p \in \hat{Q}_A$ and $[\hat{q}_{R,A}, q]$, for each $q \in \hat{Q}_B$. If we do not need a DPDA D in normal form, we can save one state by removing $[\hat{q}_{R,A}, \hat{q}_{R,B}]$ from Q_D. The modified machine will abort in the middle of the input, if *both* simulations, running in parallel, abort. □

In the above theorem, the cost of union for two DPDAs is dominated by the number of states, bounded by $O(\|Q_A\| \cdot h_A \cdot \|Q_B\| \cdot \|\Gamma_B^{\leq h_B}\|) \leq O(\|Q_A\| \cdot h_A \cdot \|Q_B\| \cdot \|\Gamma_B\|^{h_B})$, provided that $\|\Gamma_B\| \geq 2$. This reflects the fact that we keep the current states for both A and B in the finite control state, together with the entire content of the pushdown store for B, but only the pushdown height for A. This value can be slightly reduced for intersection, to $O(\|Q_A\| \cdot \|Q_B\| \cdot \|\Gamma_B\|^{h_B})$, if we do not mind aborting computations. The roles of A and B can be swapped, and hence the cost is exponential in $h = \min\{h_A, h_B\}$.

The rest of this section is devoted to proving that this exponential cost cannot be avoided. To this end, for arbitrary input alphabet Σ, we first exhibit $\{L'_n\}_{n\geq 1}$ and $\{L''_n\}_{n\geq 1}$, two families of languages accepted by DPDAs using $2 \cdot \|\Sigma\|+2$ states and $\|\Sigma\|-1$ pushdown symbols, with pushdown height $h = n$. Afterward, we show that any DPDA for the intersection of these languages must use either $(\|\Sigma\|-1)^{n/9-O(\log n)}$ states or its pushdown height must be above $(\|\Sigma\|-1)^{n/9-O(\log n)}$. An exponential lower bound $(\|\Sigma\|-1)^{n/15-O(\log n)}$ for union then follows easily, by De Morgan's laws.

First, for arbitrarily given input alphabet Σ, we fix one symbol $\$ \in \Sigma$ as special, and set $\Sigma_{\neg\$} = \Sigma \setminus \{\$\}$. Then, for each integer $n \geq 1$, we define three languages on Σ:

$$\begin{aligned} L'_n &= \{\phi u_1 \$ w_1 \$ u_2^R \$ w_2^R : \phi, u_1, w_1, u_2, w_2 \in \Sigma_{\neg\$}^*, \ |\phi u_1| \leq n, \ u_1 = u_2\}, \\ L''_n &= \{u_1 \$ \phi w_1 \$ u_2^R \$ w_2^R : \phi, u_1, w_1, u_2, w_2 \in \Sigma_{\neg\$}^*, \ |\phi w_1| \leq n, \ w_1 = w_2\}, \\ L_n &= L'_n \cap L''_n. \end{aligned} \tag{1}$$

If we consider only inputs in a special restricted form, the conditions for membership in these languages can be simplified as follows.

Lemma 4.3. *For each $u_1, w_1, u_2, w_2 \in \Sigma_{\neg\$}^*$ satisfying $|u_1| = |u_2| \leq n$ and $|w_1| = |w_2| \leq n$,*

- $u_1 \$ w_1 \$ u_2^R \$ w_2^R \in L'_n$ *if and only if* $u_1 = u_2$,
- $u_1 \$ w_1 \$ u_2^R \$ w_2^R \in L''_n$ *if and only if* $w_1 = w_2$,
- $u_1 \$ w_1 \$ u_2^R \$ w_2^R \in L_n$ *if and only if* $u_1 = u_2$ *and* $w_1 = w_2$.

Proof. We content ourselves with an argument for L'_n; the reasoning for L''_n is similar and the third statement is then trivial.

If $u_1 = u_2$, then, using $\phi = \varepsilon$ and taking $|u_1| = |u_2| \leq n$ for granted, we get $u_1 \$ w_1 \$ u_2^R \$ w_2^R \in L'_n$.

Conversely, if we have $u_1 \neq u_2$ for two strings satisfying $|u_1| = |u_2|$, then u_1 cannot be partitioned into $u_1 = \phi \hat{u}_1$ satisfying $\hat{u}_1 = u_2$, for no $\phi \in \Sigma_{\neg\$}^*$, and hence $u_1 \$ w_1 \$ u_2^R \$ w_2^R \notin L'_n$. □

All additional conditions in (1), not mentioned in the itemization presented in Lemma 4.3, were introduced just for technical reasons, making life simpler for machines accepting L'_n or L''_n with a help of a pushdown of constant height $h = n$. The additional conditions ensure that such machines do not have to verify if certain segments are of "correct" lengths, which reduces the number of states and, subsequently, increases the derived exponential gap between L'_n, L''_n and their intersection.

Lemma 4.4. *For any given Σ and each $n \geq 1$, the languages L'_n and L''_n can be accepted by* DPDAs *A_1 and A_2 using $2 \cdot \|\Sigma\| + 2$ states and $\|\Sigma\| - 1$ pushdown symbols, with pushdown height $h = n$.*

Proof. We present only the main features of A_1 and A_2, the formal definition may be easily filled by the reader.

First, on input $\phi u_1 \$ w_1 \$ u_2^R \$ w_2^R$, the DPDA A_1 for L'_n pushes the string ϕu_1 onto the pushdown (not taking care of a boundary between ϕ and u_1). If $|\phi u_1| > n$, it rejects by pushdown overflow. Otherwise, it sweeps w_1 and reaches the second occurrence of \$ along the input. At this point, A_1 checks whether $u_2 = u_1$ by scanning u_2^R and matching input symbols against the topmost $|u_2^R|$ symbols in the pushdown. If A_1 finds a pair of mismatched symbols, it rejects by undefined pop transition. If $|u_2^R| > |\phi u_1|$, it rejects by pushdown underflow. Otherwise, i.e., if all symbols match successfully until the point when A_1 scans the third occurrence of \$, A_1 consumes w_2^R, the remaining part of the input, by a cycle executed in an accepting state. (Note that A_1 leaves a copy of ϕ in the pushdown.)

Second, the DPDA A_2 for L''_n basically runs in the same way as A_1, but pushdown loading and input matching are now performed on ϕw_1 and w_2^R, respectively, in order to check whether $w_2 = w_1$.

Carefully implemented, both machines use $2 \cdot \|\Sigma\| + 2$ states, $\|\Sigma\| - 1$ pushdown symbols, and $h = n$ as a pushdown height.[4] □

We are now ready to establish a lower bound for constant height DPDAs accepting L_n. Before doing so, let us present the main idea behind the proof of Theorem 4.5.

Take a computation path of a constant height DPDA A_n accepting L_n, on an input $u \$ w \$ u^R \$ w^R$. For this path, let $\gamma_u \cdot \gamma_w$ be the pushdown content at the moment when the machine reads the second occurrence of \$. Here γ_u denotes the initial portion from the bottom, loaded into the pushdown in the course of reading u, while γ_w denotes the top part, loaded in the course of reading w. In general, the machine could also store some information about u and w in the finite state control. However, if both the number of states and the pushdown height are polynomial in n, we can fix, by a complicated pigeonhole argument, two pairs of strings $\dot{u} \neq \ddot{u}$ and $\dot{w} \neq \ddot{w}$, such that all four inputs with $u \in \{\dot{u}, \ddot{u}\}$ and $w \in \{\dot{w}, \ddot{w}\}$ share some common properties. Namely, the corresponding four computations must pass through the same states at several fixed "critical" points. Therefore, (i) to compare $\dot{u}^R$ or $\ddot{u}^R$ with the first input block $u \in \{\dot{u}, \ddot{u}\}$, the machine cannot use any information that was stored in the finite state control in the course of reading $u \$ w \$$. (Thus, to compare these blocks, the machine is forced to examine $\gamma_u \cdot \gamma_w$, stored in the pushdown.) In addition, (ii) the machine cannot compare $\dot{w}^R$ or $\ddot{w}^R$ with the second input block $w \in \{\dot{w}, \ddot{w}\}$ by the use of information that was stored in the finite state control in the course of reading $u \$ w \$ u^R \$$. (Thus, to compare $\dot{w}^R$ or $\ddot{w}^R$ with w, the machine must examine what was left in the pushdown store after reading the block $\$ u^R \$$.)

This gives us the following possibilities. First, in the course of reading $\$ u^R \$$, the machine does examine the deepest $|\gamma_u|$ symbols of the pushdown content $\gamma_u \cdot \gamma_w$. But then the top pushdown string γ_w is forgotten, and hence the machine cannot

[4] Using a transition function in a form introduced in standard textbooks, only $O(1)$ states would be required in our DPDAs. However, this would be paid by $\Theta(\|\Sigma\|^2)$ transitions, introduced in a somewhat artificial way. Moreover, neither "$O(1)$" nor "$\Theta(\|\Sigma\|^2)$" reflects the fact that the size-cost is actually linear in $\|\Gamma\| = \|\Sigma\| - 1$. Thus, our definition of transitions is more realistic if the size-cost is at stake.

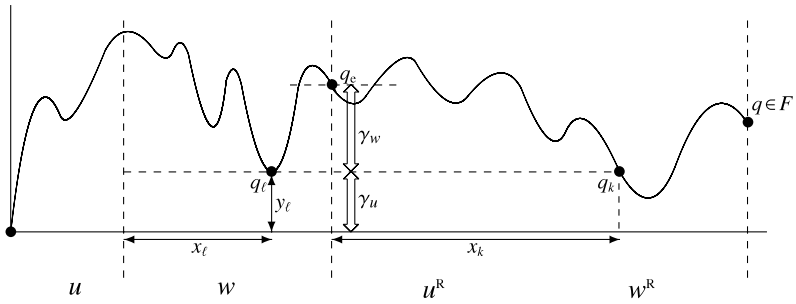

Fig. 5. Parameters and pushdown content along the computation.

distinguish between the inputs $\dot{u}\$\dot{w}\$\dot{u}^R\$\dot{w}^R$ and $\dot{u}\$\ddot{w}\$\dot{u}^R\$\dot{w}^R$. Second, in the course of reading $\$u^R\$$, the machine does not examine the deepest $|\gamma_u|$ symbols of $\gamma_u \cdot \gamma_w$. But then the bottom string γ_u is not utilized in the course of reading $\$u^R\$$, and hence the machine cannot distinguish between the inputs $\dot{u}\$\dot{w}\$\dot{u}^R\$\dot{w}^R$ and $\dot{u}\$\dot{w}\$\ddot{u}^R\$\dot{w}^R$. (Actually, we have two subcases here, depending on whether the deepest $|\gamma_u|$ symbols are examined too late, in the course of reading $\$w^R$, or not examined at all.) All these cases lead to contradictions.

Theorem 4.5. *For any given alphabet Σ, with $\|\Sigma\| \geq 3$, let $\{L_n\}_{n\geq 1}$ be the language family introduced by (1). Let $\{A_n\}_{n\geq 1}$ be any constant height* DPDA*s accepting these languages, and let $\|Q_n\|$ and h_n be, respectively, the number of states and the pushdown height in A_n. Then $(h_n+1)\cdot\|Q_n\|^2 > (\|\Sigma\|-1)^{\lfloor n/3\rfloor}/(4n^2)$, for each $n \geq 7$. (This lower bound does not depend on the cardinality of Γ_n, the pushdown alphabet used by A_n.)*

Consequently, in $\{A_n\}_{n\geq 1}$, the number of states and the pushdown height cannot be both polynomial in n; either h_n+1 or $\|Q_n\|$ (or both these values) are above $(\|\Sigma\|-1)^{\lfloor n/3\rfloor/3}/\sqrt[3]{4n^2} \geq (\|\Sigma\|-1)^{n/9-O(\log n)}$.

Proof. Assume, for contradiction, the existence of $A_n = \langle Q_n, \Sigma, \Gamma_n, H_n, q_{\mathrm{l},n}, F_n, h_n\rangle$, a constant height DPDA accepting L_n, with $p_n = (h_n+1)\cdot\|Q_n\|^2 \leq \|\Sigma_{\neg}\|^{\lfloor n/3\rfloor}/(4n^2)$, for each n. (Recall that $\|\Sigma_{\neg}\| = \|\Sigma\|-1$.) From now on, for the sake of readability, we simply write p instead of p_n, as well as A, Q, Γ, H, q_{l}, F, h instead of A_n, Q_n, Γ_n, H_n, $q_{\mathrm{l},n}$, F_n, h_n. (To avoid confusion, we keep L_n.) Thus, by assumption,

$$p = (h+1)\cdot\|Q\|^2 \leq \|\Sigma_{\neg}\|^{\lfloor n/3\rfloor}/(4n^2). \tag{2}$$

First, we define the set

$$V_0 = \{[u, w] : u, w \in \Sigma_{\neg}^*, |u| = \lfloor n/3\rfloor \text{ and } |w| = n\}.$$

Clearly,

$$\|V_0\| = \|\Sigma_{\neg}\|^{\lfloor n/3\rfloor}\cdot\|\Sigma_{\neg}\|^n = \|\Sigma_{\neg}\|^{n+\lfloor n/3\rfloor}.$$

Let us consider the computation of A on an input $u\$w\$u^R\$w^R$, where $[u, w] \in V_0$. By Lemma 4.3, this string is in L_n, and hence, for each $[u, w] \in V_0$, we have an accepting computation path reading the entire string $u\$w\$u^R\$w^R$.

For this path, the following parameters are taken: (i) $y_\ell \in \{0, \ldots, h\}$, the lowest height of pushdown store in the course of reading w, (ii) $q_\ell \in Q$, the state in which the height y_ℓ is attained for the last time, along w, (iii) $x_\ell \in \{0, \ldots, |w|\}$, the distance from the beginning of w to the input position in which q_ℓ is entered, (iv) $q_k \in Q$, the state in which the height y_ℓ is decreased to $y_\ell - 1$ for the first time, in the course of reading $\$u^R\$w^R$, (v) $x_k \in \{0, \ldots, |\$u^R\$w^R|\}$, the distance from the beginning of $\$u^R\$w^R$ to the input position in which q_k is entered. (If the machine A, along $\$u^R\$w^R$, never decreases the pushdown height to $y_\ell - 1$, we take $x_k = |\$u^R\$w^R| + 1$ and, by definition, the state q_k coincides with $q \in F$ at the end of the path. That is, the artificial position x_k is behind the input, never reached by A, representing "infinity" for the purposes of our reasoning. Except for this special case, the pushdown height is always equal to y_ℓ at the position x_k.) In addition, upon reading the second occurrence of the symbol \$ in the input word (i.e., at the end of the string w), we let: (vi) q_{e} be the starting state for the transition reading the second occurrence of \$, and (vii) $\gamma_u\cdot\gamma_w$ be the pushdown content at this moment, where γ_u denotes the initial portion from the bottom, of height y_ℓ, while γ_w is the remaining part. For better understanding, see Fig. 5.

Taking into account the lengths of w and $\$u^R\$w^R$, including the artificial position $x_k = |\$u^R\$w^R| + 1$ but excluding $x_k = 0$ (since the second part of the computation starts by reading the symbol \$ in the state q_{e}), we have that the number of different quintuples $[x_\ell, x_k, y_\ell, q_\ell, q_k]$ is bounded by $(n+1)\cdot(n+\lfloor n/3\rfloor+3)\cdot(h+1)\cdot\|Q\|^2$. It is easy to verify that $(n+1)\cdot(n+\lfloor n/3\rfloor+3) \leq 2n^2$, for each $n \geq 7$. Thus, by using also (2), the number of such quintuples can be bounded by $2n^2\cdot(h+1)\cdot\|Q\|^2 \leq 2n^2\cdot\|\Sigma_{\neg}\|^{\lfloor n/3\rfloor}/(4n^2) = \|\Sigma_{\neg}\|^{\lfloor n/3\rfloor}/2$.

Indeed, since $u\$w\$u^R\$w^R$ is in L_n for each $[u, w] \in V_0$, each pair $[u, w] \in V_0$ is associated with an accepting computation path, and also with one of these quintuples. Hence, a simple pigeonhole argument proves the existence of a set $V_1 \subseteq V_0$, with

$$\|V_1\| \geq \frac{\|V_0\|}{\|\Sigma_{\neg}\|^{\lfloor n/3\rfloor}/2} \geq \frac{\|\Sigma_{\neg}\|^{n+\lfloor n/3\rfloor}}{\|\Sigma_{\neg}\|^{\lfloor n/3\rfloor}/2} = 2\cdot\|\Sigma_{\neg}\|^n,$$

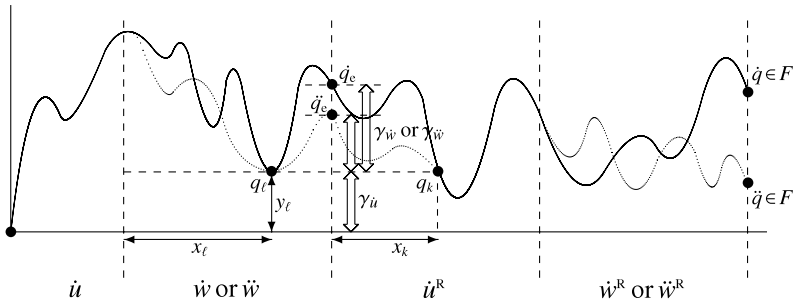

Fig. 6. Computation on inputs $\dot{u}\$\dot{w}\$\dot{u}^R\$\dot{w}^R \in L_n$ (solid line), $\dot{u}\$\ddot{w}\$\dot{u}^R\$\ddot{w}^R \in L_n$ (dotted line), and $\dot{u}\$\ddot{w}\$\dot{u}^R\$\dot{w}^R \notin L_n$, visiting the lower y_ℓ pushdown symbols in the course of reading the third segment.

such that all string pairs $[u, w] \in V_1$ share the same quintuple $[x_\ell, x_k, y_\ell, q_\ell, q_k]$. Let us now build the following "projection" sets from V_1:

$$V_{1,1} = \{u : \exists w, \text{ such that } [u, w] \in V_1\},$$
$$V_{1,2} = \{w : \exists u, \text{ such that } [u, w] \in V_1\}.$$

Clearly, $\|V_{1,1}\| \leq \|\Sigma_{-1}\|^{\lfloor n/3 \rfloor}$ and $\|V_{1,2}\| \leq \|\Sigma_{-1}\|^n$. Note also that $V_1 \subseteq V_{1,1} \times V_{1,2}$, with $\|V_1\| > 0$, and hence also with $\|V_{1,1}\| > 0$ and $\|V_{1,2}\| > 0$.

Now, let us split $V_{1,2}$ into the following two disjoint sets:

$$V_2 = \{w : \text{ there exist more than one } u \in V_{1,1} \text{ such that } [u, w] \in V_1\},$$
$$V_2^c = \{w : \text{ there exists exactly one } u \in V_{1,1} \text{ such that } [u, w] \in V_1\}.$$

Let $x = \|V_2\|$. First, it is easy to see that $\|V_2^c\| = \|V_{1,2}\| - x \leq \|\Sigma_{-1}\|^n - x$. To evaluate a lower bound for $x = \|V_2\|$, observe that $V_1 \subseteq V_{1,1} \times V_{1,2} = (V_{1,1} \times V_2^c) \cup (V_{1,1} \times V_2)$, but the contribution of elements from $V_{1,1} \times V_2^c$ to the total number of elements in V_1 must be very "small". Consequently, the contribution from $V_{1,1} \times V_2$ must be "large". More precisely, observe that if $w \in V_2^c$, then it is impossible to have $[\dot{u}, w] \in V_1$ and $[\ddot{u}, w] \in V_1$ for two different strings $\dot{u}, \ddot{u} \in V_{1,1}$. On the other hand, if $w \in V_2$, the number of pairs $[u, w] \in V_1$ sharing this w is bounded only by $\|V_{1,1}\| \leq \|\Sigma_{-1}\|^{\lfloor n/3 \rfloor}$. This gives us that $\|V_1\| \leq 1 \cdot \|V_2^c\| + \|V_{1,1}\| \cdot \|V_2\| \leq (\|\Sigma_{-1}\|^n - x) + \|\Sigma_{-1}\|^{\lfloor n/3 \rfloor} \cdot x = x \cdot (\|\Sigma_{-1}\|^{\lfloor n/3 \rfloor} - 1) + \|\Sigma_{-1}\|^n$, whence

$$\|V_2\| = x \geq \frac{\|V_1\| - \|\Sigma_{-1}\|^n}{\|\Sigma_{-1}\|^{\lfloor n/3 \rfloor} - 1} > \frac{2 \cdot \|\Sigma_{-1}\|^n - \|\Sigma_{-1}\|^n}{\|\Sigma_{-1}\|^{\lfloor n/3 \rfloor}} = \|\Sigma_{-1}\|^{n - \lfloor n/3 \rfloor}. \tag{3}$$

(This requires $\|\Sigma_{-1}\|^{\lfloor n/3 \rfloor} > 1$, which clearly holds for $\|\Sigma_{-1}\| \geq 2$ and $n \geq 7$.)

Let us fix some lexicographical order on the set $V_{1,1}$ and, for each $w \in V_2$, take the first two $\dot{u} \neq \ddot{u}$ in this ordering satisfying $[\dot{u}, w] \in V_1$ and $[\ddot{u}, w] \in V_1$. The number of such $\dot{u} \neq \ddot{u}$ is clearly bounded by $\|V_{1,1} \times V_{1,1}\| \leq \|\Sigma_{-1}\|^{2 \cdot \lfloor n/3 \rfloor}$. This, together with (3) and a pigeonhole argument, yields the existence of a nonempty set $V_3 \subseteq V_2 \subseteq V_{1,2}$, with cardinality

$$\|V_3\| \geq \frac{\|V_2\|}{\|V_{1,1} \times V_{1,1}\|} > \frac{\|\Sigma_{-1}\|^{n - \lfloor n/3 \rfloor}}{\|\Sigma_{-1}\|^{2 \cdot \lfloor n/3 \rfloor}} = \|\Sigma_{-1}\|^{n - 3 \cdot \lfloor n/3 \rfloor} \geq \|\Sigma_{-1}\|^0 = 1,$$

such that all strings $w \in V_3$ share the same pair $\dot{u} \neq \ddot{u}$ in $V_{1,1}$, with $[\dot{u}, w] \in V_1$ and $[\ddot{u}, w] \in V_1$.

Note that $\|V_3\| > 1$ implies $\|V_3\| \geq 2$. Thus, we can fix some $\dot{w} \neq \ddot{w}$ in V_3, such that $[\dot{u}, \dot{w}]$, $[\ddot{u}, \dot{w}]$, $[\dot{u}, \ddot{w}]$, $[\ddot{u}, \ddot{w}] \in V_1$. Moreover, by definition of V_1, we have that all these string pairs share the same quintuple $[x_\ell, x_k, y_\ell, q_\ell, q_k]$. Therefore, if we take the corresponding inputs $u\$w\$u^R\$w^R$, for any $u \in \{\dot{u}, \ddot{u}\}$ and any $w \in \{\dot{w}, \ddot{w}\}$, all four of them are in L_n, by Lemma 4.3, and all four of them have accepting computation paths passing through the same states q_ℓ, q_k, placed at the same positions x_ℓ, x_k along the input, with the same pushdown height y_ℓ. Depending on the value x_k, there are now the following three cases to consider.

Case 1: $x_k \in \{0, \ldots, \lfloor n/3 \rfloor + 1\}$, *that is, the four computations visit the lower content of the pushdown store below the level* y_ℓ *in the course of reading* $\$\dot{u}^R$ *or* $\$\ddot{u}^R$. This situation is depicted in Fig. 6.

Consider the inputs $\dot{z} = \dot{u}\$\dot{w}\$\dot{u}^R\$\dot{w}^R$ and $\ddot{z}_w = \dot{u}\$\ddot{w}\$\dot{u}^R\$\ddot{w}^R$, together with their crossbreed $\delta_w = \dot{u}\$\ddot{w}\$\dot{u}^R\$\dot{w}^R \notin L_n$.

First, at the moment when the machine A gets to the state q_ℓ at the position x_ℓ, the pushdown contains a word $\gamma_{\dot{u}}$, consisting of the deepest y_ℓ symbols loaded in the course of reading $\dot{u}\$$, and hence the same for both $\dot{z}$ and $\ddot{z}_w$. Depending on the first half of the input, namely, on $\dot{u}\$\dot{w}$ and $\dot{u}\$\ddot{w}$, the machine then gets, respectively, to some state $\dot{q}_e$ or $\ddot{q}_e$, with the pushdown content consisting of the word $\gamma_{\dot{u}}$ under some word $\gamma_{\dot{w}}$ or $\gamma_{\ddot{w}}$. However, on both $\dot{z}$ and $\ddot{z}_w$, A reaches the same state q_k at the same position x_k. Moreover, all information previously stored in the upper part of the pushdown (i.e., $\gamma_{\dot{w}}$ or $\gamma_{\ddot{w}}$, respectively) has been popped out, since we have reached the pushdown height y_ℓ. Thus, in both cases, the current content stored in the pushdown is the same, namely, the string $\gamma_{\dot{u}}$. So far, the computation on the input δ_w followed the

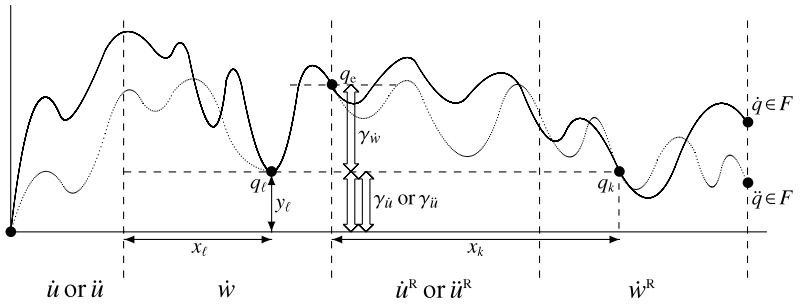

Fig. 7. Computation on inputs $\dot u\$\dot w\$\dot u^{\mathrm R}\$\dot w^{\mathrm R} \in L_n$ (solid line), $\ddot u\$\dot w\$\ddot u^{\mathrm R}\$\dot w^{\mathrm R} \in L_n$ (dotted line), and $\dot u\$\dot w\$\ddot u^{\mathrm R}\$\dot w^{\mathrm R} \notin L_n$, visiting the lower y_ℓ pushdown symbols in the course of reading the last segment.

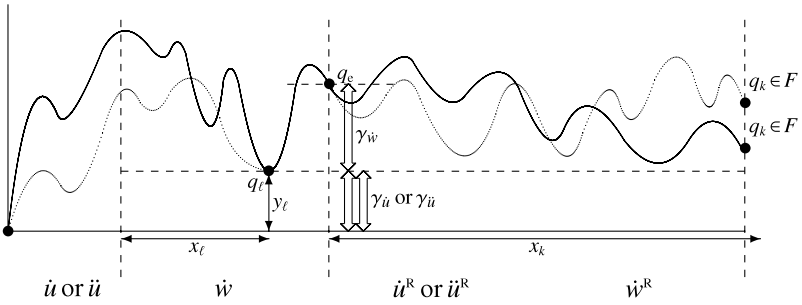

Fig. 8. Computation on inputs $\dot u\$\dot w\$\dot u^{\mathrm R}\$\dot w^{\mathrm R} \in L_n$ (solid line), $\ddot u\$\dot w\$\ddot u^{\mathrm R}\$\dot w^{\mathrm R} \in L_n$ (dotted line), and $\dot u\$\dot w\$\ddot u^{\mathrm R}\$\dot w^{\mathrm R} \notin L_n$, not visiting the lower y_ℓ pushdown symbols in the course of reading the second half of the input.

trajectory for $\ddot z_w$, reading $\dot u\$\ddot w$ and the first x_k symbols of $\$\dot u^{\mathrm R}$. Hence, also here we must come to the same state q_k with the same pushdown content $\gamma_{\ddot u}$.

From this point forward, the computation on δ_w follows the trajectory for $\dot z$, reading the remaining part of $\$\dot u^{\mathrm R}$, the string $\$\dot w^{\mathrm R}$, and stopping in some accepting[5] state $\dot q \in F$.

Thus, the machine A accepts $\delta_w = \dot u\$\ddot w\$\dot u^{\mathrm R}\$\dot w^{\mathrm R} \notin L_n$, which is a contradiction.

Case II: $x_k \in \{\lfloor n/3\rfloor+2, \ldots, n+\lfloor n/3\rfloor+2\}$, *that is, the four computations visit the lower content of the pushdown store below the level* y_ℓ *in the course of reading* $\$\dot w^{\mathrm R}$ *or* $\$\ddot w^{\mathrm R}$. This situation is depicted in Fig. 7.

This time we consider the inputs $\dot z = \dot u\$\dot w\$\dot u^{\mathrm R}\$\dot w^{\mathrm R}$ and $\ddot z_u = \ddot u\$\dot w\$\ddot u^{\mathrm R}\$\dot w^{\mathrm R}$, together with $\delta_u = \dot u\$\dot w\$\ddot u^{\mathrm R}\$\dot w^{\mathrm R} \notin L_n$.

The computation on the input $\dot z$ begins in the same way as in the previous case: when the machine A gets to the state q_ℓ at the position x_ℓ, the pushdown store contains a word $\gamma_{\dot u}$, loaded in the course of reading $\dot u\$$. On the other hand, the input $\ddot z_u$ begins with $\ddot u\$$, and hence the corresponding computation on $\ddot z_u$ enters the state q_ℓ with a word $\gamma_{\ddot u}$ in the pushdown, satisfying $|\gamma_{\ddot u}| = |\gamma_{\dot u}| = y_\ell$, at the same input position x_ℓ. The computation on the input δ_u follows the trajectory for $\dot z$ and enters q_ℓ with $\gamma_{\dot u}$ in the pushdown, reading $\dot u\$$ and the first x_ℓ symbols of $\dot w$.

After that, the deepest y_ℓ symbols in the pushdown (i.e., $\gamma_{\dot u}$ or $\gamma_{\ddot u}$, respectively) cannot come into play until the moment when the pushdown height gets below the level y_ℓ again. On both $\dot z$ and $\ddot z_u$, this will happen in the state q_k, at the position x_k from the beginning of $\$\dot u^{\mathrm R}\$\dot w^{\mathrm R}$ or $\$\ddot u^{\mathrm R}\$\dot w^{\mathrm R}$. Thus, A enters q_k with the respective string $\gamma_{\dot u}$ or $\gamma_{\ddot u}$ in the pushdown, with the input head pointing into $\$\dot w^{\mathrm R}$. Therefore, A gets from q_ℓ to q_k by reading the following portion of the input: the remaining part of $\dot w$, the entire middle string (i.e., $\$\dot u^{\mathrm R}$ or $\$\ddot u^{\mathrm R}$, respectively), and the first $x_k - |\$\dot u^{\mathrm R}|$ symbols of $\$\dot w^{\mathrm R}$.

In between q_ℓ and q_k, the computation on δ_u follows the trajectory for $\ddot z_u$, reading the same symbols along the input. However, along this path, the configurations differ from the original ones in the deepest y_ℓ pushdown symbols: $\gamma_{\ddot u} \cdot \gamma$ is replaced everywhere by $\gamma_{\dot u} \cdot \gamma$. Thus, the computation on δ_u enters q_k with $\gamma_{\dot u}$ in the pushdown.

At this point, the configuration on δ_u agrees with that on $\dot z$. Therefore, the computation on δ_u switches back to the trajectory for $\dot z$, reading the remaining part of the string $\$\dot w^{\mathrm R}$ and stopping in some accepting state $\dot q \in F$.

Thus, A accepts $\delta_u = \dot u\$\dot w\$\ddot u^{\mathrm R}\$\dot w^{\mathrm R} \notin L_n$, which is a contradiction.

Case III: $x_k = n+\lfloor n/3\rfloor+3$, *that is, after passing through* $\dot w$ *or* $\ddot w$, *the four computations never visit the lower content of the pushdown store below the level* y_ℓ. This situation is depicted in Fig. 8.

Also in this case we consider $\dot z = \dot u\$\dot w\$\dot u^{\mathrm R}\$\dot w^{\mathrm R}$ and $\ddot z_u = \ddot u\$\dot w\$\ddot u^{\mathrm R}\$\dot w^{\mathrm R}$, together with the same $\delta_u = \dot u\$\dot w\$\ddot u^{\mathrm R}\$\dot w^{\mathrm R} \notin L_n$.

As already discussed above, the machine A reaches the same state q_ℓ at the same position x_ℓ on both $\dot z$ and $\ddot z_u$, with a respective string $\gamma_{\dot u}$ or $\gamma_{\ddot u}$ in the pushdown. The computation on the input δ_u follows the trajectory for $\dot z$ and enters q_ℓ with $\gamma_{\ddot u}$ in the pushdown, reading $\dot u\$$ and the first x_ℓ symbols of $\dot w$.

[5] The computation on $\ddot z_w$ ends in some $\ddot q \in F$.

After that, by definition of q_ℓ and the assumptions for Case III, the deepest y_ℓ symbols in the pushdown (i.e., $\gamma_{\ddot{u}}$ or $\gamma_{\ddot{w}}$, respectively) cannot come into play any more. Therefore, the computation on δ_u switches to the trajectory for $\ddot{z}_u$, reading the same symbols along the input, namely, the remaining part of $\dot{w}$ and the entire string $\$\ddot{u}^R\$\dot{w}^R$. However, along this path, the configurations differ from the original ones in the deepest y_ℓ pushdown symbols: $\gamma_{\ddot{w}} \cdot \gamma$ is replaced everywhere by $\gamma_{\ddot{u}} \cdot \gamma$. Nevertheless, the computation on δ_u ends in the same[6] state $q_k \in F$.

Thus, also in this case A accepts $\delta_u \notin L_n$, which is a contradiction.

In conclusion, if a DPDA $A = A_n$ does accept the language L_n, the inequality (2) must be reversed. Thus, the value $p = p_n$ must satisfy $p_n = (h_n+1) \cdot \|Q_n\|^2 > \|\Sigma_{-1}\|^{\lfloor n/3 \rfloor}/(4n^2) = (\|\Sigma\|-1)^{\lfloor n/3 \rfloor}/(4n^2)$. Consequently, either the number of states or the pushdown height (or both) must be exponential in n, since either $\|Q_n\| > (\|\Sigma\|-1)^{\lfloor n/3 \rfloor/3}/\sqrt[3]{4n^2}$, or else $h_n+1 > (\|\Sigma\|-1)^{\lfloor n/3 \rfloor/3}/\sqrt[3]{4n^2} \geq (\|\Sigma\|-1)^{n/9-2/9}/\sqrt[3]{4n^2} \geq (\|\Sigma\|-1)^{n/9-O(\log n)}$. □

We end this section by showing also an exponential blow-up for union of two DPDAs, leaving possible improvements to the reader.

Theorem 4.6. *For any given alphabet Σ, with $\|\Sigma\| \geq 3$, let $\{L_n'^{\,c}\}_{n\geq 1}$ and $\{L_n''^{c}\}_{n\geq 1}$ be the complements of the language families introduced by (1). Then there exist constant height* DPDAs *$\{A_n'\}_{n\geq 1}$ and $\{A_n''\}_{n\geq 1}$ accepting the respective languages with size polynomial in $\|\Sigma\|$ and n. Namely, $O(\|\Sigma\|\cdot n)$ states and $\|\Sigma\|-1$ pushdown symbols are sufficient, using the pushdown height $h = n$.*

On the other hand, let $\{\hat{A}_n\}_{n\geq 1}$ be any constant height DPDAs *accepting their union $\{\hat{L}_n\}_{n\geq 1} = \{L_n'^{\,c} \cup L_n''^{c}\}_{n\geq 1}$, and let $\|\hat{Q}_n\|$ and $\hat{h}_n$ be, respectively, the number of states and the pushdown height in $\hat{A}_n$. Then $(\hat{h}_n+1)^3 \cdot \|\hat{Q}_n\|^2 > (\|\Sigma\|-1)^{\lfloor n/3 \rfloor}/(8n^2)$, for each $n \geq 7$. Consequently, either $\hat{h}_n+1$ or $\|\hat{Q}_n\|$ (or both) are above $(\|\Sigma\|-1)^{\lfloor n/3 \rfloor/5}/\sqrt[5]{8n^2} \geq (\|\Sigma\|-1)^{n/15-O(\log n)}$.*

Proof. First, for each $n \geq 1$, both L_n' and L_n'' can be accepted by DPDAs with at most $2 \cdot \|\Sigma\|+2$ states and $\|\Sigma\|-1$ pushdown symbols, using $h = n$ as the pushdown height, by Lemma 4.4. But then, by Theorem 3.2, both $L_n'^{\,c}$ and $L_n''^{c}$ can be accepted by DPDAs with at most $(2 \cdot \|\Sigma\|+2)(n+1)+1 \leq O(\|\Sigma\|\cdot n)$ states, using the same pushdown alphabet and the same pushdown height.

On the other hand, assume that $\hat{L}_n = L_n'^{\,c} \cup L_n''^{c}$ is accepted by a constant height DPDA $\hat{A}_n$ with $\|\hat{Q}\|$ states and a pushdown height $\hat{h}$. (The machine $\hat{A}_n$ may use an arbitrary pushdown alphabet.) So, by complementing $\hat{A}_n$ by the use of Theorem 3.2, we obtain a constant height DPDA A_n for L_n, with the number of states bounded by $\|Q\| \leq \|\hat{Q}\|(\hat{h}+1)+1$, and the same pushdown height $\hat{h}$. But then, by Theorem 4.5, we get that $(\hat{h}+1)\|Q\|^2 > (\|\Sigma\|-1)^{\lfloor n/3 \rfloor}/(4n^2)$. Utilizing also simple inequality $2x^2 \geq (x+1)^2$, valid for $x \geq 3$, we then get:

$$\begin{aligned}(\hat{h}+1)^3 \cdot \|\hat{Q}\|^2 &= \frac{1}{2}\cdot(\hat{h}+1)(2 \cdot \|\hat{Q}\|^2(\hat{h}+1)^2) \geq \frac{1}{2}\cdot(\hat{h}+1)(\|\hat{Q}\|(\hat{h}+1)+1)^2 \\ &\geq \frac{1}{2}\cdot(\hat{h}+1)\|Q\|^2 > \frac{1}{2}\cdot(\|\Sigma\|-1)^{\lfloor n/3 \rfloor}/(4n^2) \\ &= (\|\Sigma\|-1)^{\lfloor n/3 \rfloor}/(8n^2)\,.\end{aligned}$$

Consequently, we must get either $\|\hat{Q}\| > (\|\Sigma\|-1)^{\lfloor n/3 \rfloor/5}/\sqrt[5]{8n^2}$, or $\hat{h}+1 > (\|\Sigma\|-1)^{\lfloor n/3 \rfloor/5}/\sqrt[5]{8n^2} \geq (\|\Sigma\|-1)^{n/15-O(\log n)}$. □

5. Concluding remarks

We have analyzed the size-cost of Boolean operations for constant height DPDAs. It turns out that the size complexity of all basic Boolean operations for DPDAs with constant height pushdown reflects the closure properties of DPDAs with unrestricted pushdown for the corresponding operations, thus, the closure properties of the deterministic context-free languages (DCF, for short). This is emphasized in Table 1.

On the one hand, it is well known [2] (see also the middle column of Table 1) that the class DCF is closed under complement, but not closed under intersection and union. However, DCF is closed under some simpler operations, namely, intersection and union with regular languages. We also know that DCF contains nonregular languages and hence, in general, a DPDA with an unrestricted pushdown cannot be converted into an equivalent DFA.

On the other hand (see the right column of Table 1), DPDAs with a pushdown of constant height can recognize regular languages only, and hence they are closed under all above operations. But, taking into account their size, we have got that the cost of complement is polynomial but the costs of intersection and union are exponential. For simpler operations, intersection and union of DPDAs with DFAs, the costs are polynomial. Finally, the cost of converting a constant height DPDA into an equivalent DFA is exponential [1]. (See also Lemma 2.2.)

So far, we have spotted only one difference in this analogy, namely, intersection with deterministic counter languages: by Theorem 4.2, given two constant height DPDAs A and B, a new DPDA for their intersection does not need more than

[6] The computation starting from q_ℓ on $\dot{z}$ ends also in the same state q_k (not necessarily with the same pushdown height), but by reading a different sequence of input symbols, not corresponding to the second half of δ_u.

Table 1
Analogies between closure properties and size-cost of unrestricted and constant height DPDAs. The results in the middle column are standard (see, e.g., [2]), those in the right column are proved in this paper, except for the conversion to DFA [1].

Operation	Unrestricted DPDAs *closure/possibility*	Constant height DPDAs *size-cost*
Complementation	Closed	Polynomial
Intersection	Not closed	Exponential
Union	Not closed	Exponential
Intersection with DFA	Closed	Polynomial
Union with DFA	Closed	Polynomial
Conversion to DFA	Not possible	Exponential

$\|Q_A\| \cdot \|Q_B\| \cdot \|\Gamma_B^{\leq h_B}\|$ states, using the pushdown alphabet Γ_A and the constant height h_A. But, if B uses its pushdown as a counter, with $\|\Gamma_B\| = 1$, we need only $\|Q_A\| \cdot \|Q_B\| \cdot (h_B + 1)$ states, and hence the cost remains polynomial. On the other hand, for machines with unrestricted counters, the intersection is not necessarily context-free, and hence not in DCF.[7]

Therefore, it would be interesting to investigate the size-cost of other language operations for constant height DPDAs and compare them with unrestricted versions of pushdown automata. Similarly, many properties of nondeterministic or two-way variants of these machines have not been examined yet.

Acknowledgments

The authors wish to thank the anonymous referees for useful and kind comments. The first and second authors were supported by the Slovak Grant Agency for Science under contract VEGA 1/0479/12 "Combinatorial Structures and Complexity of Algorithms" and the Slovak Research and Development Agency under contract APVV-0035-10 "Algorithms, Automata, and Discrete Data Structures". A preliminary version of this work was presented at the *13th Int. Conf. Descriptional Complexity of Formal Systems, Limburg, Germany, July 25–27, 2011.*

References

[1] V. Geffert, C. Mereghetti, B. Palano, More concise representation of regular languages by automata and regular expressions, Inform. and Comput. 208 (2010) 385–394.
[2] J. Hopcroft, R. Motwani, J. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 2001.
[3] M. Holzer, M. Kutrib, Nondeterministic descriptional complexity of regular languages, Internat. J. Found. Comput. Sci. 14 (2003) 1087–1102.
[4] S. Yu, Q. Zhuang, K. Salomaa, The state complexities of some basic operations on regular languages, Theoret. Comput. Sci. 125 (1994) 315–328.
[5] A. Ehrenfeucht, P. Zieger, Complexity measures for regular expressions, J. Comput. System Sci. 12 (1976) 134–146.
[6] H. Gruber, M. Holzer, Language operations with regular expressions of polynomial size, Theoret. Comput. Sci. 410 (2009) 3281–3289.
[7] M. Kutrib, A. Malcher, D. Wotschke, The Boolean closure of linear context-free languages, Acta Inform. 45 (2008) 177–191.
[8] A. Meyer, M. Fischer, Economy of description by automata, grammars, and formal systems, in: Proc. IEEE Symp. Switching & Automata Theory, 1971, pp. 188–91.

[7] As an example, consider $L = \{a^i b^j c^k : i \geq j\} \cap \{a^i b^j c^k : j \geq k\}$, an intersection of two languages accepted by DPDAs with unary pushdown alphabet (rejecting some inputs by pushdown underflow). To see that L is not context-free, take $a^n b^n c^n \in L$. This word could be partitioned by the Pumping Lemma into $uvwxy$ such that both uwy and $uvvwxxy$ should be in L (with $vx \neq \varepsilon$). But at least one of these two strings must not be in L.