Height-Deterministic Pushdown Automata

Dirk Nowotka[1] and Jiří Srba[2,⋆]

[1] Institut für Formale Methoden der Informatik
Universität Stuttgart, Germany
[2] BRICS⋆⋆, Department of Computer Science
Aalborg University, Denmark

Abstract. We define the notion of height-deterministic pushdown automata, a model where for any given input string the stack heights during any (nondeterministic) computation on the input are a priori fixed. Different subclasses of height-deterministic pushdown automata, strictly containing the class of regular languages and still closed under boolean language operations, are considered. Several such language classes have been described in the literature. Here, we suggest a natural and intuitive model that subsumes all the formalisms proposed so far by employing height-deterministic pushdown automata. Decidability and complexity questions are also considered.

1 Introduction

Visibly pushdown automata [3], a natural and well motivated subclass of pushdown automata, have been recently introduced and intensively studied [8,2,4]. The theory found a number of interesting applications, e.g. in program analysis [1,9] and XML processing [10]. The corresponding class of visibly pushdown languages is more general than regular languages while it still possesses nice closure properties and the language equivalence problem as well as simulation/bisimulation equivalences are decidable [3,11]. Several extensions [7,5] have been proposed in order to preserve these nice properties while describing a larger class of systems. These studies have been particularly motivated by applications in the field of formal verification. However, unlike the natural model of visibly pushdown automata, these extensions are rather technical and less intuitive.

In this paper we suggest the model of height-deterministic pushdown automata which strictly subsumes all the models mentioned above and yet possesses desirable closure and decidability properties. This provides a uniform framework for the study of more general formalisms.

The paper is organized as follows. Section 2 contains basic definitions. Section 3 introduces height-deterministic pushdown automata, or *hpda*. It studies the languages recognized by real-time and deterministic *hpda*, and proves a number of interesting closure properties. Section 4 shows that these classes properly contain the language class of [7] and the classes defined in [3] and [5].

⋆ Partially supported by the research center ITI, project No. 1M0021620808.
⋆⋆ Basic Research In Computer Science, Danish National Research Foundation.

L. Kučera and A. Kučera (Eds.): MFCS 2007, LNCS 4708, pp. 125–134, 2007.

2 Preliminaries

Let $\Sigma = \{a, b, c, \ldots\}$ be a finite set of *letters*. The set Σ^* denotes all finite words over Σ. The *empty word* is denoted by λ. A subset of Σ^* is called a *language*. Given a nonempty word $w \in \Sigma^*$ we write $w = w_{(1)}w_{(2)} \cdots w_{(n)}$ where $w_{(i)} \in \Sigma$ denotes the *i-th letter* of w for all $1 \le i \le n$. The *length* $|w|$ of w is n and $|\lambda| = 0$. By abuse of notation $|\cdot|$ also denotes the *cardinality* of a set, the *absolute value* of an integer, and the *size* of a pushdown automaton (see definition below). We denote by $\bullet w$ the word $w_{(2)}w_{(3)} \cdots w_{(n)}$, and define further $\bullet a = \lambda$ for every $a \in \Sigma$ and $\bullet\lambda = \lambda$. Finally, we let L^c abbreviate $\Sigma^* \setminus L$ for $L \subseteq \Sigma^*$.

Finite State Automata. A finite state automaton (*fsa*) R over Σ is a tuple $(S, \Sigma, s_0, \varrho, F)$ where $S = \{s, t, \ldots\}$ is a finite set of *states*, $s_0 \in S$ is the *initial state*, $\varrho \subseteq S \times \Sigma \times S$ is a set of *rules*, and $F \subseteq S$ is the set of final states. We call R a *deterministic finite state automaton* (*dfsa*) if for every $s \in S$ and every $a \in \Sigma$ there is exactly one $t \in \Sigma$ such that $(s, a, t) \in \varrho$, i.e., the relation ϱ can be understood as a function $\varrho\colon S \times \Sigma \to S$. Given a nonempty $w \in \Sigma^*$ we write $s \xrightarrow[R]{w} t$ (or just $s \xrightarrow{w} t$ if R is understood) if either $w \in \Sigma$ and $(s, w, t) \in \varrho$ or there exists an $s' \in S$ such that $(s, w_{(1)}, s') \in \varrho$ and $s' \xrightarrow{\bullet w} t$. We say that R *recognizes* the language $\mathcal{L}(R) = \{w \in \Sigma^* \mid s_0 \xrightarrow[R]{w} t, t \in F\}$. A language is *regular* if it is recognized by some *fsa*. The class of all regular languages is denoted by *REG*.

Finite State Transducers. A finite state transducer (*fst*) T from Σ^* to a monoid M (in this paper we have either $M = \Sigma^*$ or $M = \mathbb{Z}$), is a tuple $(S, \Sigma, M, s_0, \varrho, F)$ where $(S, \Sigma \times M, s_0, \varrho', F)$ is an *fsa* and $\varrho = \{(s, a, m, t) \mid (s, (a, m), t) \in \varrho'\}$. Given $w \in \Sigma^*$ and $m \in M$, we write $s \xrightarrow[T]{w,m} t$ (or $s \xrightarrow{w,m} t$ if T is understood) if either $w \in \Sigma$ and $(s, w, m, t) \in \varrho$ or if there exists an $s' \in S$ such that $(s, w_{(1)}, m_1, s') \in \varrho$, $s' \xrightarrow{\bullet w, m_2} t$ and $m = m_1 \oplus m_2$, where $\oplus$ is the operation associated with the monoid M. Given $L \subseteq \Sigma^*$, the *image of L under T*, denoted by $T(L)$, is the set of elements m such that $s_0 \xrightarrow[T]{w,m} t$ for some $t \in F$ and $w \in L$.

Pushdown Automata. A pushdown automaton (*pda*) A over an alphabet Σ is a tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where $Q = \{p, q, r, \ldots\}$ is a finite set of *states*, $\Gamma = \{X, Y, Z, \ldots\}$ is a finite set of *stack symbols* such that $Q \cap \Gamma = \emptyset$, $\delta \subseteq Q \times \Gamma \times (\Sigma \cup \{\varepsilon\}) \times Q \times \Gamma^* \cup Q \times \{\bot\} \times (\Sigma \cup \{\varepsilon\}) \times Q \times \Gamma^*\{\bot\}$ is a finite *set of rules*, where $\bot \notin \Gamma$ (empty stack) and $\varepsilon \notin \Sigma$ (empty input word) are special symbols, $q_0 \in Q$ is the *initial state*, and $F \subseteq Q$ is a set of *final states*. The *size* $|A|$ of a *pda* A is defined as $|Q| + |\Sigma| + |\Gamma| + \{|pXq\alpha| \mid (p, X, a, q, \alpha) \in \delta\}$. We usually write $pX \overset{a}{\underset{A}{\longmapsto}} q\alpha$ (or just $pX \overset{a}{\longmapsto} q\alpha$ if A is understood) for $(p, X, a, q, \alpha) \in \delta$. We say that a rule $pX \overset{a}{\underset{A}{\longmapsto}} q\alpha$ is a *push*, *internal*, or *pop* rule if $|\alpha| = 2, 1,$ or 0, respectively. A *pda* is called *real-time* (*rpda*) if $pX \overset{a}{\longmapsto} q\alpha$ implies $a \neq \varepsilon$. A *pda* is called *deterministic* (*dpda*) if for every $p \in Q$, $X \in \Gamma \cup \{\bot\}$ and $a \in \Sigma \cup \{\varepsilon\}$ we have (i) $|\{q\alpha \mid pX \overset{a}{\longmapsto} q\alpha\}| \le 1$ and (ii) if $pX \overset{\varepsilon}{\longmapsto} q\alpha$ and $pX \overset{a}{\longmapsto} q'\alpha'$ then $a = \varepsilon$. A real-time deterministic pushdown automaton is denoted by *rdpda*.

The set $Q\Gamma^*\bot$ is the set of *configurations* of a *pda*. The configuration $q_0\bot$ is called *initial*. The transition relation between configurations is defined by: if

$pX \overset{a}{\longmapsto} q\alpha$, then $pX\beta \xrightarrow{a} q\alpha\beta$ for every $\beta \in \Gamma^*$. A transition $p\alpha \xrightarrow{\varepsilon} q\beta$ is called *ε-transition* or *ε-move*. The *labelled transition system* generated by A is the edge-labeled, directed graph $(Q\Gamma^*\bot, \bigcup_{a\in\Sigma\cup\{\epsilon\}} \xrightarrow{a})$. Wherever convenient we use common graph theoretic terminology, like *(w-labeled) path* or *reachability*. Given $w \in \Sigma^*$, we write $p\alpha \overset{w}{\underset{A}{\Longrightarrow}} q\beta$ (or just $p\alpha \overset{w}{\Longrightarrow} q\beta$ if A is understood) if there exists a finite w'-labeled path from $p\alpha$ to $q\beta$ in A such that $w' \in (\Sigma\cup\{\varepsilon\})^*$ and w is the projection of w' onto Σ. We say that A is *complete* if $q_0\bot \overset{w}{\underset{A}{\Longrightarrow}} q\alpha$ for every $w \in \Sigma^*$. We say that A *recognizes* the language $\mathcal{L}(A) = \{w \in \Sigma^* \mid q_0\bot \overset{w}{\Longrightarrow} p\alpha, p \in F\}$. A language recognized by a *pda* (*dpda*, *rpda*, *rdpda*) is called (deterministic, real-time, real-time deterministic) *context-free* and the class of all such languages is denoted by *CFL*, *dCFL*, *rCFL*, and *rdCFL*, respectively.

Pushdown automata may reject a word because they get stuck before they read it completely, or because after reading it they get engaged in an infinite sequence of ε-moves that do not visit any final state. They may also scan a word and then make several ε-moves that visit both final and non-final states in arbitrary ways. Moreover, in a rule $pX \overset{a}{\longmapsto} q\alpha$ the word α can be arbitrary. For our purposes it is convenient to eliminate these "anomalies" by introducing a normal form.

Definition 1. *A pushdown automaton* $A = (Q, \Sigma, \Gamma, \delta, q_0, F)$ *is* normalized *if*

(i) *A is complete;*
(ii) *for all* $p \in Q$*, all rules in* δ *of the form* $pX \overset{a}{\longmapsto} q\alpha$ *either satisfy* $a \in \Sigma$ *or all of them satisfy* $a = \epsilon$*, but not both;*
(iii) *every rule in* δ *is of the form* $pX \overset{a}{\longmapsto} q\lambda$*,* $pX \overset{a}{\longmapsto} qX$*, or* $pX \overset{a}{\longmapsto} qYX$ *where* $a \in \Sigma \cup \{\varepsilon\}$*.*

States which admit only ε-transitions (see property (ii)), are called *ε-states*.

Lemma 1. *For every pda (dpda, rpda, rdpda) there is a normalized pda (dpda, rpda, rdpda, respectively), that recognizes the same language.*

3 Height Determinism

Loosely speaking, a *pda* is height-deterministic if the stack height is determined solely by the input word; more precisely, a *pda* A is height-deterministic if all runs of A on input $w \in (\Sigma \cup \{\varepsilon\})^*$ (here, crucially, ε is considered to be a part of the input) lead to configurations of the same stack height. Given two height-deterministic *pda* A and B, we call them synchronized if their stack heights coincide after reading the same input words (again, this includes reading the same number of ε's between two letters). The idea of height-determinism will be discussed more formally below.

Definition 2. *Let* A *be a pda over the alphabet* Σ *with the initial state* q_0*, and let* $w \in (\Sigma\cup\{\varepsilon\})^*$*. The set* $N(A, w)$ *of* stack heights reached by A after reading w *is* $\{|\alpha| \mid q_0\bot \xrightarrow[A]{w} q\alpha\bot\}$*. A* height-deterministic *pda (hpda) A is a pda that is*

(i) normalized, and
(ii) $|N(A, w)| \leq 1$ *for every* $w \in (\Sigma \cup \{\varepsilon\})^*$.

A language recognized by some hpda is height-deterministic context-free. *The class of height-deterministic context-free languages is denoted by hCFL.*

Note that every normalized *dpda* is trivially an *hpda*.

Definition 3. *Two hpda* A *and* B *over the same alphabet* Σ *are* synchronized, *denoted by* $A \sim B$, *if* $N(A, w) = N(B, w)$ *for every* $w \in (\Sigma \cup \{\varepsilon\})^*$.

Intuitively, two *hpda* are synchronized if their stacks increase and decrease in lockstep at every run on the same input. Note that $\sim$ is an equivalence relation over all *hpda*. Let $[A]_\sim$ denote the equivalence class containing the *hpda* A, and let A-*hCFL* denote the class of languages $\{\mathcal{L}(A) \mid A \in [A]_\sim\}$ recognized by any *hpda* synchronized with A.

In the following subsections we will study some properties of general, real-time, and deterministic *hpda*.

3.1 The General Case

Let us first argue that height-determinism does not restrict the power of *pda*.

Theorem 1. *hCFL* = *CFL*.

The basic proof idea is that for any context-free language L a *pda* A can be constructed such that $\mathcal{L}(A) = L$ and for every non-deterministic choice of A a different number of ε-moves is done.

Proof. Let $L \in$ *CFL*. There exists an *rpda* $A = (Q, \Sigma, \Gamma, \delta, q_0, F)$ with $\mathcal{L}(A) = L$. We can assume that A is normalized by Lemma 1. Certainly, $|N(A, w)| \leq 1$ for every $w \in \Sigma^*$ does not hold in general. However, we can construct a *pda* $A' = (Q', \Sigma, \Gamma, \delta', q_0, F)$ from A such that a different number of ε-moves is done for every non-deterministic choice of A after reading a letter. In this way every run of A' on some input w is uniquely determined by the number of ε-moves between reading letters from the input. Hence, $|N(A, w)| \leq 1$ for every $w \in (\Sigma \cup \{\varepsilon\})^*$ (condition (ii) of the Definition 2) is satisfied.

Formally, over all $p \in Q$ and $X \in \Gamma \cup \{\bot\}$ and $a \in \Sigma$, let m be the maximum number of rules of the form $pX \overset{a}{\longmapsto} q\alpha$ for some q and α. For every $q\alpha$ appearing on the right-hand side of some rule, we introduce m new states $p^1_{q\alpha}, p^2_{q\alpha}, \ldots, p^m_{q\alpha}$ and for every $X \in \Gamma \cup \{\bot\}$ and $1 \leq i < m$ we add the rules

$$p^i_{q\alpha} X \overset{\varepsilon}{\longmapsto} p^{i+1}_{q\alpha} X \qquad \text{and} \qquad p^m_{q\alpha} X \overset{\varepsilon}{\longmapsto} q\alpha\ .$$

Now, for all $p \in Q$, $X \in \Gamma \cup \{\bot\}$ and $a \in \Sigma$, let

$$pX \overset{a}{\longmapsto} q_1\alpha_1,\ pX \overset{a}{\longmapsto} q_2\alpha_2,\ \ldots, pX \overset{a}{\longmapsto} q_n\alpha_n$$

be all rules under the action a with the left-hand side pX; we replace all these rules with the following ones:

$$pX \overset{a}{\longmapsto} p^1_{q_1\alpha_1} X,\ pX \overset{a}{\longmapsto} p^2_{q_2\alpha_2} X,\ \ldots, pX \overset{a}{\longmapsto} p^n_{q_n\alpha_n} X\ .$$

Note that A' is normalized if A is normalized, and that $\mathcal{L}(A') = \mathcal{L}(A)$. $\square$

Theorem 2. *Let A be any hpda. Then $REG \subseteq A$-hCFL.*

In particular, if R is a complete dfsa then there exists an hpda $B \in A$-hCFL such that $\mathcal{L}(B) = \mathcal{L}(R)$ and $|B| = \mathcal{O}(|A|\,|R|)$. Moreover, if A is deterministic, then B is deterministic.

Proof. Let $L \in REG$, and let R be a *dfsa* recognizing L. W.l.o.g. we can assume that R is complete, that is, for every $a \in \Sigma$ and state r in R there is a transition $r \xrightarrow[R]{a} r'$. We construct a *pda* B as the usual product of (the control part of) A with R: for all $a \in \Sigma$, B has a rule $(q, r)X \underset{B}{\overset{a}{\longmapsto}} (q', r')\alpha$ if and only if $qX \underset{A}{\overset{a}{\longmapsto}} q'\alpha$ and $r \xrightarrow[R]{a} r'$; for every state r of R, B has an ε-rule $(q, r)X \underset{B}{\overset{\varepsilon}{\longmapsto}} (q', r)\alpha$ if and only if $qX \underset{A}{\overset{\varepsilon}{\longmapsto}} q'\alpha$. The final states of B are the pairs (q, r) such that r is a final state of R. Clearly, we have $|B| = \mathcal{O}(|A|\,|R|)$. Moreover, every run of B on some $w \in \Sigma^*$ ends in a final state (q, r) if and only if R is in r after reading w, and hence, $\mathcal{L}(B) = L$.

Next we show that B is *hpda*. Firstly, condition (ii) of Definition 2 and completeness (Definition 1(i)) clearly hold. Secondly, every state of B either admits only ε-transitions or non-ε-transitions but not both (Definition 1(ii)) since $(p, r)X \underset{B}{\overset{\varepsilon}{\longmapsto}} (q, r)\alpha$ and $(p, r)Y \underset{B}{\overset{a}{\longmapsto}} (q', r')\beta$ implies $pX \underset{A}{\overset{\varepsilon}{\longmapsto}} q\alpha$ and $pY \underset{A}{\overset{a}{\longmapsto}} q'\beta$, contradicting the normalization of A. Finally, Definition 1(iii) follows trivially from the fact that A is normalized. It remains to prove $A \sim B$, however, this follows easily because the height of B's stack is completely determined by A. □

Note that the *pda* B in Theorem 2 is real-time (deterministic) if A is real-time (deterministic). The following closure properties are easily proved using classical constructions.

Theorem 3. *Let A be any hpda. Then A-hCFL is closed under union and intersection.*

In particular, let A and B be two hpda with $A \sim B$.

(i) The language $\mathcal{L}(A) \cup \mathcal{L}(B)$ is recognized by some hpda C of size $\mathcal{O}(|A| + |B|)$ such that $A \sim C \sim B$.
(ii) If A and B are deterministic, then the language $\mathcal{L}(A) \cup \mathcal{L}(B)$ is recognized by some deterministic hpda C of size $\mathcal{O}(|A|\,|B|)$ such that $A \sim C \sim B$.
(iii) The language $\mathcal{L}(A) \cap \mathcal{L}(B)$ is recognized by some hpda C of size $\mathcal{O}(|A|\,|B|)$ such that $A \sim C \sim B$. If A and B are deterministic, then C is deterministic.

Moreover, we have in all cases that if both A and B are rpda, then C is an rpda.

3.2 The Real-Time Case

Let *rhpda* denote a real-time *hpda*, and let *rhCFL* denote the class of languages generated by *rhpda*. We remark that *rhpda* contain visibly pushdown automata introduced in [3] but not vice versa as shown in Example 1 below. A visibly pushdown automaton A (*vpda*) over Σ is an *rpda* together with a fixed partition of $\Sigma = \Sigma_c \cup \Sigma_i \cup \Sigma_r$ such that if $pX \underset{A}{\overset{a}{\longmapsto}} qYX$ then $a \in \Sigma_c$ and if $pX \underset{A}{\overset{a}{\longmapsto}} qX$ then $a \in \Sigma_i$ and if $pX \underset{A}{\overset{a}{\longmapsto}} q\lambda$ then $a \in \Sigma_r$. By *vCFL* we denote the class of languages generated by *vpda*.

Example 1. Consider the language $L_1 = \{a^n b a^n \mid n \geq 0\}$ which is not recognized by any *vpda*; see also [3]. Indeed, a *vpda* recognizing L_1 would have to either only push or only pop or only change its state whenever the letter a is read, but then the two powers of a in an input word from a^*ba^* could not be compared for most inputs. However, the obvious *rdpda* that pushes the first block of a's into the stack, reads the b, reads the second block of a's while popping the first block from the stack, and compares whether they have the same length, is a *rhpda* that accepts L_1. □

On the other hand, it is easy to see that not every language accepted by an *rpda* can also be accepted by a *rhpda*. For example, the language of all palindromes over Σ is in *rCFL* but not in *rhCFL*. This follows from the fact that this language does not belong to *rdCFL*, and from the fact that *rdCFL* = *rhCFL*, which is proved below in Theorem 4. All together, we get the following hierarchy.

$$REG \subsetneq vCFL \subsetneq rhCFL = rdCFL \subsetneq rCFL = hCFL = CFL$$

The next theorem shows that real-time *hpda* can be determinised. The proof of this theorem uses the same basic technique as for determinising *vpda* [3].

Theorem 4. *rhCFL = rdCFL.*

In particular, we can construct for every rhpda A a deterministic rhpda B such that $\mathcal{L}(A) = \mathcal{L}(B)$ *and* $A \sim B$ *and B has* $\mathcal{O}\left(2^{n^2}\right)$ *many states and a stack alphabet of size* $\mathcal{O}\left(|\Sigma| 2^{n^2}\right)$ *where n is the number of pairs of states and stack symbols of A.*

It follows from Theorem 4 and the closure of *rdCFL* under complement that a complement A^c exists for every *rhpda* A. However, the following corollary more precisely shows that A^c can be chosen to satisfy $A^c \sim A$.

Corollary 1. *rhCFL is closed under complement.*

In particular, for every rhpda A there exists an rhpda B such that $\mathcal{L}(B) = \mathcal{L}(A)^c$ *and* $A \sim B$ *and* $|B| = 2^{\mathcal{O}(|A|^2)}$.

The emptiness problem can be decided in time $\mathcal{O}(n^3)$ for any *pda* of size n; see for example [6]. In combination with the previous results we get the bound on the equivalence problem.

Theorem 5. *Language equivalence of synchronized rhpda is decidable.*

In particular, let A and B be two rhpda with $A \sim B$, *and let* $n = |A|$ *and* $m = |B|$. *We can decide* $\mathcal{L}(A) \stackrel{?}{=} \mathcal{L}(B)$, *in time* $2^{\mathcal{O}(n^2+m^2)}$.

3.3 The Deterministic Case

Contrary to the real-time case, arbitrary *hpda* cannot always be determinised, as shown by Theorem 1. For this reason we investigate the synchronization relation $\sim$ restricted to the class of deterministic pushdown automata. Certainly, *dhCFL* = *dCFL* since every *dpda* can be normalized by Lemma 1 and then it is trivially height-deterministic. However, we lay the focus in this section on the closure of each equivalence class of $\sim$ under complement. Therefore, we denote a deterministic *hpda* by *dhpda*. The class of languages recognized by some *dhpda* synchronized with the *dhpda* A is denoted by A-*dhCFL*.

First, we show that, as in the real-time case, every *dhpda* can be complemented without leaving its equivalence class. The proof is, however, more delicate due to the presence of ε-rules. In fact, the normalization of Definition 1 has been carefully chosen to make this theorem possible.

Theorem 6. *Let A be any dhpda. Then A-dhCFL is closed under complement.*

In particular, for every dhpda B there exists a complement dhpda B^c such that $B^c \sim B$ and $|B^c| = \mathcal{O}(|B|)$.

Proof. Let $B = (Q, \Sigma, \Gamma, \delta, q_0, F)$. Let $Q' \subseteq Q$ be the set of all ε-states of B and let $Q'' = Q \setminus Q'$. We construct B^c by first defining an *dhpda* B' equivalent to B such that a word is accepted if and only if it can be accepted with a state in Q'', that is, a state which allows only non-ε-moves. Then the set of accepting states is a subset of states in Q'' that do not accept $\mathcal{L}(B)$. This gives the complement of B.

We will define a *dhpda* B' such that $B \sim B'$ and $\mathcal{L}(B') = \mathcal{L}(B)$ and every accepting path in the transition system generated by B' ends in a state in $Q' \cup (Q'' \cap F)$, that is, when B' accepts a word w, then B' shall end in a final state after reading w with a maximal (and finite by property (i) in Definition 1) number of ε moves after reading the last letter of w. Note that the completeness property of B in Definition 1 implies that B is always in a state in Q'' after reading w followed by a maximal number of ε-transitions.

Let $B' = (Q \times \{0, 1\}, \Sigma, \Gamma, \vartheta, q_0', F')$ with $F' = Q \times \{1\}$, and $q_0' = (q_0, 1)$ if $q_0 \in F$ and $q_0' = (q_0, 0)$ otherwise. The set of rules ϑ is defined as follows:

- $((p, i), X, e, (q, 1), \alpha) \in \vartheta$ if $(p, X, e, q, \alpha) \in \delta$ and $q \in F$,
- $((p, i), X, a, (q, 0), \alpha) \in \vartheta$ if $(p, X, a, q, \alpha) \in \delta$ and $q \notin F$, and
- $((p, i), X, \varepsilon, (q, i), \alpha) \in \vartheta$ if $(p, X, \varepsilon, q, \alpha) \in \delta$ and $q \notin F$.

where $e \in \Sigma \cup \{\varepsilon\}$ and $i \in \{0, 1\}$ and $a \in \Sigma$. We have now $\mathcal{L}(B') = \mathcal{L}(B)$. Indeed, we have two copies, indexed with 0 and 1, of B in B' and whenever an accepting state is reached in B then it is reached in the 1-copy of B in B' (the first two items in the definition of ϑ above) and B' is in an accepting state and both B and B' accept the word read so far. The set of accepting states of B' is only left when the next letter is read from the input and B reaches a non-accepting state (the third item in the definition of ϑ above). Otherwise, B' remains in the respective copy of B (first and fourth item in the definition of ϑ above). Clearly, $B' \sim B$.

Now, $B^c = (Q \times \{0, 1\}, \Sigma, \Gamma, \vartheta, q_0', Q'' \times \{0\})$. ☐

The equivalence checking problem for two synchronized *dhpda* is, like in the real-time case, decidable.

Theorem 7. *Language equivalence of synchronized dhpda is decidable.*

In particular, for any dhpda A and B such that $A \sim B$, we can decide whether $\mathcal{L}(A) \stackrel{?}{=} \mathcal{L}(B)$ in time $\mathcal{O}\left(|A|^3\,|B|^3\right)$.

4 Other Language Classes — A Comparison

In this section height-deterministic context-free languages are compared to two other recent approaches of defining classes of context-free languages closed under boolean operations. In [5], Caucal introduced an extension of Alur and Madhusudan's visibly pushdown languages [3], and proved that it forms a boolean algebra. The second class is the one introduced by Fisman and Pnueli in [7]. We show in this section that *rhCFL* (which is a proper subclass of *dhCFL*) properly contains these two classes.

4.1 Caucal's Class

Caucal's class is defined with the help of a notion of synchronization, just as our *hCFL* class.[1] Before we can define Caucal's synchronization, we need some preliminaries.

A *fst* is *input deterministic*, if $(s, a, m, t) \in \varrho$ and $(s, a, n, t') \in \varrho$ implies that $m = n$ and $t = t'$. Caucal considers input deterministic transducers from Σ^* to $\mathbb{Z}$ (the additive monoid of integers) where every state accepts, i.e., transducers whose transitions are labeled with a letter from Σ and an integer. When the transducer reads a word over Σ, it outputs the sum of the integers of the transitions visited. Notice that if a transducer T is input deterministic then the set $T(w)$ is a singleton, i.e., a set containing one single integer. By abuse of notation, we identify $T(w)$ with this integer. We let $|T(w)|$ denote the absolute value of $T(w)$.

Given an input deterministic *fst* T from Σ^* to $\mathbb{Z}$ and an *rpda* A over Σ with initial state q_0, we say that A is a *T-synchronized pda* (*T-spda*) if $q_0\bot \xrightarrow[A]{w} p\alpha\bot$ implies $|\alpha| = |T(w)|$ for every $w \in \Sigma^*$ and every configuration $p\alpha$ of A. Let *wSCFL* denote the class of all languages that are recognized by some *T-spda* for some T. (See also Caucal's introduction of *wSCFL* in [5]).

Theorem 8. *wSCFL* $\subsetneq$ *rhCFL.*

In particular, the language

$$L_3 = \{a^m b^n w \mid m > n > 0, |w|_a = |w|_b, w_{(1)} = a \text{ if } w \neq \lambda\}$$

belongs to rhCFL but not to wSCFL.

4.2 Fisman and Pnueli's Class

We define the class of M-synchronized *pda*, which is the formalism used by Fisman and Pnueli in their approach to non-regular model-checking [7].

Let $M = (\Delta, \Gamma, \delta)$ be a *1-rdpda*, let $R = (Q, \Sigma \times \Gamma, q_0, \varrho, F)$ be a *dfsa*, and let $\phi\colon \Sigma \to \Delta$ be a substitution. The *cascade product* $M \circ_\phi R$ is the *rdpda* $(Q, \Sigma, \Gamma, \delta', q_0, F)$ with $qX \overset{a}{\longmapsto} \varrho(q, (a, X))\delta(\phi(a), X)$ for all $q \in Q$, $a \in \Sigma$ and $X \in \Gamma \cup \{\bot\}$. An *rdpda* A is called *M-synchronized* (*M-spda*) if there exists

[1] In fact, Caucal's class was the starting point of our study.

a substitution ϕ and a *dfsa* R such that $A = M \circ_\phi R$. Let *1SCFL* denote the class of all languages that are recognized by some *M-spda* for some *1-rdpda M*. See also Fisman and Pnueli's introduction of *1SCFL* in [7].

Theorem 9. *1SCFL* $\subsetneq$ *rhCFL.*
In particular, the language

$$L_4 = \{a^n b a^n \mid n \geq 0\} \cup \{a^n c a^{2n} \mid n \geq 0\}$$

belongs to rhCFL but not to 1SCFL.

5 Conclusion

We have introduced several (sub)classes of the class of context-free languages that are closed under boolean operations. Our key technical tools are height-deterministic pushdown automata (*hpda*) and synchronization between *hpda*. These notions are inspired by and generalize Caucal's work on real-time synchronized pushdown graphs [5]. In fact, our results can be seen as an extension of Caucal's ideas to pushdown automata with ϵ-transitions. This extension has turned out to be rather delicate. Both Theorem 2 (*REG* $\subseteq$ *A-hCFL*) and Theorem 6 (*A-hCFL* is closed under complement) depend crucially on the normalization of Definition 1 which had to be carefully chosen. In a sense, one of the contributions of the paper is to have worked out the right notion of normalization. We have also showed that language equivalence of real-time height-deterministic pushdown automata is decidable in EXPTIME.

Both this paper and Caucal's have been also inspired by Alur and Madhusudan's work on visibly pushdown automata, initiated in [3]. From an automata-theoretic point of view, we have extended the theorem of [3], stating that visibly pushdown automata are closed under boolean operations, to deterministic *hpda*. This is rather satisfactory, because deterministic *hpda* recognize all deterministic context-free languages, while visibly *pda* are far from it. Remarkably, the extension is be achieved at a very low cost; in our opinion, height-deterministic *pda* are, at least from the semantical point of view, as natural and intuitive as visibly *pda*.

Acknowledgments. The authors are deeply indebted to Javier Esparza who contributed to this work in many ways. We also thank to the anonymous referees for their useful remarks.

References

1. Alur, R., Etessami, K., Madhusudan, P.: A temporal logic of nested calls and returns. In: Jensen, K., Podelski, A. (eds.) TACAS 2004. LNCS, vol. 2988, pp. 467–481. Springer, Heidelberg (2004)

2. Alur, R., Kumar, V., Madhusudan, P., Viswanathan, M.: Congruences for visibly pushdown languages. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1102–1114. Springer, Heidelberg (2005)
3. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: ACM Symposium on Theory of Computing (STOC'04), pp. 202–211. ACM Press, New York (2004)
4. Bárány, V., Löding, C., Serre, O.: Regularity problems for visibly pushdown languages. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 420–431. Springer, Heidelberg (2006)
5. Caucal, D.: Synchronization of pushdown automata. In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 120–132. Springer, Heidelberg (2006)
6. Esparza, J., Hansel, D., Rossmanith, P., Schwoon, S.: Efficient algorithms for model checking pushdown systems. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 232–247. Springer, Heidelberg (2000)
7. Fisman, D., Pnueli, A.: Beyond regular model checking. In: Hariharan, R., Mukund, M., Vinay, V. (eds.) FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science. LNCS, vol. 2245, pp. 156–170. Springer, Heidelberg (2001)
8. Löding, Ch., Madhusudan, P., Serre, O.: Visibly pushdown games. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS 2004. LNCS, vol. 3328, pp. 408–420. Springer, Heidelberg (2004)
9. Murawski, A., Walukiewicz, I.: Third-order idealized algol with iteration is decidable. In: Sassone, V. (ed.) FOSSACS 2005. LNCS, vol. 3441, pp. 202–218. Springer, Heidelberg (2005)
10. Pitcher, C.: Visibly pushdown expression effects for XML stream processing. In: Proceedings of Programming Language Technologies for XML (PLAN-X), pp. 5–19 (2005)
11. Srba, J.: Visibly pushdown automata: From language equivalence to simulation and bisimulation. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 89–103. Springer, Heidelberg (2006)