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 [9, 2, 4]. The theory found a number of interesting applications, e.g. in program analysis [1, 10] and XML processing [11]. 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, 12]. 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.
** **B**asic **R**esearch **I**n **C**omputer **S**cience, Danish National Research Foundation.

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 $1 \leq i \leq n$. 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) $|\{q\alpha \mid pX \overset{a}{\longmapsto} q\alpha\}| \leq 1$ 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 $\{|\alpha| \mid q_0\bot \overset{w}{\underset{A}{\longrightarrow}} q\alpha\bot\}$. 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 $(q,r)X \overset{a}{\underset{B}{\longmapsto}} (q',r')\alpha$ if and only if $qX \overset{a}{\underset{A}{\longmapsto}} q'\alpha$ and $r \xrightarrow[R]{a} r'$; for every state r of R, B has an ε-rule $(q,r)X \overset{\varepsilon}{\underset{B}{\longmapsto}} (q',r)\alpha$ if and only if $qX \overset{\varepsilon}{\underset{A}{\longmapsto}} q'\alpha$. 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 $(p,r)X \overset{\varepsilon}{\underset{B}{\longmapsto}} (q,r)\alpha$ and $(p,r)Y \overset{a}{\underset{B}{\longmapsto}} (q',r')\beta$ implies $pX \overset{\varepsilon}{\underset{A}{\longmapsto}} q\alpha$ and $pY \overset{a}{\underset{A}{\longmapsto}} q'\beta$, 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. $\square$

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 $pX \xmapsto[A]{a} qYX$ then $a \in \Sigma_c$ and if $pX \xmapsto[A]{a} qX$ then $a \in \Sigma_i$ and if $pX \xmapsto[A]{a} q\lambda$ 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 $a^* b a^*$ 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. $\square$

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, $B^c = (Q \times \{0, 1\}, \Sigma, \Gamma, \vartheta, q'_0, Q'' \times \{0\})$. $\square$

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.[3] 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

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

belongs to rhCFL but not to wSCFL.

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

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 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. R. Alur, K. Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In *Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, volume 2988 of *LNCS*, pages 467–481. Springer-Verlag, 2004.
2. R. Alur, V. Kumar, P. Madhusudan, and M. Viswanathan. Congruences for visibly pushdown languages. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *LNCS*, pages 1102–1114. Springer-Verlag, 2005.
3. R. Alur and P. Madhusudan. Visibly pushdown languages. In *ACM Symposium on Theory of Computing (STOC'04)*, pages 202–211. ACM Press, 2004.
4. V. Bárány, Ch. Löding, and O. Serre. Regularity problems for visibly pushdown languages. In *Proceedings of the 23rd Annual Symposioum on Theoretical Aspects of Computer Science (STACS'06)*, volume 3884 of *LNCS*, pages 420–431. Springer-Verlag, 2006.
5. D. Caucal. Synchronization of pushdown automata. In *Developments in Laguage Theory (DLT'06)*, volume 4036 of *LNCS*, pages 120–132, Berlin, 2006. Springer-Verlag.
6. J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *Computer Aided Verification (CAV'00)*, number 1855 in LNCS, pages 232–247, Berlin, 2000. Springer-Verlag.
7. D. Fisman and A. Pnueli. Beyond regular model checking. In *Foundations of Software Technology and Theoretical Computer Science (FST&TCS'01)*, volume 2245 of *LNCS*, pages 156–170, Berlin, 2001. Springer-Verlag.
8. J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
9. Ch. Löding, P. Madhusudan, and O. Serre. Visibly pushdown games. In *Proceedings of the 24th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'04)*, volume 3328 of *LNCS*, pages 408–420. Springer-Verlag, 2004.
10. A. Murawski and I. Walukiewicz. Third-order idealized algol with iteration is decidable. In *Proceedings of the 8th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'05)*, volume 3441 of *LNCS*, pages 202–218, 2005.
11. C. Pitcher. Visibly pushdown expression effects for XML stream processing. In *Proceedings of Programming Language Technologies for XML (PLAN-X)*, pages 5–19, 2005.
12. J. Srba. Visibly pushdown automata: From language equivalence to simulation and bisimulation. In *Proceedings of the 15th Annual Conference of the European Association for Computer Science Logic (CSL'06)*, volume 4207 of *LNCS*, pages 89–103. Springer-Verlag, 2006.

A Proofs from Sections 2 and 3

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

Proof. We proceed as follows. Given a *pda* A, we construct *pda* A_1, A_2, A_3 recognizing the same language as A such that A_1 satisfies (iii), A_2 satisfies (iii) and (ii), and A_3 satisfies (iii), (ii), and (i), respectively. It will follow immediately from the constructions that if A is real-time or deterministic then so is A_3. In what follows, the components of the automata A_1, A_2 and A_3 will be indexed accordingly, i.e., for example the set of control states of A_1 will be denoted by Q_1, the set rules of A_2 will be denoted by δ_2 etc.

For A_1, we first use the standard construction that yields an automaton A' equivalent to A but in which every rule $pX \overset{a}{\longmapsto} q\alpha$ satisfies $|\alpha| \leq 2$. Then, we construct A_1 as described in Lemma 10.2 of [8]; A_1 has $Q_1 = Q' \times \Gamma$ as the set of states and is defined to satisfy the property that $pX\alpha \overset{w}{\underset{A}{\Longrightarrow}} qY\beta$ holds iff $[p, X]\alpha \overset{w}{\underset{A}{\Longrightarrow}} [q, Y]\beta$ does. It also follows from the lemma that if A is deterministic, then so is A_1 (actually, the lemma is stated for *dpda*, but the construction works in general).

The *pda* A_2 is the final result of the following procedure: For every pair of rules of A_1 of the form $pX \overset{a}{\longmapsto} q\alpha$ and $pY \overset{\varepsilon}{\longmapsto} r\beta$ such that $a \in \Sigma$, add a state p_X not in A_2 and new rules $pX \overset{\varepsilon}{\longmapsto} p_X X$ and $p_X X \overset{a}{\longmapsto} q\alpha$ to A_1, and remove the rule $pX \overset{a}{\longmapsto} q\alpha$ from δ_1. Clearly, A_2 satisfies (ii) and, since A_1 satisfies (iii) so does A_2. Observe also that if A_1 is deterministic, then $X \neq Y$ and, for example, $pX \overset{a}{\longmapsto} q\alpha$ and $pX \overset{b}{\longmapsto} q'\alpha'$ is replaced by $pX \overset{\varepsilon}{\longmapsto} p_X X$ and $p_X X \overset{a}{\longmapsto} q\alpha$ and $p_X X \overset{b}{\longmapsto} q'\alpha'$, and so A_2 is also deterministic.

The *pda* A_3 is obtained by adding to A_2 a dead state d, the rules $dX \overset{a}{\longmapsto} dX$ for every $X \in \Gamma \cup \{\perp\}$ and $a \in \Sigma$, a rule $pX \overset{a}{\longmapsto} dX$ for every non-ε-state $p \in Q$ and every $X \in \Gamma \cup \{\perp\}$ and $a \in \Sigma$ whenever $pX \overset{a}{\longmapsto} q\alpha$ is not already in δ_2 for some q and α, and, if A is non-deterministic, a rule $pX \overset{\varepsilon}{\longmapsto} dX$ for every ε-state $p \in Q$. Obviously, this construction does not violate the properties (ii) and (iii) nor does it change the accepted language for $\delta_2 \subseteq \delta_3$ and every added rule leads to the dead state d. Moreover, we have $q_0\perp \overset{w}{\Longrightarrow} q\alpha$ for every $w \in \Sigma^*$, and hence, satisfying property (i), if A_2 is not deterministic and the labeled transition system generated by A_2 does not contain an infinite ε-path reachable from $q_0\perp$. This is the only case left. Next, we will show that A_2 can be suitably modified in order to avoid infinite ε-paths.

Assume that A_2 is deterministic. We show how a *dpda* A_2' with $\mathcal{L}(A_2') = \mathcal{L}(A_2)$ can be constructed from A_2 such that infinite ε-paths are avoided. We proceed as in the proof of Lemma 10.3 of [8]. We add to A_2 a non-final state d' and a final state f, rules $d'X \overset{a}{\longmapsto} d'X$ for every $X \in \Gamma \cup \{\perp\}$ and $a \in \Sigma$, rules $fX \overset{\epsilon}{\longmapsto} d'X$ for every $X \in \Gamma \cup \{\perp\}$, and a further set of rules according to the following criterion. For every rule $pX \overset{\epsilon}{\longmapsto} q\alpha$ of A_2 such that the labelled transition system generated by A_2 contains an infinite ε-labelled path whose first edge goes from pX to $q\alpha$, add a new rule $pX \overset{\epsilon}{\longmapsto} fX$ to A_2' if the path visits some final state, and a rule $pX \overset{\epsilon}{\longmapsto} d'X$ otherwise (and remove the corresponding old one). Clearly, A'_2 inherits properties (ii) and (iii) from A_2, and it is deterministic. It is also easy to see that A'_2 and A_2 recognize the same language. It remains to show that the labelled transition system of A'_2 contains no ε-labelled infinite paths. Assume that it does. By construction the path does not visit the states d' or f, and so it is also a path of the transition system of A_2. Let $pX\alpha$ be any configuration of the path such that α has minimal length. Then, by the definition of the semantics of a pushdown automaton, the infinite sequence of rules applied from $pX\alpha$ can also be applied from pX, and so the transition systems of A_2 and A'_2 also have an infinite ε-labelled path starting at pX. It follows that A'_2 contains a rule of the form $pX \overset{\epsilon}{\longmapsto} q\alpha$, contradicting the definition of A'_2. $\square$

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.

Proof. (i) Let $A = (Q, \Sigma, \Gamma, \delta, q_0, F)$ and $B = (S, \Sigma, \Delta, \vartheta, s_0, G)$ such that $A \sim B$. Consider the automaton $C = (Q \uplus S \uplus \{p\}, \Sigma, \Gamma \uplus \Delta, \rho, p, F \uplus G)$ where $\uplus$ denotes the disjoint union of sets, and $\rho = (\delta \uplus \vartheta) \cup \{(p, \bot, a, p', \alpha) \mid (q_0, \bot, a, p', \alpha) \in \delta \text{ or } (s_0, \bot, a, p', \alpha) \in \vartheta\}$. We have that $\mathcal{L}(C) = \mathcal{L}(A) \cup \mathcal{L}(B)$ and C has size $\mathcal{O}(|A| + |B|)$ and $A \sim C \sim B$.

(ii) and (iii) Consider the automaton $C = (Q \times S, \Sigma, \Gamma \times \Delta, \rho, (q_0, s_0), H)$ where we define $\rho = \{((q, s), (X, Y), a, (r, t), (\alpha, \beta)) \mid (q, X, a, r, \alpha) \in \delta \text{ and } (s, Y, a, t, \beta) \in \vartheta\}$. Since $A \sim B$, the stacks of A and B run in lockstep on any input word. Moreover, C is normalized since both A and B are normalized (see also the proof of Theorem 2). We have that $\mathcal{L}(C) = \mathcal{L}(A) \cap \mathcal{L}(B)$ if $H = \{(q, s) \mid q \in F \text{ and } s \in G\}$, and $\mathcal{L}(C) = \mathcal{L}(B) \cup \mathcal{L}(C)$ if $H = \{(q, s) \mid q \in F \text{ or } s \in G\}$. The size of C is $\mathcal{O}(|A|\,|B|)$. Moreover, $A \sim C \sim B$ and C is deterministic if both A and B are deterministic. $\square$

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.

Proof. Clearly, *rhCFL* $\supseteq$ *rdCFL* since by Lemma 1 every *rdpda* can be normalized and then it is trivially height-deterministic.

Let $A = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be an *rhpda*. We describe a construction of a deterministic *rhpda* B such that $\mathcal{L}(B) = \mathcal{L}(A)$. The main idea of this proof is a subset construction where B postpones the handling of a push transition of A until the corresponding pop transition occurs. All possible push transitions of A at a given configuration and input symbol are stored by B on the stack. The states and the stack symbols of B have two components $\mathcal{S}$ and $\mathcal{R}$ where in $\mathcal{S}$ (the "summary" set) all pairs of pairs of states and stack symbols of A are stored which are the beginning and end points of transition paths between the occurrence of a push transition and its corresponding pop and $\mathcal{R}$ (the "reachable" set) contains all pairs of states and stack symbols of A reachable from q_0. The sets $\mathcal{S}$ and $\mathcal{R}$ are appropriately updated at every transition of B as defined below. This procedure follows the proof idea from [3] for the closure of visibly pushdown automata under determinisation.

For example, a push transition stores the sets $\mathcal{S}$ and $\mathcal{R}$ of the current state of B (encoded into a stack symbol) onto the stack and goes to a new state containing $\mathcal{S}'$ and $\mathcal{R}'$ where the summary $\mathcal{S}'$ is initialized by the identity relation Id_P, where $P = Q \times (\Gamma \cup \{\bot\})$, and the set $\mathcal{R}'$ of the reachable pairs of states and topmost stack symbols is updated by the states and topmost stack symbols reachable from $\mathcal{R}$ in A. When the corresponding pop transition occurs, the sets $\mathcal{S}$ and $\mathcal{R}$ of the current state in C are updated to $\mathcal{S}''$ and $\mathcal{R}''$, respectively, where $\mathcal{S}''$ contains the beginning and end points of the paths that where stored at the previous push transition in $\mathcal{S}'$ extended by the actual push transitions in B, the transition paths taken between this push-pop pair (kept in $\mathcal{S}$), and the transitions that the current pop transition causes in A. Note that it is crucial that the stack height is only controlled by the input and not by the non-deterministic choices A might take, that is, both A and B run in lock-step w.r.t. their stack heights.

We now formally define an *rdpda* $B = (S, \Sigma, \Delta, \vartheta, s_0, G)$ with $\mathcal{L}(B) = \mathcal{L}(A)$. Let $P = Q \times (\Gamma \cup \{\bot\})$ and let $S = 2^{P \times P} \times 2^P$, $\Delta = 2^{P \times P} \times 2^P \times \Sigma$, $s_0 = (Id_{Q \times \{\bot\}}, \{(q_0, \bot)\})$ and $G = \{(\mathcal{S}, \mathcal{R}) \in S \mid \mathcal{R} \cap (F \times (\Gamma \cup \{\bot\})) \neq \emptyset\}$. The transition relation ϑ is given by:

Internal: $((\mathcal{S}, \mathcal{R}), X, a, (\mathcal{S}', \mathcal{R}'), X) \in \vartheta$ where
- $\mathcal{S}' = \{(x, (q, Y)) \mid \exists (r, Y) \in P : (x, (r, Y)) \in \mathcal{S}, (r, Y, a, q, Y) \in \delta\}$ and
- $\mathcal{R}' = \{(q, Y) \mid \exists (p, Y) \in \mathcal{R} : (p, Y, a, q, Y) \in \delta\}$,

Push: $((\mathcal{S}, \mathcal{R}), X, a, (Id_P, \mathcal{R}''), (\mathcal{S}, \mathcal{R}, a)X) \in \vartheta$ where
- $\mathcal{R}'' = \{(q, Z) \mid \exists (p, Y) \in \mathcal{R} : (p, Y, a, q, ZY) \in \delta\}$,

Pop: $((\mathcal{S}, \mathcal{R}), (\mathcal{S}', \mathcal{R}', a), b, (\mathcal{S}'', \mathcal{R}''), \lambda) \in \vartheta$ where
- $\mathcal{S}'' = \{(x, y) \mid \exists z : (x, z) \in \mathcal{S}', (z, y) \in \mathit{Update}\}$ and
- $\mathcal{R}'' = \{x \mid \exists y \in \mathcal{R}' : (y, x) \in \mathit{Update}\}$ with
- $\mathit{Update} = \{((p, Y), (q, Y)) \mid \exists (r, Z), (r', Z) \in P : (p, Y, a, r, ZY) \in \delta$ and $((r, Z), (r', Z)) \in \mathcal{S}$ and $(r', Z, b, q, \lambda) \in \delta\}$.

Note that B is real-time (no ε rules are introduced) and normalized (B is complete if A is complete). Moreover, B is deterministic and therefore (together with normalization) height-deterministic. We have that $A \sim B$ since internal, push, and pop transitions in A are mapped to internal, push, and pop transitions in B, respectively. $\square$

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 $|B| = 2^{\mathcal{O}(|A|^2)}$.

Proof. Assume a given *rhpda* A. By Theorem 4 there is some deterministic *rhpda* $B = (Q, \Sigma, \Gamma, \delta, q_0, F)$ that recognizes $\mathcal{L}(A)$ and $B \sim A$. Because B is complete and deterministic, the *dpda* $C = (Q, \Sigma, \Gamma, \delta, q_0, Q \setminus F)$ satisfies $\mathcal{L}(C) = \mathcal{L}(A)^c$ and $C \sim B \sim A$. □

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 $2^{\mathcal{O}(n^2+m^2)}$.

Proof. By Theorem 3 and 4 and Corollary 1 we can construct an *rhpda* C of size $\mathcal{O}\left(2^{n^2+m^2}\right)$ such that $\mathcal{L}(C) = \left(\mathcal{L}(A)\cap\mathcal{L}(B)^c\right)\cup\left(\mathcal{L}(A)^c\cap\mathcal{L}(B)\right)$ and $A \sim C \sim B$. Now $\mathcal{L}(A) = \mathcal{L}(B)$ iff $\mathcal{L}(C) = \emptyset$. Since the emptiness problem for *pda* can be solved in cubic time, the results follow. □

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 $\mathcal{L}(A) \stackrel{?}{=} \mathcal{L}(B)$ in time $\mathcal{O}\left(|A|^3\,|B|^3\right)$.

Proof. By Theorem 3 and 6 we can construct an *dhpda* C of size $\mathcal{O}\left(|A|\,|B|\right)$ such that $\mathcal{L}(C) = \left(\mathcal{L}(A) \cap \mathcal{L}(B)^c\right) \cup \left(\mathcal{L}(A)^c \cap \mathcal{L}(B)\right)$ and $A \sim C \sim B$. Now $\mathcal{L}(A) = \mathcal{L}(B)$ iff $\mathcal{L}(C) = \emptyset$. Since the emptiness problem for *pda* can be solved in cubic time, the result follows. □

B Proof of Theorem 8

Remark 1. Caucal's arguments in [5] are graph theoretical and motivate very well the use of an *fst* from Σ^* to $\mathbb{Z}$. However, considering context-free languages only, increments and decrements of single stack symbols are sufficient to recognize any language in *wSCFL*.

Indeed, consider some T-*spda* A. The increment and decrement of the stack length at any step is bounded by some finite k depending on T. Let us sketch the construction of an *fst* T' and an T'-*spda* A' such that $\mathcal{L}(A) = \mathcal{L}(A')$ and T' is from Σ^* to $\{-1, 0, 1\}$. Let $Q \times \{0, 1, \ldots, k\}$ be the set of states and Γ^{2k} be the set of stack symbols of A' where Q and Γ are the sets of states and stack symbols of A, respectively. The idea is that a stack symbol of A' stores at least the topmost k and at most the topmost $2k$ stack symbols of a configuration of A (given the stack is large enough; short stacks are handled as special cases). When $k+n$, with $0 \leq n \leq k$, stack symbols of a configuration $p\alpha$ of A are stored in the topmost stack symbol of a configuration of A', then A' is in the state (p, n).

Let us consider the following examples of rules for push and pop transitions. Let $X = (x_1, \ldots, x_{2k})$ be a stack symbol of A'. Let $px_{k+n} \stackrel{a}{\longmapsto} qy_1 \cdots y_m$ be a rule in A so that $m + n > k$. Then A' contains a rule $(p, n)X \stackrel{a}{\longmapsto}$

$(q, m + n - k - 1)YZ$ where we define $Z = (x_1, \dots, x_{k+n-1}, y_1, \dots, y_{k-n+1})$ and $Y = (x_{k+1}, \dots, x_{k+n-1},\ y_1, \dots,\ y_{k-n+1}, \dots y_m, z, z, \dots, z)$ where $z \in \Gamma$ is an arbitrary chosen stack symbol. If A pops m symbols (where now $m > n$) with a rule $py_1 \cdots y_m \xmapsto{a} q\lambda$, then A' has the rule $(p, n)X \xmapsto{a} (q, k + n - m)\lambda$ where $x_{k+n-i+1} = y_i$ for all $1 \le i \le m$. Other cases can be coded accordingly.

The transducer T' can be constructed from T in a similar fashion. Let S be the state set of T then $S \times \{0, \dots, k - 1\}$ is the state set of T'. For every rule $s \xrightarrow{a,m} t$ in T there are the rules $(s, n) \xrightarrow{a,k_n} (t, (m + n) \mod k)$ in T' for all $1 \le n \le k$ with $k_n = -1$ if $m + n < 0$, $k_n = 0$ if $0 \le m + n < k$, and $k_n = +1$ if $m + n \ge k$. Hence, we can w.l.o.g. assume that $\varrho \subseteq S \times \Sigma \times \{-1, 0, +1\} \times S$ in the *fst* T'.

In this section we consider an *rdcfl* L_3 which is not in *wSCFL*. In order to show this we argue that for any *rdpda* A in order to recognize L_3 it is necessary that A must control its stack movement depending on the stack content. Obviously, A cannot be T-synchronized since T is an *fst* reading the same input as A but has only finite memory.

The following lemma formalizes the rather obvious fact that on input a^n any *fst* from Σ^* to $\mathbb{Z}$ can only produce either unboundedly large (small) output for growing n with a lower (upper) bound k on all outputs or all outputs are bounded from above and below by k and $-k$, respectively, where k depends on T only.

Lemma 2. *Let T be a complete fst from Σ^* to $\mathbb{Z}$. Then one of the following holds for any $a \in \Sigma$.*

1. $\exists k : \forall n : -k < T(a^n)$ *and* $\forall m : \exists n : T(a^n) > m$.
2. $\exists k : \forall n : T(a^n) < k$ *and* $\forall m : \exists n : T(a^n) < -m$.
3. $\exists k : \forall n : -k < T(a^n) < k$.

Proof. Let $s_0 \xrightarrow{a,m_1} s_1 \xrightarrow{a,m_2} s_2 \cdots \xrightarrow{a,m_n} s_n$ be the run of T on a^n, and let ℓ be the number of states of T. The claim is trivial for all $n < \ell$. Assume $n \ge \ell$. There exists an $\ell' < \ell$ such that $s_{\ell'} = s_\ell$. Let $i = \sum_{j=\ell'+1}^{\ell} m_j$. Clearly, the cases 1, 2, and 3 follow from $i > 0$, $i < 0$, and $i = 0$, respectively, where $k = \ell \cdot \max\{|m_1|, |m_2|, \dots, |m_\ell|\}$ is a witness. $\square$

Only languages over $\Sigma = \{a, b\}$ are considered for the rest of this section.

Example 2. The following language is in *wSCFL*; see also [5].

$$L = \{w \mid |w|_a = |w|_b\} \tag{1}$$

Indeed, the following *rpda* A is T-synchronized, where we define T by the tuple $(\{s\}, \Sigma, \mathbb{Z}, s, \{(s, a, 1, s), (s, b, -1, s)\})$, and A recognizes L. Let A be defined as $(\{q_0, q_1, q_2\}, \Sigma, \{X, Y\}, \delta, q_0, \{q_0\})$ where

$$\begin{aligned}\delta = \{&(q_0, \bot, a, q_1, X\bot), (q_0, \bot, b, q_2, X\bot),\\ &(q_1, X, a, q_1, YX), (q_1, Y, a, q_1, YY), (q_1, X, b, q_0, \lambda), (q_1, Y, b, q_1, \lambda),\\ &(q_2, X, b, q_2, YX), (q_2, Y, b, q_2, YY), (q_2, X, a, q_0, \lambda), (q_2, Y, a, q_2, \lambda)\}\ .\end{aligned}$$

Theorem 8. *wSCFL* $\subsetneq$ *rdCFL.*
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 rdCFL but not to wSCFL.

Proof. It has already been shown in [5] that *wSCFL* $\subseteq$ *rdCFL*. Consider now the language L_3. We will show that $L_3 \notin$ *wSCFL* but $L_3 \in$ *rdCFL* next.

Let us give the proof idea first. Note that L_3 is similar to L in Example 2 which is in *wSCFL*. However, the witness automaton given in Example 2 relies on the particular feature of T-synchronized automata that the absolute value of $T(v)$ for some $v \in \Sigma^*$ is taken to determine the stack height. This means that T might for example give a decreasing value, translated into pop transitions from a stack of positive height, for an input sequence long enough so that negative numbers are reached, translated then into push transitions after the empty stack was hit. In fact, this feature must be used by any T-synchronized automaton recognizing L. However, this feature, necessary for recognizing the suffix of words in L_3 after the first a and b sequences, is not available for recognizing L_3 since recognizing $a^m b^n$ with $m > n$ enforces an arbitrarily high stack for suitable m and n. Hence, the general stack movement cannot change from shrinking to growing of arbitrary length within the same sequence of input symbols.

Let us be more precise. Assume there exists a T-*spda* $A = (Q, \Sigma, \Gamma, \delta, q_0, F)$ such that $\mathcal{L}(A) = L_3$.

For every $\ell > 0$ there exists an m such that $q_0\bot \xrightarrow[A]{a^m} q\alpha$ with $|\alpha| > \ell$. Indeed, if A's stack height is bounded by some constant after reading an arbitrary sequence of a's (that is there are only finitely many configurations of A after reading a sequence of a's) then there exist $m_1 < m_2$ such that $q_0\bot \xrightarrow[A]{a^{m_1}} q\alpha$ and $q_0\bot \xrightarrow[A]{a^{m_2}} q\alpha$. However, now both $a^{m_1}b^{m_2-1}ab \notin L_3$ and $a^{m_2}b^{m_2-1}ab \in L_3$ are either accepted or rejected by A; a contradiction. We consider for the rest of this proof words in $a^m\Sigma^*$ such that A is, after reading a^m, in a configuration from which the bottom of the stack is not reached anymore when reading of the rest of the word (this can be done because A is an *rpda*).

Next, we show that A cannot decide for all $w \in \{a^i b^j a^k b^\ell \mid i, j, k, \ell > 0\}$ if $a^m bw \in L_3$ or not. Assume on the contrary that A ends in a final state exactly if $\ell = i - j + k$.

Since A is T-synchronized, Lemma 2 implies that A can only either grow (shrink) its stack size over all bounds or change it only within a fixed bound when reading a sequence of some letter. Surely, the latter alternative leads to a contradiction since then A can be only in a finite number of different configurations after reading a sequence of a letter. Let us take the reading of b^j as example. Then there exist $j_1 \neq j_2$ such that A is in the same configuration after reading $a^m ba^i b^{j_1}$ and $a^m ba^i b^{j_2}$, respectively, and we have that both $a^m ba^i b^{j_1} a^k b^\ell$ and $a^m ba^i b^{j_2} a^k b^\ell$ are either accepted or rejected by A (for any k and ℓ); a contradiction. The rest of our proof argument distinguishes the cases 1 and 2 of Lemma 2 for every power of a's and b's in $a^i b^j a^k b^\ell$.

Assume that T follows case 1 of Lemma 2 when a^i is read. If T follows case 1 of Lemma 2 when b^j is read, then consider the input word $a^m ba^i b^j ab$ where i and j can be chosen large enough such that the equality of i and j is not testable by A since the size of i is stored on the stack but cannot be accessed (except for the κ topmost symbols for some κ fixed by T) when $b^j ab$ is read. If T follows case 2 of Lemma 2 when b^j is read, then let j' be the smallest natural such that $|\beta| > |\beta'|$ where $q_0\bot \xrightarrow{a^m b} p\beta$ and $p\beta \xrightarrow{a^i b^{j'}} p'\beta'$. Now, j can be chosen large enough such that $j - j'$ can be neither stored (in the κ topmost symbols) on the stack (for some fixed κ depending on T)nor in $Q \times \Gamma$. However, then $\ell = i - j + k$ cannot be tested without knowing j.

Assume that T follows case 2 of Lemma 2 when a^i is read. Then i can be chosen large enough so that the value of i is neither stored on the stack nor stored in $Q \times \Gamma$. Then $\ell = i - j + k$ cannot be tested without knowing i.

Hence, A cannot recognize for every $w \in \{a^i b^j a^k b^\ell \mid i, j, k, \ell > 0\}$ whether or not $a^m bw$ is in L_3, and we have $L_3 \notin$ *wSCFL*.

However, the *rdpda* $(\{q_0, \dots, q_3\}, \Sigma, \{X, Y, Z\}, \delta, q_0, \{q_1\})$ with

$$\begin{aligned}\delta = \{&(q_0, \bot, a, q_0, X\bot), (q_0, X, a, q_0, XX), (q_0, X, b, q_1, \lambda),\\ &(q_1, X, b, q_1, \lambda), (q_1, X, a, q_2, YZ), (q_1, Z, a, q_2, YZ), (q_1, Z, b, q_3, YZ)\\ &(q_2, Y, a, q_2, XY), (q_2, X, a, q_2, XX), (q_2, Y, b, q_1, \lambda), (q_2, X, b, q_2, \lambda),\\ &(q_3, Y, b, q_3, XY), (q_3, X, b, q_3, XX), (q_3, Y, a, q_1, \lambda), (q_3, X, a, q_3, \lambda)\}\end{aligned}$$

recognizes L_3, and hence, $L_3 \in$ *rdCFL*. $\square$

C Proof of Theorem 9

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

$$L_4 = \{a^n ba^n \mid n \geq 0\} \cup \{a^n ca^{2n} \mid n \geq 0\}$$

belongs to rdCFL but not to 1SCFL.

Proof. Clearly, *1SCFL* $\subseteq$ *rdCFL* follows from the definitions. Consider now the language L_4. Assume that there exists an M-*spda* A such that $\mathcal{L}(A) = L_4$. Comparing two arbitrary powers of letters in a word cannot be done with finite memory, that is, the finite control of A can only recognize L_4 if the stack content reflects the comparison of the number of a's before and after the single letter b or c. So, when the first series of a's is read the stack has to grow in order to encode n. When the second series of a's is read, the information about n has to be accessed (that is popped) in order to compare it to the number of a's after b or c. In that process the same configuration $q\alpha$ (of the *1-rdpda* M that controls the stack movement of A) is sooner or later met in both cases b and c where $q\bot \xrightarrow{a^k} qX\alpha$. From that point on M behaves exactly in the same way for at least the next $\mathcal{O}(n)$ a's that are read from the input. If $\{a^n ba^n \mid n \geq 0\}$ is recognized by A then also all the information about the first series of a's (except a fixed finite part) has been popped. So, the series of the first sequence of a's cannot be used both for comparing it with ba^n and ca^{2n}.

However, the *rdpda* $(\{q_0, p, r, s, t, q\}, \{a, b\}, \{X, Y\}, \delta, q_0, \{q\})$ with

$$\begin{aligned}\delta = \{&(q_0, \bot, a, p, \bot), (q_0, \bot, b, q, \bot), (q_0, \bot, c, q, \bot),\\ &(p, \bot, a, p, X\bot), (p, X, a, p, XX), (p, \bot, b, r, \bot), (p, X, b, r, X),\\ &(p, \bot, c, s, \bot), (p, X, c, s, X),\\ &(r, X, a, r, \lambda), (r, \bot, a, q, \bot), (s, \bot, a, t, \bot), (t, \bot, a, q, \bot),\\ &(s, X, a, s, Y), (s, Y, a, s, \lambda)\}\end{aligned}$$

recognizes L_4, and hence, $L_4 \in rdCFL$. $\square$