# Bisimulation Equivalence of First-Order Grammars is ACKERMANN-Complete

Petr Jančar
Dept of Computer Science, Faculty of Science
Palacký University in Olomouc
Czechia

Sylvain Schmitz
LSV, ENS Paris-Saclay & CNRS
Université Paris-Saclay
France

***Abstract*—Checking whether two pushdown automata with restricted silent actions are weakly bisimilar was shown decidable by Sénizergues (1998, 2005). We provide the first known complexity upper bound for this famous problem, in the equivalent setting of first-order grammars. This ACKERMANN upper bound is optimal, and we also show that strong bisimilarity is primitive-recursive when the number of states of the automata is fixed.**

***Key Words*—Pushdown system, first-order grammar, bisimilarity, fast-growing complexity**

## I. Introduction

*Bisimulation equivalence* plays a central role among the many notions of semantic equivalence studied in verification and concurrency theory [11]. Indeed, two bisimilar processes always satisfy exactly the same specifications written in modal logics [2] or in the modal $\mu$-calculus [14], allowing one to replace for instance a naive implementation with a highly optimised one without breaking the conformance. As a toy example, the two recursive Erlang functions below implement the same stateful message relaying service, that either receives {upd, M1} and updates its internal message from M to M1, or receives {rel ,C} and sends the message M to the client C.

```
1 serverA(M) →                          serverB(M) →
2   receive                               M2 = receive
3     {upd, M1} → serverA(M1);            {upd, M1} → M1;
4     {rel , C } → C!M,                   {rel , C } → C!M, M;
5                  serverA(M);            end,
6   end.                                  serverB(M2).
```

The two programs are weakly bisimilar if we only observe the input (**receive**) and output (C!M) actions, but the one on the left is not tail-recursive and might perform poorly compared to the one on the right.

In a landmark 1998 paper, Sénizergues [32, 34] proved the decidability of bisimulation equivalence for rooted equational graphs of finite out-degree. The proof extends his previous seminal result [31, 33], which is the decidability of language equivalence for deterministic pushdown automata (DPDA), and entails that weak bisimilarity of pushdown processes where silent actions are deterministic is decidable; a silent action (also called an $\varepsilon$-step) is deterministic if it has no alternative when enabled. Because the control flow of a first-order recursive program is readily modelled by a pushdown process, one can view this result as showing that the equivalence of recursive programs (like the two Erlang functions above) is decidable as far as their observable behaviours are concerned, provided silent moves are deterministic. Regarding decidability, Sénizergues' result is optimal in the sense that bisimilarity becomes undecidable if we consider either nondeterministic (popping) $\varepsilon$-steps [15], or second-order pushdown processes with no $\varepsilon$-steps [4]. Note that the decidability border was also refined in [39] by considering branching bisimilarity, a stronger version of weak bisimilarity.

*Computational Complexity:* While this delineates the decidability border for equivalences of pushdown processes, the computational complexity of the bisimilarity problem is open. Sénizergues' algorithm consists in two semi-decision procedures, with no clear means of bounding its complexity, and subsequent works like [18] have so far not proven easier to analyse. We know however that this complexity must be considerable, as the problem is TOWER-hard in the real-time case (i.e., without silent actions, hence for *strong* bisimilarity) [1] and ACKERMANN-hard in the general case (with deterministic silent actions) [19]—we are employing here the 'fast-growing' complexity classes defined in [29], where $\mathsf{TOWER} = \mathsf{F}_3$ is the lowest non elementary class and $\mathsf{ACKERMANN} = \mathsf{F}_\omega$ the lowest non primitive-recursive one.

In fact, the precise complexity of deciding equivalences for pushdown automata and their restrictions is often not known—as is commonplace with infinite-state processes [35]. For instance, language equivalence of deterministic pushdown automata is P-hard and was shown to be in TOWER by Stirling [37] (see [19] for an explicit upper bound), and bisimilarity of BPAs (i.e., real-time pushdown processes with a single state) is EXPTIME-hard [22] and in 2EXPTIME [5] (see [17] for an explicit proof). There are also a few known completeness results in restricted cases: bisimilarity of normed BPAs is P-complete [13] (see [10] for the best known upper bound), bisimilarity of real-time one-counter processes (i.e., of pushdown processes with a singleton stack alphabet) is PSPACE-complete [3], and bisimilarity of visibly pushdown processes is EXPTIME-complete [36].

*Contributions:* In this paper, we prove that the bisimilarity problem for pushdown processes is in ACKERMANN, even the weak bisimilarity problem when silent actions are deterministic. Combined with the already mentioned lower bound from [19],

978-1-7281-3608-0/19/$31.00 

TABLE I
THE COMPLEXITY OF EQUIVALENCE PROBLEMS OVER PUSHDOWN PROCESSES

| Problem | Lower bound | Upper bound |
|---|---|---|
| DPDA lang. equ. | P | TOWER [37, 19] |
| strong bisim. | TOWER [1] | ACKERMANN [this paper] |
| weak bisim.[a] | ACKERMANN [19] | ACKERMANN [this paper] |

[a] silent actions must be deterministic

this shows the problem to be ACKERMANN-complete. This is the first instance of a complexity completeness result in the line of research originating from Sénizergues' work [31, 32, 33, 34]; see Table I.

Rather than working with rooted equational graphs of finite out-degree or with pushdown processes with deterministic silent actions, our proof is cast in the formalism of *first-order grammars* (see Section II), which are term rewriting systems with a head rewriting semantics, and are known to generate the same class of graphs [7].

Our proof heavily relies on the main novelty from [18]: the bisimilarity of two arbitrary terms according to a first-order grammar essentially hinges on a finite *basis* of pairs of *non-equivalent terms*, which can be constructed from the grammar independently of the terms provided as input. The basis provides a number that allows us to compute a bound on the 'equivalence-level' of two non-equivalent terms; this is the substance of the decision procedure (see Section III). Both in [18] and in its reworked version in [21], such a basis is obtained through a brute force argument, which yields no complexity statement. In Section IV we exhibit a concrete algorithm computing the basis, and we analyse its complexity in the framework of [28, 29, 30] in Section V, yielding the ACKERMANN upper bound.

Finally, although our results do not match the TOWER lower bound of Benedikt et al. [1] in the case of real-time pushdown processes, we nevertheless show in Section VI that bisimilarity becomes primitive-recursive in that case if additionally the number of control states of the pushdown processes is fixed.

## II. FIRST-ORDER GRAMMARS

*First-order grammars* are labelled term rewriting systems with a head rewriting semantics. They are a natural model of first-order functional programs with a call-by-name semantics, and were shown to generate the class of rooted equational graphs of finite out-degree by Caucal [6, 7], where they are called *term context-free grammars*. Here we shall use the terminology and notations from [21].

### A. Regular Terms

Let $\mathcal{N}$ be a finite ranked alphabet, i.e., where each symbol $A$ in $\mathcal{N}$ comes with an arity $r(A)$ in $\mathbb{N} \stackrel{\text{def}}{=} \{0, 1, 2, \dots\}$, and $\text{VAR} \stackrel{\text{def}}{=} \{x_1, x_2, \dots\}$ a countable set of variables, all with arity zero. We work with possibly infinite *regular terms* over $\mathcal{N}$ and VAR, i.e., terms with finitely many distinct subterms. Let $\text{TERMS}_{\mathcal{N}}$ denote the set of all regular terms over $\mathcal{N}$ and VAR.

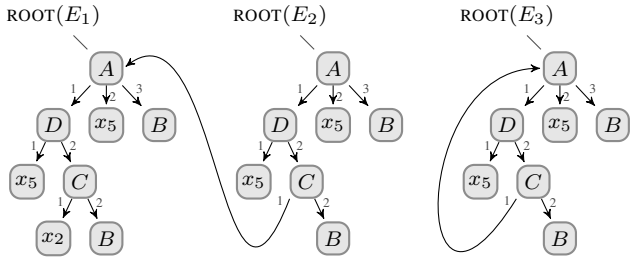


Fig. 1. Graph representations of two finite terms $E_1$ and $E_2$, and of an infinite regular term $E_3$.

We further use $A, B, C, D$ for nonterminals, and $E, F$ for terms, possibly primed and/or with subscripts.

*1) Representations:* Such terms can be represented by finite directed graphs as shown in Figure 1, where each node has a label in $\mathcal{N} \cup \text{VAR}$ and a number of ordered outgoing arcs equal to its arity. The unfolding of the graph representation is the desired term, and there is a bijection between the nodes of the *least* graph representation of a term $E$ and the set of subterms of $E$.

*2) Size and Height:* We define the *size* $\text{SIZE}(E)$ of a term $E$ as its number of distinct subterms. For instance, $\text{SIZE}(E_1) = 6$, $\text{SIZE}(E_2) = 9$, and $\text{SIZE}(E_3) = 5$ in Figure 1. For two terms $E$ and $F$, we also denote by $\text{SIZE}(E, F)$ the number of distinct subterms of $E$ and $F$; note that $\text{SIZE}(E, F)$ can be smaller than $\text{SIZE}(E) + \text{SIZE}(F)$, as they might share some subterms. For instance, $\text{SIZE}(E_1, E_2) = 9$ in Figure 1. We let $\text{NTSIZE}(E)$ denote the number of distinct subterms of $E$ with root labels in $\mathcal{N}$; e.g., $\text{NTSIZE}(E_1) = 4$ in Figure 1. A term $E$ is thus *finite* if and only if its graph representation is acyclic, in which case it has a *height* $\text{HEIGHT}(E)$, which is the maximal length of a path from the root to a leaf; for instance $\text{HEIGHT}(E_1) = 3$ in Figure 1. Finally, we let $\text{VAR}(E)$ denote the set of variables occurring in $E$, and let $\text{VAR}(E, F) \stackrel{\text{def}}{=} \text{VAR}(E) \cup \text{VAR}(F)$; e.g., $\text{VAR}(E_1, E_2) = \{x_2, x_5\}$ in Figure 1.

### B. Substitutions

A *substitution* $\sigma$ is a map $\text{VAR} \to \text{TERMS}_{\mathcal{N}}$ whose *support* $\text{SUPP}(\sigma) \stackrel{\text{def}}{=} \{x \in \text{VAR} \mid \sigma(x) \neq x\}$ is finite. This map is lifted to act over terms by

$$x\sigma \stackrel{\text{def}}{=} \sigma(x) \ , \quad A(E_1, \dots, E_{r(A)})\sigma \stackrel{\text{def}}{=} A(E_1\sigma, \dots, E_{r(A)}\sigma)$$

for all $x$ in VAR, $A$ in $\mathcal{N}$, and $E_1, \dots, E_{r(A)}$ in $\text{TERMS}_{\mathcal{N}}$. For instance, in Figure 1, $E_2 = E_1\sigma$ if $\sigma(x_2) = E_1$ and $\sigma(x_5) = x_5$.

### C. Grammars

A *first-order grammar* is a tuple $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$ where $\mathcal{N}$ is a finite ranked alphabet of nonterminals, $\Sigma$ a finite alphabet of actions, and $\mathcal{R}$ a finite set of labelled term rewriting rules of the form $A(x_1, \dots, x_{r(A)}) \stackrel{a}{\to} E$ where $A \in \mathcal{N}$, $a \in \Sigma$, and $E$ is a finite term in $\text{TERMS}_{\mathcal{N}}$ with $\text{VAR}(E) \subseteq \{x_1, \dots, x_{r(A)}\}$.

*1) Head Rewriting Semantics:* A first-order grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$ defines an infinite *labelled transition system*

$$\mathcal{L}_\mathcal{G} \stackrel{\text{def}}{=} (\textsc{Terms}_\mathcal{N}, \Sigma, (\xrightarrow{a})_{a\in\Sigma})$$

over $\textsc{Terms}_\mathcal{N}$ as set of states, $\Sigma$ as set of actions, and with a transition relation $\xrightarrow{a} \subseteq \textsc{Terms}_\mathcal{N} \times \textsc{Terms}_\mathcal{N}$ for each $a \in \Sigma$, where each rule $A(x_1, \ldots, x_{r(A)}) \xrightarrow{a} E$ of $\mathcal{R}$ induces a transition

$$A(x_1, \ldots, x_{r(A)})\sigma \xrightarrow{a} E\sigma$$

for every substitution $\sigma$. This means that rewriting steps can only occur at the root of a term, rather than inside a context. For instance, the rules $A(x_1, x_2, x_3) \xrightarrow{a} C(x_2, D(x_2, x_1))$ and $A(x_1, x_2, x_3) \xrightarrow{b} x_2$ give rise on the terms of Figure 1 to the transitions $E_1 \xrightarrow{a} C(x_5, D(x_5, D(x_5, C(x_2, B))))$ and $E_1 \xrightarrow{b} x_5$. The transition relations $\xrightarrow{a}$ are extended to $\xrightarrow{w}$ for words $w \in \Sigma^*$ in the standard way.

Note that variables $x \in \textsc{Var}$ are 'dead', in that no transitions can be fired from a variable. In fact, in Section III-A we discuss that for technical reasons we could formally assume that each variable $x$ has its unique action $a_x$ and a transition $x \xrightarrow{a_x} x$.

*2) Grammatical Constants:* Let us fix a first-order grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$. We define its size as

$$|\mathcal{G}| \stackrel{\text{def}}{=} \sum_{A(x_1,\ldots,x_{r(A)})\xrightarrow{a}E \,\in\mathcal{R}} r(A) + 1 + \textsc{size}(E)\ . \quad (1)$$

Let $\textsc{rhs}$ be the set of terms appearing on the right-hand sides of $\mathcal{R}$ (which are finite terms by definition). We let

$$m \stackrel{\text{def}}{=} \max_{A\in\mathcal{N}} r(A)\ , \quad (2)$$

$$\textsc{hinc} \stackrel{\text{def}}{=} \max_{E\in\textsc{rhs}} \textsc{height}(E) - 1\ , \quad (3)$$

$$\textsc{sinc} \stackrel{\text{def}}{=} \max_{E\in\textsc{rhs}} \textsc{ntsize}(E) \quad (4)$$

bound respectively the maximal arity of its nonterminals, its maximal height increase in one transition step, and its maximal size increase in one transition step.

If $A(x_1, \ldots, x_{r(A)}) \xrightarrow{w} x_i$ in $\mathcal{L}_\mathcal{G}$ for some $i$ in $\{1, \ldots, r(A)\}$ and $w$ in $\Sigma^*$, then we call $w$ an $(A, i)$-*sink word*. Observe that $w \neq \varepsilon$, hence $w = aw'$ with $A(x_1, \ldots, x_{r(A)}) \xrightarrow{a} E$ in $\mathcal{R}$ and $E \xrightarrow{w'} x_i$, where either $w' = \varepsilon$ and $E = x_i$ or $E$ 'sinks' to $x_i$ when applying $w'$. Thus, for each $A \in \mathcal{N}$ and $i \in \{1, \ldots, r(A)\}$ we can compute some shortest $(A, i)$-sink word $w_{[A,i]}$ by dynamic programming; in the cases where no $(A, i)$-sink word exist, we can formally put $w_{[A,i]} \stackrel{\text{def}}{=} \varepsilon$. In turn, this entails that the maximal length of shortest sink words satisfies

$$d_0 \stackrel{\text{def}}{=} 1 + \max_{A\in\mathcal{N}, 1\leq i\leq r(A)} |w_{[A,i]}| \leq 1 + (2 + \textsc{hinc})^{|\mathcal{N}|m}\ ; \quad (5)$$

here and in later instances, we let $\max\emptyset \stackrel{\text{def}}{=} 0$.

Finally, the following grammatical constant $n$ from [21] is important for us:

$$n \stackrel{\text{def}}{=} m^{d_0}\ ; \quad (6)$$

note that $n$ is at most doubly exponential in the size of $\mathcal{G}$. This $n$ was chosen in [21] so that each $E$ can be written as $E'\sigma$ where $\textsc{height}(E') \leq d_0$ and $\textsc{var}(E') \subseteq \{x_1, \ldots, x_n\}$, and it is guaranteed that each path $E \xrightarrow{w} F$ where $|w| \leq d_0$ can be presented as $E'\sigma \xrightarrow{w} F'\sigma$ where $E' \xrightarrow{w} F'$. Put simply: $n$ bounds the number of depth-$d_0$ subterms for each term $E$.

## III. Bisimulation Equivalence

Bisimulation equivalence has been introduced independently in the study of modal logics [2] and in that of concurrent processes [25, 26]. We recall here the basic notions surrounding bisimilarity before we introduce the key notion of *candidate bases* as defined in [21].

### A. Equivalence Levels

Consider a labelled transition system

$$\mathcal{L} = (\mathcal{S}, \Sigma, (\xrightarrow{a})_{a\in\Sigma})$$

like the one defined by a first-order grammar, with set of states $\mathcal{S}$, set of actions $\Sigma$, and a transition relation $\xrightarrow{a} \subseteq \mathcal{S} \times \mathcal{S}$ for each $a$ in $\Sigma$. We work in this paper with *image-finite* labelled transition systems, where $\{s' \in \mathcal{S} \mid s \xrightarrow{a} s'\}$ is finite for every $s$ in $\mathcal{S}$ and $a$ in $\Sigma$. In this setting, the coarsest (strong) *bisimulation* $\sim$ can be defined through a chain

$$\sim_0 \supseteq \sim_1 \supseteq \cdots \supseteq \sim$$

of equivalence relations over $\mathcal{S} \times \mathcal{S}$: let $\sim_0 \stackrel{\text{def}}{=} \mathcal{S} \times \mathcal{S}$ and for each $k$ in $\mathbb{N}$, let $s \sim_{k+1} t$ if $s \sim_k t$ and

- (zig) if $s \xrightarrow{a} s'$ for some $a \in \Sigma$, then there exists $t'$ such that $t \xrightarrow{a} t'$ and $s' \sim_k t'$, and
- (zag) if $t \xrightarrow{a} t'$ for some $a \in \Sigma$, then there exists $s'$ such that $s \xrightarrow{a} s'$ and $s' \sim_k t'$.

We put $\sim_\omega \stackrel{\text{def}}{=} \bigcap_{k\in\mathbb{N}} \sim_k$; hence $\sim = \sim_\omega$.

For each pair $s, t$ of states in $\mathcal{S}$, we may then define its *equivalence level* $\textsc{el}(s, t)$ in $\omega + 1 = \mathbb{N} \uplus \{\omega\}$ as

$$\textsc{el}(s, t) \stackrel{\text{def}}{=} \sup\{k \in \mathbb{N} \mid s \sim_k t\}\ . \quad (7)$$

Here we should add that—to be consistent with [21]—we stipulate that $\textsc{el}(x, E) = 0$ when $E \neq x$; in particular $\textsc{el}(x_i, x_j) = 0$ when $i \neq j$. This would automatically hold if we equipped each $x \in \textsc{Var}$ with a special transition $x \xrightarrow{a_x} x$ in $\mathcal{L}_\mathcal{G}$, as we already mentioned. This stipulation guarantees that $\textsc{el}(E, F) \leq \textsc{el}(E\sigma, F\sigma)$.

Two states $s, t$ are (strongly) *bisimilar* if $s \sim t$, which is if and only if $\textsc{el}(s, t) = \omega$. We will later show an algorithm computing the equivalence level of two given terms in the labelled transition system defined by a given first-order grammar. The main decision problem in which we are interested is the following.

**Problem** (Bisimulation)**.**

- input A first-order grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$ and two terms $E, F$ in $\textsc{Terms}_\mathcal{N}$.
- question Is $\textsc{el}(E, F) = \omega$ in the labelled transition system $\mathcal{L}_\mathcal{G}$?

### B. Bisimulation Game

Observe that the following variant of the bisimulation problem is decidable.

**Problem** (Bounded Equivalence Level)**.**

input A first-order grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$, two terms $E, F$ in $\text{TERMS}_{\mathcal{N}}$, and $e$ in $\mathbb{N}$.

question Is $\text{EL}(E, F) \leq e$ in the labelled transition system $\mathcal{L}_{\mathcal{G}}$?

Indeed, as is well-known, the zig-zag condition can be recast as a *bisimulation game* between two players called Spoiler and Duplicator. A position of the game is a pair $(s_1, s_2) \in \mathcal{S} \times \mathcal{S}$. Spoiler wants to prove that the two states are not bisimilar, while Duplicator wants to prove that they are bisimilar. The game proceeds in rounds; in each round,

- Spoiler chooses $i \in \{1, 2\}$ and a transition $s_i \xrightarrow{a} s_i'$ (if no such transition exists, Spoiler loses), then
- Duplicator chooses a transition $s_{3-i} \xrightarrow{a} s_{3-i}'$ with the same label $a$ (if no such transition exists, Duplicator loses);

the game then proceeds to the next round from position $(s_1', s_2')$. Then $\text{EL}(s_1, s_2) \leq k$ if and only if Spoiler has a strategy to win in the $(k+1)$th round at the latest when starting the game from $(s_1, s_2)$. Note that this game is determined and memoryless strategies suffice.

Thus, the bounded equivalence level problem can be solved by an alternating Turing machine that first writes the representation of $E$ and $F$ on its tape, and then plays at most $e$ rounds of the bisimulation game, where each round requires at most a polynomial number of computational steps in the size of the grammar (assuming a somewhat reasonable tape encoding of the terms).

**Fact 1.** *The bounded equivalence level problem is in* $\mathsf{ATIME}\big(\text{SIZE}(E, F) + \mathsf{poly}(|\mathcal{G}|) \cdot e\big)$.

### C. Candidate Bases

Consider some fixed first-order grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$. Given three numbers $n$, $s$, and $g$ in $\mathbb{N}$—which will depend on $\mathcal{G}$—, an $(n, s, g)$*-candidate basis* for non-equivalence is a set of pairs of terms $\mathcal{B} \subseteq \text{TERMS}_{\mathcal{N}} \times \text{TERMS}_{\mathcal{N}}$ associated with two sequences of numbers $(s_i)_{0 \leq i \leq n}$ and $(e_i)_{0 \leq i \leq n}$ such that

1) $\mathcal{B} \subseteq \nsim$,
2) for each $(E, F) \in \mathcal{B}$ there is $i \in \{0, \ldots, n\}$ such that $\text{VAR}(E, F) = \{x_1, \ldots, x_i\}$ and $\text{SIZE}(E, F) \leq s_i$,
3) $s_n \stackrel{\text{def}}{=} s$, and the remaining numbers are defined inductively by

$$e_i \stackrel{\text{def}}{=} \max_{(E,F) \in \mathcal{B} \wedge \text{SIZE}(E,F) \leq s_i} \text{EL}(E, F) \ , \tag{8}$$

$$s_{i-1} \stackrel{\text{def}}{=} 2s_i + g + e_i(\text{SINC} + g) \ . \tag{9}$$

Note that the numbers $(s_i)_{0 \leq i \leq n}$ and $(e_i)_{0 \leq i \leq n}$ are entirely determined by $\mathcal{B}$ and $n$, $s$, and $g$. An $(n, s, g)$-candidate basis $\mathcal{B}$ yields a *bound* $\mathcal{E}_{\mathcal{B}}$ defined by

$$\mathcal{E}_{\mathcal{B}} \stackrel{\text{def}}{=} n + 1 + \sum_{i=0}^{n} e_i \ . \tag{10}$$

*1) Full Bases:* For $0 \leq i \leq n$, let

$$\text{PAIRS}_i \stackrel{\text{def}}{=} \{(E, F) \mid \exists j \leq i \,.\, \text{VAR}(E, F) = \{x_1, \ldots, x_j\} \wedge \text{SIZE}(E, F) \leq s_i\} \ . \tag{11}$$

An $(n, s, g)$-candidate basis $\mathcal{B}$ is *full below* some equivalence level $e \in \omega + 1$ if, for all $0 \leq i \leq n$ and all $(E, F) \in \text{PAIRS}_i$ such that $\text{EL}(E, F) < e$ we have $(E, F) \in \mathcal{B}$. We say that $\mathcal{B}$ is *full* if it is full below $\omega$. In other words and because $\mathcal{B} \subseteq \nsim$, $\mathcal{B}$ is full if and only if, for all $0 \leq i \leq n$, $\text{PAIRS}_i \setminus \mathcal{B} \subseteq \sim$.

**Proposition 2** ([21, Prop. 9])**.** *For any $n, s, g$, there is a unique full $(n, s, g)$-candidate basis, denoted by $\mathcal{B}_{n,s,g}$.*

*Proof.* The full candidate basis $\mathcal{B}_{n,s,g}$ is constructed by induction over $n$. Let $s_n \stackrel{\text{def}}{=} s$ and consider the finite set $S_n \stackrel{\text{def}}{=} \{(E, F) \in \text{TERMS}_{\mathcal{N}} \times \text{TERMS}_{\mathcal{N}} \mid E \nsim F \wedge \exists j \leq n \,.\, \text{VAR}(E, F) = \{x_1, \ldots, x_j\} \wedge \text{SIZE}(E, F) \leq s_n\}$; $S_n$ has a maximal equivalence level $e_n \stackrel{\text{def}}{=} \max_{(E,F) \in S_n} \text{EL}(E, F)$. If $n = 0$, we define $\mathcal{B}_{0,s,g} \stackrel{\text{def}}{=} S_0$. Otherwise, we let $s_{n-1} \stackrel{\text{def}}{=} 2s_n + g + e_n(\text{SINC} + g)$ as in (9); by induction hypothesis there is a unique full $(n-1, s_{n-1}, g)$-candidate basis $\mathcal{B}_{n-1,s_{n-1},g}$ and we set $\mathcal{B}_{n,s,g} \stackrel{\text{def}}{=} S_n \cup \mathcal{B}_{n-1,s_{n-1},g}$. □

*2) Main Result:* The main result from [21] can now be stated.

**Theorem 3** ([21, Thm. 7])**.** *Let $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$ be a first-order grammar. Then one can compute a grammatical constant $g$ exponential in $|\mathcal{G}|$ and grammatical constants $n$, $s$, and $c$ doubly exponential in $|\mathcal{G}|$ such that, for all terms $E, F$ in $\text{TERMS}_{\mathcal{N}}$ with $E \nsim F$,*

$$\text{EL}(E, F) \leq c \cdot \big(\mathcal{E}_{\mathcal{B}_{n,s,g}} \cdot \text{SIZE}(E, F) + \text{SIZE}(E, F)^2\big) \ .$$

Theorem 3 therefore shows that the bisimulation problem can be reduced to the bounded equivalence level problem, provided one can compute the full $(n, s, g)$-candidate basis for suitable $n$, $s$, and $g$—see Table II in the appendix for details on how the grammatical constants $n$, $s$, $c$, and $g$ are defined in [21]. Our goal in Section IV will thus be to exhibit a concrete algorithm computing the full candidate basis $\mathcal{B}_{n,s,g}$, in order to derive an upper bound on $\mathcal{E}_{\mathcal{B}_{n,s,g}}$.

The proof of [21, Thm. 7] relies on the following insight, which we will also need in order to prove the correctness of our algorithm.

**Lemma 4** ([21, Eq. 39])**.** *Let $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$ be a first-order grammar, $g, n, s, c$ be defined as in Theorem 3, $E, F$ be two terms in $\text{TERMS}_{\mathcal{N}}$ with $E \nsim F$, and $\mathcal{B}$ be an $(n, s, g)$-candidate basis full below $\text{EL}(E, F)$. Then*

$$\text{EL}(E, F) \leq c \cdot \big(\mathcal{E}_{\mathcal{B}} \cdot \text{SIZE}(E, F) + \text{SIZE}(E, F)^2\big) \ .$$

## IV. Computing Candidate Bases

Theorem 3 shows that, in order to solve the bisimulation problem, it suffices to compute $c$ and $\mathcal{E}_{\mathcal{B}_{n,s,g}}$ and then solve the bounded equivalence problem, for which Fact 1 provides a complexity upper bound. In this section, we show how to compute $\mathcal{E}_{\mathcal{B}_{n,s,g}}$ for an input first-order grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$. Note that this grammatical constant was shown computable in [18, 21] through a brute-force argument, but here we want a concrete algorithm, whose complexity will be analysed in Section V. We proceed in two steps, by first considering a non effective version of the algorithm in Section IV-A, whose correctness is straightforward, and then the actual algorithm in Section IV-B.

### A. Non Effective Version

Throughout this section, we consider $n$ as a fixed parameter. We first assume that we have an oracle $\textsc{EqLevel}(\mathcal{G}, \mathcal{E}_{\mathcal{B}}, c, E, F)$ at our disposal, that returns the equivalence level $\textsc{el}(E, F)$ in $\mathcal{L}_{\mathcal{G}}$; the parameters $\mathcal{E}_{\mathcal{B}}, c$ will be used in the effective version in Section IV-B. The following procedure then constructs full $(n, s, g)$-candidate basis $\mathcal{B}_{n,s,g}$ and its associated bound $\mathcal{E}_{\mathcal{B}_{n,s,g}}$, by progressively adding pairs from the sets $\textsc{Pairs}_i$ until the candidate basis is full. In order not to clutter the presentation too much, we assume implicitly that the equivalence level $e$ of each pair $(E, F)$ added to $\mathcal{B}$ on line 15 is implicitly stored, thus it does not need to be recomputed on line 20.

```
1  procedure CandidateBound_n(G, s, g, c)
2      B ← ∅                                          ▷ Initialisation
3      for i ← 0, ..., n do
4          e_i ← 0
5      s_n ← s
6      for i ← n − 1, ..., 0 do
7          s_i ← 2 s_{i+1} + g
8      E_B ← n + 1
9      for i ← n, ..., 0 do
10         P_i ← Pairs_i \ ⋃_{i<j≤n} P_j
11     while ∃i ∈ {0, 1, ..., n}, ∃(E, F) ∈ P_i :
12      EqLevel(G, E_B, c, E, F) < ω do               ▷ Main loop
13         e ← EqLevel(G, E_B, c, E, F)
14         P_i ← P_i \ {(E, F)}
15         B ← B ∪ {(E, F)}
16         if e > e_i then                            ▷ If so, then update
17             e_i ← e
18             for j ← i − 1, ..., 0 do
19                 s_j ← 2 s_{j+1} + g + e_{j+1}(sinc + g)
20                 e_j ← max_{(E,F)∈B ∧ size(E,F)≤s_j} el(E, F)
21                 P_j ← Pairs_j \ (B ∪ ⋃_{j<k≤n} P_k)
22             E_B ← n + 1 + Σ_{0≤j≤n} e_j
23     return E_B
```

*1) Invariant:* The procedure $\textsc{CandidateBound}_n$ maintains as an invariant of its main loop on lines 12–22 that $\mathcal{B}$ is an $(n, s, g)$-candidate basis associated with the numbers $(s_i)_{0 \le i \le n}$ and $(e_i)_{0 \le i \le n}$, and that $\mathcal{E}_{\mathcal{B}}$ is its associated bound. This holds indeed after the initialisation phase on lines 2–8, and is enforced in the main loop by the update instructions on lines 16–22.

*2) Correctness:* Let us check that, if it terminates, this non effective version does indeed return the bound $\mathcal{E}_{\mathcal{B}_{n,s,g}}$ associated with the unique full $(n, s, g)$-candidate basis $\mathcal{B}_{n,s,g}$. By the previous invariant, it suffices to show that $\mathcal{B}$ is full when the procedure terminates. Consider for this some index $0 \le i \le n$ and a pair $(E, F) \in \textsc{Pairs}_i$ with $\textsc{el}(E, F) = e$ for some $e < \omega$. By definition of the sets $(\mathcal{P}_i)_{0 \le i \le n}$ on lines 9–10 and their updates on lines 14 and 21 in the main loop, the pair $(E, F)$ must have been added to some $\mathcal{P}_j$ for $j \ge i$. Then the pair must have been selected by the condition of the main loop on line 12, and added to $\mathcal{B}$.

*3) Termination:* Although we are still considering a non effective version of the algorithm, the proof that it always terminates is the same as the one for the effective version in Section IV-B. We exhibit a ranking function on the main loop, thereby showing that it must stop eventually. More precisely, each time we enter the main loop on line 12, we associate to the current state of the procedure the ordinal rank below $\omega^{n+1}$ defined by

$$\alpha \stackrel{\text{def}}{=} \omega^n \cdot |\mathcal{P}_n| + \cdots + \omega^0 \cdot |\mathcal{P}_0| \,. \tag{12}$$

Note that this is equivalent to defining the rank as the tuple $(|\mathcal{P}_n|, \ldots, |\mathcal{P}_0|)$ in $\mathbb{N}^{n+1}$, ordered lexicographically, but ordinal notations are more convenient for our analysis in Section V. This defines a descending sequence of ordinals

$$\alpha_0 > \alpha_1 > \cdots \,, \tag{13}$$

where $\alpha_\ell$ is the rank after $\ell$ iterations of the main loop. Indeed, each time we enter the loop, the cardinal $|\mathcal{P}_i|$ of the set under consideration strictly decreases on line 14, and is not modified by the updates on line 21, which only touch the sets $\mathcal{P}_j$ for $j < i$. Hence $\textsc{CandidateBound}_n$ terminates.

### B. Effective Version

In order to render $\textsc{CandidateBound}_n$ effective, we provide an implementation of $\textsc{EqLevel}$ that does not require an oracle for the bisimulation problem, but relies instead on Lemma 4 and the bounded equivalence level problem, which as we saw in Section III-B is decidable.

```
1  procedure EqLevel(G, E_B, c, E, F)
2      if el(E, F) ≤ c · (E_B · size(E, F) + size(E, F)^2)
       then
3          return el(E, F)
4      else
5          return ω
```

We establish the correctness of this effective variant in the following theorem, which uses the same reasoning as the proof of [21, Thm. 7].

**Theorem 5.** *The effective version of procedure* $\textsc{CandidateBound}_n(\mathcal{G}, s, g, c)$ *terminates and, provided* $n$*,* $s$*,* $c$*, and* $g$ *are defined as in Theorem 3, returns the bound* $\mathcal{E}_{\mathcal{B}_{n,s,g}}$.

*Proof.* Termination is guaranteed by the ranking function defined by (12). Regarding correctness, assume the provided $g$, $n$, $s$, and $c$ are defined as in Theorem 3, and let us define a (reflexive and symmetric) relation $\dot{\sim}_k$ on $\textsc{Terms}_{\mathcal{N}}$ by $E \mathrel{\dot{\sim}_k} F$ if and only if $\textsc{el}(E,F) > c \cdot \big(k \cdot \textsc{size}(E,F) + \textsc{size}(E,F)^2\big)$. Clearly, $\sim \subseteq \dot{\sim}_k$ for all $k$ in $\mathbb{N}$. We say that an $(n,s,g)$-candidate basis $\mathcal{B}$ is *$k$-complete* if, for all $0 \leq i \leq n$, $\textsc{Pairs}_i \setminus \mathcal{B} \subseteq \dot{\sim}_k$. We call $\mathcal{B}$ *complete* if it is $\mathcal{E}_{\mathcal{B}}$-complete. By the reasoning we used for showing the correctness of the non effective version, when the effective version of $\textsc{CandidateBound}_n$ terminates, $\mathcal{B}$ is complete.

It remains to show that $\mathcal{B}$ is complete if and only if it is full. First observe that, if $\mathcal{B}$ is full, then it is complete: indeed, $\mathcal{B}$ being full entails that, for all $E \nsim F$ in $\textsc{Pairs}_i$, $(E,F)$ is in $\mathcal{B} \subseteq \nsim$, hence $\textsc{Pairs}_i \setminus \mathcal{B} \subseteq \sim \subseteq \dot{\sim}_{\mathcal{E}_{\mathcal{B}}}$.

Conversely, assume that $\mathcal{B}$ is complete, and let us show that it is full; it suffices to show that, in that case, $\dot{\sim}_{\mathcal{E}_{\mathcal{B}}} \subseteq \sim$. By contradiction, consider a pair $E \nsim F$ with $E \mathrel{\dot{\sim}_{\mathcal{E}_{\mathcal{B}}}} F$; without loss of generality, $\textsc{el}(E,F)$ can be assumed minimal among all such pairs. Then $\mathcal{B}$ is full below $\textsc{el}(E,F)$: indeed, if $(E',F') \in \textsc{Pairs}_i$ and $\textsc{el}(E',F') < \textsc{el}(E,F)$, since $\textsc{el}(E,F)$ was taken minimal, $E' \mathrel{\dot{\nsim}_{\mathcal{E}_{\mathcal{B}}}} F'$ and therefore $(E',F')$ belongs to $\mathcal{B}$ since $\mathcal{B}$ is complete. Thus Lemma 4 applies and shows that $E \mathrel{\dot{\nsim}_{\mathcal{E}_{\mathcal{B}}}} F$, a contradiction. □

## V. Complexity Upper Bounds

In this section, we analyse the procedure $\textsc{CandidateBound}_n$ to derive an upper bound on the computed $\mathcal{E}_{\mathcal{B}}$. In turn, by Fact 1 and Theorem 3, this bound will allow us to bound the complexity of the bisimulation problem. The idea is to analyse the ranking function defined by (12) in order to bound how many times the main loop of $\textsc{CandidateBound}_n$ can be executed. We rely for this on a so-called 'length function theorem' from [28] to bound the length of descending sequences of ordinals like (13). Finally, we classify the final upper bound using the 'fast-growing' complexity classes defined in [29]. A general introduction to these techniques can be found in [30]. Throughout this section, we assume that the values of $g$, $n$, $s$, and $c$ are the ones needed for Theorem 3 to hold.

### A. Controlled Descending Sequences

Though all descending sequences of ordinals are finite, we cannot bound their lengths in general; e.g., $K+1 > K > K-1 > \cdots > 0$ and $\omega > K > K-1 > \cdots > 0$ are descending sequences of length $K+2$ for all $K$ in $\mathbb{N}$. Nevertheless, the sequence (13) produced by $\textsc{CandidateBound}_n$ is not arbitrary, because the successive ranks are either determined by the input and the initialisation phase, or the result of some computation, hence one cannot use an arbitrary $K$ as in these examples.

This intuition is captured by the notion of *controlled sequences*. For an ordinal $\alpha < \omega^\omega$ (like the ranks defined by (12)), let us write $\alpha$ in Cantor normal form as

$$\alpha = \omega^n \cdot c_n + \cdots + \omega^0 \cdot c_0$$

with $c_0, \ldots, c_n$ and $n$ in $\mathbb{N}$, and define its *size* as

$$\|\alpha\| \stackrel{\text{def}}{=} \max\{n, \max_{0 \leq i \leq n} c_i\} \,. \tag{14}$$

Let $N_0$ be a natural number in $\mathbb{N}$ and $h\colon \mathbb{N} \to \mathbb{N}$ a monotone inflationary function, i.e., $x \leq y$ implies $h(x) \leq h(y)$, and $x \leq h(x)$. A sequence $\alpha_0, \alpha_1, \ldots$ of ordinals below $\omega^\omega$ is *$(N_0,h)$-controlled* if, for all $\ell$ in $\mathbb{N}$,

$$\|\alpha_\ell\| \leq h^\ell(N_0) \,, \tag{15}$$

i.e., the size of the $\ell$th ordinal $\alpha_\ell$ is bounded by the $\ell$th iterate of $h$ applied to $N_0$; in particular, $\|\alpha_0\| \leq N_0$. Because for each $N \in \mathbb{N}$, there are only finitely many ordinals below $\omega^\omega$ of size at most $N$, the length of controlled descending sequences is bounded [see, e.g., 28]. One can actually give a precise bound on this length in terms of *subrecursive functions*, whose definition we are about to recall.

### B. Subrecursive Functions

Algorithms shown to terminate via an ordinal ranking function can have a very high worst-case complexity. In order to express such large bounds, a convenient tool is found in subrecursive hierarchies, which employ recursion over ordinal indices to define faster and faster growing functions. We define here two such hierarchies.

*1) Fundamental Sequences:* A *fundamental sequence* for a limit ordinal $\lambda$ is a strictly ascending sequence $(\lambda(x))_{x<\omega}$ of ordinals $\lambda(x) < \lambda$ with supremum $\lambda$. We use the standard assignment of fundamental sequences to limit ordinals $\lambda \leq \omega^\omega$, defined inductively by

$$\omega^\omega(x) \stackrel{\text{def}}{=} \omega^{x+1} \,, \quad (\beta + \omega^{k+1})(x) \stackrel{\text{def}}{=} \beta + \omega^k \cdot (x+1) \,,$$

where $\beta + \omega^{k+1}$ is in Cantor normal form. This particular assignment satisfies, e.g., $0 < \lambda(x) < \lambda(y)$ for all $x < y$. For instance, $\omega(x) = x+1$ and $(\omega^3 + \omega^3 + \omega)(x) = \omega^3 + \omega^3 + x + 1$.

*2) Hardy and Cichoń Hierarchies:* In the context of controlled sequences, the hierarchies of Hardy and Cichoń turn out to be especially well-suited [8]. Let $h\colon \mathbb{N} \to \mathbb{N}$ be a function. For each such $h$, the *Hardy hierarchy* $(h^\alpha)_{\alpha \leq \omega^\omega}$ and the *Cichoń hierarchy* $(h_\alpha)_{\alpha \leq \omega^\omega}$ relative to $h$ are two families of functions $h^\alpha, h_\alpha\colon \mathbb{N} \to \mathbb{N}$ defined by induction over $\alpha$ by

$$\begin{aligned} h^0(x) &\stackrel{\text{def}}{=} x \,, & h_0(x) &\stackrel{\text{def}}{=} 0 \,, \\ h^{\alpha+1}(x) &\stackrel{\text{def}}{=} h^\alpha(h(x)) \,, & h_{\alpha+1}(x) &\stackrel{\text{def}}{=} 1 + h_\alpha(h(x)) \,, \\ h^\lambda(x) &\stackrel{\text{def}}{=} h^{\lambda(x)}(x) \,, & h_\lambda(x) &\stackrel{\text{def}}{=} h_{\lambda(x)}(x) \,. \end{aligned}$$

The Hardy functions are well-suited for expressing a large number of iterations of the provided function $h$. For instance, $h^k$ for some finite $k$ is simply the $k$th iterate of $h$. This intuition carries over: $h^\alpha$ is a 'transfinite' iteration of the function $h$, using a kind of diagonalisation in the fundamental sequences to handle limit ordinals. For instance, if we use the successor function $H(x) = x+1$ as our function $h$, we see that a first diagonalisation yields $H^\omega(x) = H^{x+1}(x) = 2x+1$. The next diagonalisation occurs at $H^{\omega \cdot 2}(x) = H^{\omega + x + 1}(x) = H^\omega(2x+1) = 4x+3$. Fast-forwarding a bit, we get for instance a function of exponential growth $H^{\omega^2}(x) = 2^{x+1}(x+1) - 1$,

and later a non-elementary function $H^{\omega^3}$ akin to a tower of exponentials, and a non primitive-recursive function $H^{\omega^\omega}$ of Ackermannian growth.

In the following, we will use the following property of Hardy functions [38, 8], which can be checked by induction provided $\alpha+\beta$ is in Cantor normal form (and justifies the use of superscripts):

$$h^\alpha \circ h^\beta(x) = h^{\alpha+\beta}(x) \ , \tag{16}$$

and if $h$ is monotone inflationary, then so is $h^\alpha$:

$$\text{if } x \leq y, \text{ then } x \leq h^\alpha(x) \leq h^\alpha(y) \ . \tag{17}$$

Regarding the Cichoń functions, an easy induction on $\alpha$ shows that $H^\alpha(x) = H_\alpha(x) + x$ for the hierarchy relative to $H(x) \stackrel{\text{def}}{=} x+1$. But the main interest of Cichoń functions is that they capture how many iterations are performed by Hardy functions [8]:

$$h^{h_\alpha(x)}(x) = h^\alpha(x) \ . \tag{18}$$

*3) Length Function Theorem:* We can now state a 'length function theorem' for controlled descending sequences of ordinals.

**Theorem 6** ([28, Thm. 3.3])**.** *Let $N_0 \geq n+1$. The maximal length of $(N_0, h)$-controlled descending sequences of ordinals in $\omega^{n+1}$ is $h_{\omega^{n+1}}(N_0)$.*

### C. Controlling the Candidate Computation

*1) General Approach:* Consider an execution of CANDIDATEBOUND$_n$ entering the main loop at line 12 and let us define

$$N \stackrel{\text{def}}{=} \max\{n+1, \mathcal{E}_\mathcal{B}, \max_{0\leq i\leq n} s_i, \max_{0\leq i\leq n} |\mathcal{P}_i|\} \ . \tag{19}$$

We are going to exhibit $h\colon \mathbb{N} \to \mathbb{N}$ monotone and inflationary such that, along any execution of CANDIDATEBOUND$_n$, the sequence of successive values $N_0, N_1, \ldots$ defined by (19) each time the execution enters the main loop on line 12 satisfies

$$N_\ell \leq h^\ell(N_0) \tag{20}$$

for all $\ell$ in $\mathbb{N}$. By definition of the ordinal size in (14) of the ranks from (12), $\|\alpha_\ell\| \leq N_\ell$. Hence, this will show that the corresponding sequence of ranks $\alpha_0 > \alpha_1 > \cdots$ is $(N_0, h)$-controlled. Therefore, Theorem 6 can be applied since furthermore $N_0 \geq n+1$, showing that the number of loop iterations is bounded by

$$L \stackrel{\text{def}}{=} h_{\omega^{n+1}}(N_0) \ . \tag{21}$$

By (18), this will entail an upper bound on the returned $\mathcal{E}_\mathcal{B}$ when CANDIDATEBOUND$_n$ terminates:

$$\mathcal{E}_\mathcal{B} \leq N_L \leq h^L(N_0) = h^{\omega^{n+1}}(N_0) \ . \tag{22}$$

*2) Controlling one Loop Execution:* As a preliminary, let us observe that, for all $0 \leq i \leq n$, the number of elements of PAIRS$_i$ (defined in (11)) can be bounded by

$$|\text{PAIRS}_i| \leq \big((|\mathcal{N}|+i)\cdot s_i^m\big)^{s_i} \cdot s_i^2 \leq 2^{3 s_i |\mathcal{G}| \log n \log s_i} \ . \tag{23}$$

Indeed, the graph representation of some pair $(E, F)$ in PAIRS$_i$ has at most $s_i$ vertices, each labelled by a nonterminal symbol from $\mathcal{N}$ or a variable from $\{x_1, \ldots, x_i\}$ and with at most $m$ outgoing edges; finally the two roots must be distinguished.

Let us turn our attention to the contents of the main loop.

**Lemma 7.** *For all $\ell$ in $\mathbb{N}$ we have $N_{\ell+1} \leq G_\mathcal{G}(N_\ell)$ where*

$$G_\mathcal{G}(x) \stackrel{\text{def}}{=} 2^{2^{2n+6} c^2 g^2 |\mathcal{G}|^3 x^4} \ .$$

*Proof.* Assume we enter the main loop for the $\ell$th time with $N_\ell$ as defined in (19). On line 13, a new equivalence level $e$ is introduced, with $e \leq 2cN_\ell^2$ since $\mathcal{E}_\mathcal{B} \leq N_\ell$ and $\text{SIZE}(E,F) \leq N_\ell$, thus in case of an update on line 17, we have $e_i \leq 2cN_\ell^2$. Consider now the **for** loop on lines 18–21. Regarding line 20, observe that $\max_{(E,F)\in\mathcal{B}} \text{EL}(E,F) \leq \max\{e, \mathcal{E}_\mathcal{B}\} \leq 2cN_\ell^2$, thus

$$e_j \leq 2cN_\ell^2 \tag{24}$$

for all $j$ in $\{i, \ldots, 0\}$ and $s_i \leq N_\ell$ by assumption. Thus, regarding line 19, for all $j$ in $\{i-1, \ldots, 0\}$,

$$\begin{aligned} s_j &\leq 2^{i-j} N_\ell + (2^{i-j}-1)(g + 2cN_\ell^2(\text{SINC}+g)) \\ &\leq 2^{n+2} cg |\mathcal{G}| N_\ell^2 \ . \end{aligned} \tag{25}$$

Regarding line 21, by (23), (25) entails that for all $j$ in $\{i-1, \ldots, 0\}$,

$$|\mathcal{P}_j| \leq 2^{2^{2n+6} c^2 g^2 |\mathcal{G}|^3 N_\ell^4} \ . \tag{26}$$

Finally, regarding line 22, by (24), $\mathcal{E}_\mathcal{B} \leq 2(n+1)cN_\ell^2$. □

*3) Final Bound:* Let us finally express (22) in terms of $n$ and $|\mathcal{G}|$. First observe that, at the end of the initialisation phase of lines 2–8, $e_i = 0$, $s_i \leq 2^{n+1} g$, $|\mathcal{P}_i| \leq 2^{2^{2n+5} s^2 g^2 \log |\mathcal{G}|}$, and $\mathcal{E}_\mathcal{B} = n+1$, thus

$$N_0 \leq 2^{2^{2n+5} s^2 g^2 \log |\mathcal{G}|} \ . \tag{27}$$

Then, because the bounds in Lemma 7 and Eq. (27) are in terms of $|\mathcal{G}|$ (recall that the grammatical constant $g$ is exponential and $n$, $s$, and $c$ are doubly exponential in terms of $|\mathcal{G}|$), there exists a constant $d$ independent from $\mathcal{G}$ such that $|\mathcal{G}| \leq N_0 \leq H^{\omega^2\cdot d}(|\mathcal{G}|)$ and $G_\mathcal{G}(x) \leq H^{\omega^2\cdot d}(\max\{x, |\mathcal{G}|\})$ for all $\mathcal{G}$ and $x$, where according to (16) $H^{\omega^2\cdot d}$ is the $d$th iterate of $H^{\omega^2}(x) = 2^{x+1}(x+1)-1$. Then by (17), $h(x) \stackrel{\text{def}}{=} H^{\omega^2\cdot d}(x)$ is a suitable control function that satisfies (20) and therefore (22).

Finally, because $n \leq N_0 \leq h(|\mathcal{G}|)$ and by (17), $h^{\omega^{n+1}}(N_0) \leq h^{\omega^\omega}(h(|\mathcal{G}|))$. We have just shown the following upper bound.

**Lemma 8.** *Let $\mathcal{G}$ be a first-order grammar and $n$, $s$, and $g$ be defined as in Theorem 3. Then $\mathcal{E}_{\mathcal{B}_{n,s,g}} \leq h^{\omega^\omega}(h(|\mathcal{G}|))$ where $h(x) \stackrel{\text{def}}{=} H^{\omega^2\cdot d}(x)$ for some constant $d$.*

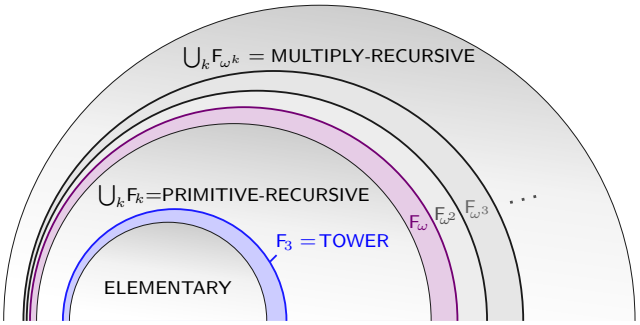


Fig. 2. Pinpointing $\mathsf{F}_\omega = \mathsf{ACKERMANN}$ among the complexity classes beyond ELEMENTARY [29].

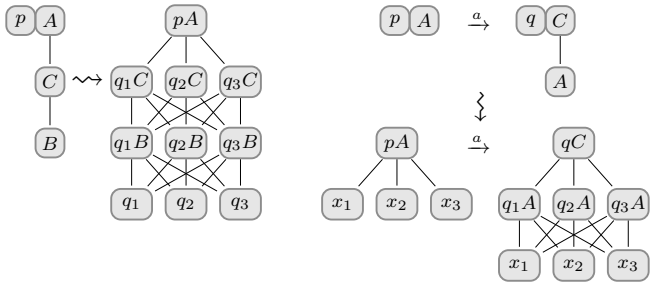


Fig. 3. The PDS configuration $pACB$ encoded as a term (left), and the translation of the PDS rule $pA \xrightarrow{a} qCA$ into a first-order rule (right).

### D. Fast-Growing Complexity

It remains to combine Fact 1 with Lemma 8 in order to provide an upper bound for the bisimilarity problem. We will employ for this the *fast-growing* complexity classes defined in [29]. This is an ordinal-indexed hierarchy of complexity classes $(\mathsf{F}_\alpha)_{\alpha<\varepsilon_0}$, that uses the Hardy functions $(H^\alpha)_\alpha$ relative to $H(x) \stackrel{\text{def}}{=} x+1$ as a standard against which we can measure high complexities.

*1) Fast-Growing Complexity Classes:* Let us first define

$$\mathscr{F}_{<\alpha} \stackrel{\text{def}}{=} \bigcup_{\beta<\omega^\alpha} \mathsf{FDTIME}\big(H^\beta(n)\big) \tag{28}$$

as the class of functions computed by deterministic Turing machines in time $O(H^\beta(n))$ for some $\beta < \omega^\alpha$. This captures for instance the class of Kalmar elementary functions as $\mathscr{F}_{<3}$ and the class of primitive-recursive functions as $\mathscr{F}_{<\omega}$ [23, 38]. Then we let

$$\mathsf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{p\in\mathscr{F}_{<\alpha}} \mathsf{DTIME}\big(H^{\omega^\alpha}(p(n))\big) \tag{29}$$

denote the class of decision problems solved by deterministic Turing machines in time $O\big(H^{\omega^\alpha}(p(n))\big)$ for some function $p \in \mathscr{F}_{<\alpha}$. The intuition behind this quantification over $p$ is that, just like e.g. $\mathsf{EXPTIME} = \bigcup_{p\in\mathsf{poly}} \mathsf{DTIME}\big(2^{p(n)}\big)$ quantifies over polynomial functions to provide enough 'wiggle room' to account for polynomial reductions, $\mathsf{F}_\alpha$ is closed under $\mathscr{F}_{<\alpha}$ reductions [29, Thms. 4.7 and 4.8].

For instance, $\mathsf{TOWER} \stackrel{\text{def}}{=} \mathsf{F}_3$ defines the class of problems that can be solved using computational resources bounded by a tower of exponentials of elementary height in the size of the input, $\bigcup_{k\in\mathbb{N}} \mathsf{F}_k$ is the class of primitive-recursive decision problems, and $\mathsf{ACKERMANN} \stackrel{\text{def}}{=} \mathsf{F}_\omega$ is the class of problems that can be solved using computational resources bounded by the Ackermann function applied to some primitive-recursive function of the input size—here it does not matter for $\alpha > 2$ whether we are considering deterministic, nondeterministic, alternating, time, or space bounds [29, Sec. 4.2.1]. See Figure 2 for a depiction.

**Theorem 9.** *The bisimulation problem for first-order grammars is in* ACKERMANN*, and in* $\mathsf{F}_{n+4}$ *if* $n$ *is fixed.*

*Proof.* This is a consequence of Fact 1 combined with Theorem 3 and Lemma 8; the various overheads on top of the bound on $\mathcal{E}_{\mathcal{B}_{n,s,g}}$ are of course negligible for such high complexities [29, Lem. 4.6]. We rely here on [29, Thm. 4.2] to translate from $h^{\omega^{n+1}}$ with $h = H^{\omega^2\cdot d} \in \mathscr{F}_{<3}$ into a bound in terms of $H^{\omega^{n+4}}$. □

## VI. Pushdown Processes

The complexity upper bounds obtained in Section V are stated in terms of first-order grammars. In this section, we revisit the known reduction from pushdown systems to first-order grammars (as given in [16, 20]), and we also give a direct reduction from first-order grammars to pushdown systems (instead of giving just a general reference to [9, 7]). We do this first to make clear that the reductions are primitive recursive (in fact, they are polynomial-time reductions), and second to show that, in the real-time case, Theorem 9 provides primitive-recursive bounds for pushdown systems with a fixed number of states.

*1) Pushdown Systems:* Let us first recall that a *pushdown system* (*PDS*) is a tuple $M = (Q, \Sigma, \Gamma, \Delta)$ of finite sets where the elements of $Q, \Sigma, \Gamma$ are called *control states*, *actions* (or *terminal letters*), and *stack symbols*, respectively; $\Delta$ contains *transition rules* of the form $pY \xrightarrow{a} q\gamma$ where $p, q \in Q$, $Y \in \Gamma$, $a \in \Sigma \uplus \{\varepsilon\}$, and $\gamma \in \Gamma^*$. A pushdown system is called *real-time* if $a$ is restricted to be in $\Sigma$, i.e., if no $\varepsilon$ transition rules appear in $\Delta$.

A PDS $M = (Q, \Sigma, \Gamma, \Delta)$ generates the labelled transition system

$$\mathcal{L}_M \stackrel{\text{def}}{=} (Q\times\Gamma^*, \Sigma\uplus\{\varepsilon\}, (\xrightarrow{a})_{a\in\Sigma\cup\{\varepsilon\}})$$

where each rule $pY \xrightarrow{a} q\gamma$ induces transitions $pY\gamma' \xrightarrow{a} q\gamma\gamma'$ for all $\gamma' \in \Gamma^*$. Note that $\mathcal{L}_M$ might feature *$\varepsilon$-transitions* (also called *$\varepsilon$-steps*) $pY\gamma' \xrightarrow{\varepsilon} q\gamma\gamma'$ if the PDS is not real-time.

### A. From PDS to First-Order Grammars

We recall a construction already presented in the appendix of the extended version of [20]. The idea is that, although first-order grammars lack the notion of control state, the behaviour of a pushdown system can nevertheless be captured by a first-order grammar that uses $m$-ary terms where $m$ is the number of control states.

Figure 3 (left) presents a configuration of a PDS—i.e., a state in $\mathcal{L}_M$—as a term; here we assume that $Q = \{q_1, q_2, q_3\}$. The string $pACB$, depicted on the left in a convenient vertical form, is translated into a term presented by an acyclic graph in the figure. On the right in Figure 3 we can see the translation of the PDS transition rule $pA \xrightarrow{a} qCA$ into a rule of a first-order grammar.

*1) Real-Time Case:* Let us first assume that $M$ is a real-time PDS, i.e., that each PDS transition rule $pY \xrightarrow{a} q\gamma$ is such that $a$ is in $\Sigma$. We are interested in the following decision problem.

**Problem** (Strong Bisimulation)**.**

input A real-time pushdown system $M = (Q, \Sigma, \Gamma, \Delta)$ and two configurations $pY, qZ$ in $Q \times \Gamma$.

question Is $pY \sim qZ$ in the labelled transition system $\mathcal{L}_M$?

Formally, for a real-time PDS $M = (Q, \Sigma, \Gamma, \Delta)$, where $Q = \{q_1, q_2, \ldots, q_m\}$, we can define the first-order grammar

$$\mathcal{G}_M \stackrel{\text{def}}{=} (\mathcal{N}, \Sigma, \mathcal{R})$$

where $\mathcal{N} \stackrel{\text{def}}{=} Q \cup (Q \times \Gamma)$, with $r(q) \stackrel{\text{def}}{=} 0$ and $r((q, X)) \stackrel{\text{def}}{=} m = |Q|$ for all $q$ in $Q$ and $X$ in $\Gamma$; the set $\mathcal{R}$ is defined below. We write $[q]$ and $[qY]$ for nonterminals $q$ and $(q, Y)$, respectively, and we map each configuration $p\gamma$ to a (finite) term $\mathcal{T}(p\gamma)$ in $\text{TERMS}_{\mathcal{N}}$ defined by structural induction:

$$\mathcal{T}(p\varepsilon) \stackrel{\text{def}}{=} [p] \,, \tag{30}$$

$$\mathcal{T}(pY\gamma) \stackrel{\text{def}}{=} [pY](\mathcal{T}(q_1\gamma), \mathcal{T}(q_2\gamma), \ldots, \mathcal{T}(q_m\gamma)) \,. \tag{31}$$

For a smooth translation of rules, we introduce a special 'stack variable' $x$, and we set

$$\mathcal{T}(q_i x) \stackrel{\text{def}}{=} x_i \tag{32}$$

for all $i \in \{1, \ldots, m\}$.

A PDS transition rule $pY \xrightarrow{a} q\gamma$ in $\Delta$ with $a$ in $\Sigma$ is then translated into the first-order grammar rule

$$\mathcal{T}(pYx) \xrightarrow{a} \mathcal{T}(q\gamma x) \tag{33}$$

in $\mathcal{R}$. Hence $pY \xrightarrow{a} q_i$ is translated into

$$[pY](x_1, \ldots, x_m) \xrightarrow{a} x_i$$

and $pY \xrightarrow{a} qZ\gamma$ is translated into

$$[pY](x_1, \ldots, x_m) \xrightarrow{a} [qZ](\mathcal{T}(q_1\gamma x), \ldots, \mathcal{T}(q_m\gamma x)) \,.$$

It should be obvious that the labelled transition system $\mathcal{L}_M$ is isomorphic with the restriction of the labelled transition system $\mathcal{L}_{\mathcal{G}_M}$ to the states $\mathcal{T}(p\gamma)$ where $p\gamma$ are configurations of $M$; moreover, the set $\{\mathcal{T}(p\gamma) \mid p \in Q, \gamma \in \Gamma^*\}$ is closed w.r.t. reachability in $\mathcal{L}_{\mathcal{G}_M}$: if $\mathcal{T}(p\gamma) \xrightarrow{a} F$ in $\mathcal{L}_{\mathcal{G}_M}$, then $F = \mathcal{T}(q\gamma')$ where $p\gamma \xrightarrow{a} q\gamma'$ in $\mathcal{L}_M$.

**Corollary 10.** *The strong bisimulation problem for real-time pushdown systems is in* ACKERMANN*, and in* $\mathsf{F}_{|Q|+4}$ *if the number* $|Q|$ *of states is fixed.*

*Proof.* What we have sketched above is a polynomial-time (in fact, logspace) reduction from the strong bisimulation problem in (real-time) pushdown systems to the bisimulation problem in first-order grammars, for which we can apply Theorem 9. Observe that, in this translation and according to the discussion after (6), we may bound $n$ by the number $|Q|$ of states of the given pushdown system, which justifies the primitive-recursive $\mathsf{F}_{|Q|+4}$ upper bound when the number of states is fixed. (Figure 3 makes clear that all branches in $\mathcal{T}(p\gamma)$ have the same lengths, and there are precisely $|Q|$ depth-$d$ subterms of $\mathcal{T}(p\gamma)$, for each $d \leq \text{HEIGHT}(\mathcal{T}(p\gamma))$.) □

*2) General Case:* In the case of labelled transition systems $\mathcal{L} = (\mathcal{S}, \Sigma, (\xrightarrow{a})_{a \in \Sigma \uplus \{\varepsilon\}})$ with a *silent action* $\varepsilon$, by $s \stackrel{w}{\Rightarrow} t$, for $w \in \Sigma^*$, we denote that there are $s_0, s_1, \ldots, s_\ell \in \mathcal{S}$ and $a_1, \ldots, a_\ell \in \Sigma \uplus \{\varepsilon\}$ such that $s_0 = s$, $s_\ell = t$, $s_{i-1} \xrightarrow{a_i} s_i$ for all $i \in \{1, \ldots, \ell\}$, and $w = a_1 \cdots a_\ell$. Thus $s \stackrel{\varepsilon}{\Rightarrow} t$ denotes an arbitrary sequence of silent steps, and $s \stackrel{a}{\Rightarrow} t$ for $a \in \Sigma$ denotes that there are $s', t'$ such that $s \stackrel{\varepsilon}{\Rightarrow} s' \xrightarrow{a} t' \stackrel{\varepsilon}{\Rightarrow} t$.

A relation $R \subseteq \mathcal{S} \times \mathcal{S}$ is a *weak bisimulation* if the following two conditions hold:

(zig) if $s \,R\, t$ and $s \xrightarrow{a} s'$ for some $a \in \Sigma \uplus \{\varepsilon\}$, then there exists $t'$ such that $t \stackrel{a}{\Rightarrow} t'$ and $s' \,R\, t'$;

(zag) if $s \,R\, t$ and $t \xrightarrow{a} t'$ for some $a \in \Sigma \uplus \{\varepsilon\}$, then there exists $s'$ such that $s \stackrel{a}{\Rightarrow} s'$ and $s' \,R\, t'$.

By $\approx$ we denote *weak bisimilarity*, i.e., the largest weak bisimulation (the union of all weak bisimulations), which is an equivalence relation.

We are now interested in the following problem.

**Problem** (Weak Bisimulation)**.**

input A pushdown system $M = (Q, \Sigma, \Gamma, \Delta)$ and two configurations $pY, qZ$ in $Q \times \Gamma^*$.

question Is $pY \approx qZ$ in the labelled transition system $\mathcal{L}_M$?

Unfortunately, in general the weak bisimulation problem for PDS is undecidable, already for one-counter systems [24]; we can also refer, e.g., to [15] for further discussion. As already mentioned in the introduction, we thus consider PDS with (very) *restricted silent actions*: each rule $pY \xrightarrow{\varepsilon} q\gamma$ in $\Delta$ is *deterministic* (i.e., alternative-free), which means that there is no other rule with the left-hand side $pY$. From now on, by *restricted PDS* we mean PDS whose $\varepsilon$-rules are deterministic.

We aim to show that the weak bisimulation problem for restricted PDS reduces to the (strong) bisimulation problem for first-order grammars (where silent actions are not allowed by our definition). For this it is convenient to make a standard transformation [see, e.g., 12, Sec. 5.6] of our restricted PDS that removes non-popping $\varepsilon$-rules; an $\varepsilon$-rule $pY \xrightarrow{\varepsilon} q\gamma$ is called *popping* if $\gamma = \varepsilon$. This is captured by the next proposition. (When comparing two states from different LTSs, we implicitly refer to the disjoint union of these LTSs.)

**Proposition 11.** *There is a polynomial-time transformation of a restricted PDS* $M = (Q, \Sigma, \Gamma, \Delta)$ *to* $M' = (Q, \Sigma, \Gamma, \Delta')$ *in which each* $\varepsilon$*-rule is deterministic and popping, and* $pY$ *in* $\mathcal{L}_M$ *is weakly bisimilar with* $pY$ *in* $\mathcal{L}_{M'}$*.*

*Proof.* Given a restricted PDS $M = (Q, \Sigma, \Gamma, \Delta)$, we proceed as follows. First we find all $pY$ such that

$$pY \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} pY\gamma \tag{34}$$

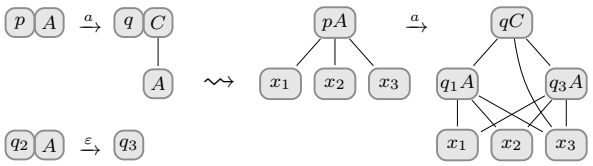

Fig. 4. Deterministic popping silent steps are 'preprocessed.'

for some $\gamma \in \Gamma^*$, and remove the respective rules $pY \xrightarrow{\varepsilon} \cdots$. Then for each $pY$ such that

$$pY \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q \;, \tag{35}$$

we add the popping rule $pY \xrightarrow{\varepsilon} q$, and for each $pY$ where

$$pY \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} qB\gamma \tag{36}$$

and each rule $qB \xrightarrow{a} q'\gamma'$ with $a \in \Sigma$ we add the rule $pY \xrightarrow{a} q'\gamma'\gamma$. Finally we remove all the non-popping $\varepsilon$-rules. Thus $M' = (Q, \Sigma, \Gamma, \Delta')$ arises. Identifying the configurations that satisfy conditions (34–36) can be performed in polynomial time through a saturation algorithm. The claim on the relation of $\mathcal{L}_M$ and $\mathcal{L}_{M'}$ is straightforward. □

A *stable configuration* is either a configuration $p\varepsilon$, or a configuration $pY\gamma$ where there is no $\varepsilon$-rule of the form $pY \xrightarrow{\varepsilon} q\gamma'$. In a restricted PDS with only popping $\varepsilon$-rules, any unstable configuration $p\gamma$ only allows to perform a finite sequence of silent popping steps until it reaches a stable configuration. It is natural to restrict our attention to the transitions $p\gamma \xrightarrow{a} q\gamma'$ with $a \in \Sigma$ between stable configurations; such transitions might encompass sequences of popping $\varepsilon$-steps.

When defining the grammar $\mathcal{G}_M$, we can avoid the explicit use of deterministic popping silent steps, by 'preprocessing' them: we apply the inductive definition of the translation operator $\mathcal{T}$ from (30–32) to stable configurations, while if $pY$ is unstable, then there is exactly one applicable rule, $pY \xrightarrow{\varepsilon} q$, and in this case we let

$$\mathcal{T}(pY\gamma) \stackrel{\text{def}}{=} \mathcal{T}(q\gamma) \;. \tag{37}$$

Figure 4 (right) shows the grammar-rule

$$\mathcal{T}(pAx) \xrightarrow{a} \mathcal{T}(qCAx)$$

(arising from the PDS-rule $pA \xrightarrow{a} qCA$), when $Q = \{q_1, q_2, q_3\}$ and there is a PDS-rule $q_2A \xrightarrow{\varepsilon} q_3$, while $q_1A$, $q_3A$ are stable.

**Corollary 12.** *The weak bisimulation problem for restricted pushdown systems (i.e., where $\varepsilon$-rules are deterministic) is in* ACKERMANN.

*Proof.* By Proposition 11 it suffices to consider a PDS $M = (Q, \Sigma, \Gamma, \Delta)$ where each $\varepsilon$-rule is deterministic and popping. Since it is clear that $pY \approx qZ$ in $\mathcal{L}_M$ iff $\mathcal{T}(pY) \sim \mathcal{T}(qZ)$ in $\mathcal{L}_{\mathcal{G}_M}$, the claim follows from Theorem 9. □

Note that, due to our preprocessing, the terms $\mathcal{T}(p\gamma)$ may have branches of varying lengths, which is why $n$ as defined in (6) might not be bounded by the number of states as in Corollary 10.

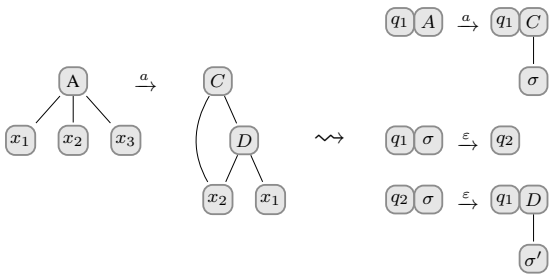

Fig. 5. The transformation from first-order grammars to pushdown processes with deterministic $\varepsilon$-steps. In this example, $x_1\sigma = x_2, x_2\sigma = D(x_2, x_1)$, and $x_1\sigma' = x_2, x_2\sigma' = x_1$.

### B. From First-Order Grammars to PDS

We have shown the ACKERMANN-membership for bisimilarity of first-order grammars (Theorem 9), and thus also for weak bisimilarity of pushdown processes with deterministic $\varepsilon$-steps (Corollary 12). By adding the lower bound from [19], we get the ACKERMANN-completeness for both problems.

In fact, the ACKERMANN-hardness in [19] was shown in the framework of first-order grammars. The case of pushdown processes was handled by a general reference to the equivalences that are known, e.g., from [9] and the works referred there; another relevant reference for such equivalences is [7]. Nevertheless, in our context it seems more appropriate to show a direct transformation from first-order grammars to pushdown processes (with deterministic $\varepsilon$-steps), which can be argued to be primitive-recursive; in fact, it is a logspace reduction.

Let $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$ be a first-order grammar. For a term $F \in \text{TERMS}_{\mathcal{N}}$ such that $F \notin \text{VAR}$ (hence the root of $F$ is a nonterminal $A$) we define its *root-substitution* to be the substitution $\sigma$ where $F = A(x_1, \ldots, x_{r(A)})\sigma$ and $x\sigma = x$ for all $x \notin \{x_1, \ldots, x_{r(A)}\}$. A substitution $\sigma$ is an *rhs-substitution* for $\mathcal{G}$ if it is the root-substitution of a subterm $F$ of the right-hand side $E$ of a rule $A(x_1, \ldots, x_{r(A)}) \xrightarrow{a} E$ in $\mathcal{R}$ (where $F \notin \text{VAR}$); we let $\text{RSUBS}_{\mathcal{G}}$ denote the set of rhs-substitutions for $\mathcal{G}$.

We define the PDS $M_{\mathcal{G}} \stackrel{\text{def}}{=} (Q, \Sigma, \Gamma, \Delta)$ where

$$Q \stackrel{\text{def}}{=} \{q_1, \ldots, q_m\}$$

for $m$ as defined in (2)—or $Q \stackrel{\text{def}}{=} \{q_1\}$ if $m = 0$—,

$$\begin{aligned}
\Gamma &\stackrel{\text{def}}{=} \mathcal{N} \uplus \text{RSUBS}_{\mathcal{G}} \;, \\
\Delta &\stackrel{\text{def}}{=} \{q_1A \xrightarrow{a} q_i \mid (A(x_1, \ldots, x_{r(A)}) \xrightarrow{a} x_i) \in \mathcal{R}\} \\
&\quad \cup \{q_1A \xrightarrow{a} q_1B\sigma \mid \sigma \in \text{RSUBS}_{\mathcal{G}} \wedge \\
&\qquad\qquad (A(x_1, \ldots, x_{r(A)}) \xrightarrow{a} B(x_1, \ldots, x_{r(B)})\sigma) \in \mathcal{R}\} \\
&\quad \cup \{q_i\sigma \xrightarrow{\varepsilon} q_j \mid 1 \le i \le m \wedge \sigma \in \text{RSUBS}_{\mathcal{G}} \wedge \sigma(x_i) = x_j\} \\
&\quad \cup \{q_i\sigma \xrightarrow{\varepsilon} q_1C\sigma' \mid 1 \le i \le m \wedge \sigma, \sigma' \in \text{RSUBS}_{\mathcal{G}} \wedge \\
&\qquad\qquad \sigma(x_i) = C(x_1, \ldots, x_{r(C)})\sigma'\} \;.
\end{aligned}$$

See Figure 5 for an example. Note that the $\varepsilon$-rules are indeed deterministic; moreover, any non-popping $\varepsilon$-step, hence of the form $q_i\sigma\gamma \xrightarrow{\varepsilon} q_1C\sigma'\gamma$, cannot be followed by another $\varepsilon$-step.

It should be obvious that a state $A(x_1, \ldots, x_{r(A)})$ in $\mathcal{L}_\mathcal{G}$ is weakly bisimilar with the state $q_1 A$ in $\mathcal{L}_{M_\mathcal{G}}$. In particular we note that $q_1 A \stackrel{w}{\Rightarrow} q_i\gamma$ in $\mathcal{L}_{M_\mathcal{G}}$ (where also $\varepsilon$-steps might be comprised) entails that $\gamma = \sigma_0\sigma_1 \ldots \sigma_\ell$ (in which case $q_i\gamma$ represents the term $x_i\sigma_0\sigma_1, \ldots \sigma_\ell$), or $\gamma = B\sigma_1 \ldots \sigma_\ell$ when $i = 1$ (in which case $q_1\gamma$ represents the term $B(x_1, \ldots, x_{r(B)})\sigma_1, \ldots \sigma_\ell$).

We could add a technical discussion about how to represent all the terms from $\text{TERMS}_\mathcal{N}$ (including the infinite regular terms) in an enhanced version of $\mathcal{L}_{M_\mathcal{G}}$, but this is not necessary since the lower bound construction in [19] uses only the states of $\mathcal{L}_\mathcal{G}$ that are reachable from 'initial' terms of the form $A(x_1, \ldots, x_{r(A)})$ (more precisely, of the form $A(\bot, \ldots, \bot)$ for a nullary nonterminal $\bot$).

**Corollary 13.** *The weak bisimulation problem for pushdown systems whose $\varepsilon$-rules are deterministic and popping is* ACKERMANN*-hard.*

*Proof.* In [19], the ACKERMANN-hardness of the control-state reachability problem for reset counter machines is recalled [27], and its polynomial-time (in fact, logspace) reduction to the bisimulation problem for first-order grammars is shown. The reduction guarantees that a given control state is reachable from the initial configuration of a given reset counter machine $R$ iff $A(\bot, \ldots, \bot) \not\sim B(\bot, \ldots, \bot)$ in $\mathcal{L}_{\mathcal{G}_R}$ for the constructed grammar $\mathcal{G}_R$. As shown above, the question whether $A(\bot, \ldots, \bot) \sim B(\bot, \ldots, \bot)$ in $\mathcal{L}_{\mathcal{G}_R}$ can be further reduced to an instance of the weak bisimulation problem for the pushdown system $M_{\mathcal{G}_R}$. □

## VII. Concluding Remarks

Theorem 9 and Corollary 12 provide the first known worst-case upper bounds, in ACKERMANN, for the strong bisimulation equivalence of first-order grammars and the weak bisimulation equivalence of pushdown processes restricted to deterministic silent steps. By the lower bound shown in [19] and Corollary 13, this is moreover optimal. An obvious remaining problem is to close the complexity gap in the case of strong bisimulation for real-time pushdown processes, which is only known to be TOWER-hard [1], and for which we do not expect Corollary 10 to provide tight upper bounds.

## Appendix

The proof of Theorem 3 in [21, Thm. 7] relies on the definition of several grammatical constants, which depend solely on the given first-order grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$. In Table II we summarise their definitions as a reference for the reader.

## Acknowledgements

P. Jančar acknowledges the support of the Grant Agency of Czech Rep., GAČR 18-11193S; part of this research was conducted while he held an invited professorship at ENS Paris-Saclay. S. Schmitz is partially funded by Institut Universitaire de France and ANR-17-CE40-0028 BRAVAS.

## References

[1] M. Benedikt, S. Göller, S. Kiefer, and A. S. Murawski, "Bisimilarity of pushdown automata is nonelementary," in *Proc. LICS'13*. IEEE, 2013, pp. 488–498.

[2] J. van Benthem, "Modal correspondence theory," Ph.D. dissertation, Mathematisch Instituut & Instituut voor Grondslagenonderzoek, University of Amsterdam, 1975.

[3] S. Böhm, S. Göller, and P. Jančar, "Bisimulation equivalence and regularity for real-time one-counter automata," *J. Comput. Syst. Sci.*, vol. 80, no. 4, pp. 720–743, 2014.

[4] C. Broadbent and S. Göller, "On bisimilarity of higher-order pushdown automata: Undecidability at order two," in *Proc. FSTTCS'12*, ser. Leibniz Int. Proc. Inf., vol. 18. LZI, 2012, pp. 160–172.

[5] O. Burkart, D. Caucal, and B. Steffen, "An elementary bisimulation decision procedure for arbitrary context-free processes," in *Proc. MFCS'95*, ser. Lect. Notes in Comput. Sci., vol. 969. Springer, 1995, pp. 423–433.

[6] D. Caucal, "Monadic theory of term rewritings," in *Proc. LICS'92*. IEEE, 1992, pp. 266–273.

[7] ——, "Bisimulation of context-free grammars and pushdown automata," in *Modal Logic and Process Algebra: A Bisimulation Perspective*, ser. CSLI Lecture Notes, A. Ponse, M. de Rijke, and Y. Venema, Eds. CSLI Publications, 1995, vol. 53, ch. 5, pp. 85–106.

[8] E. A. Cichoń and E. Tahhan Bittar, "Ordinal recursive bounds for Higman's Theorem," *Theor. Comput. Sci.*, vol. 201, no. 1–2, pp. 63–84, 1998.

[9] B. Courcelle, "Recursive applicative program schemes," in *Handbook of Theoretical Computer Science, vol. B*. MIT Press, 1990, ch. 9, pp. 459–492.

[10] W. Czerwiński and S. Lasota, "Fast equivalence-checking for normed context-free processes," in *Proc. FSTTCS'10*, ser. Leibniz Int. Proc. Inf., vol. 8. LZI, 2010, pp. 260–271.

[11] R. J. van Glabbeek, "The linear time — branching time spectrum I. The semantics of concrete, sequential processes," in *Handbook of Process Algebra*, 2001, ch. 1, pp. 3–99.

[12] M. A. Harrison, *Introduction to Formal Language Theory*. Addison-Wesley, 1978.

[13] Y. Hirshfeld, M. Jerrum, and F. Moller, "A polynomial algorithm for deciding bisimilarity of normed context-free processes," *Theor. Comput. Sci.*, vol. 158, no. 1–2, pp. 143–159, 1996.

[14] D. Janin and I. Walukiewicz, "On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic," in *Proc. Concur'96*, ser. Lect. Notes in Comput. Sci., vol. 1119. Springer, 1996, pp. 263–277.

[15] P. Jančar and J. Srba, "Undecidability of bisimilarity by Defender's forcing," *J. ACM*, vol. 55, no. 1, pp. 5:1–5:26, 2008.

[16] P. Jančar, "Decidability of DPDA language equivalence via first-order grammars," in *Proc. LICS'12*. IEEE, 2012, pp. 415–424.

[17] ——, "Bisimilarity on basic process algebra is in 2EXPTIME (an explicit proof)," *Logic. Meth. in Comput. Sci.*, vol. 9, no. 1, pp. 10:1–10:19, 2013.

[18] ——, "Bisimulation equivalence of first-order grammars," in *Proc. ICALP'14*, ser. Lect. Notes in Comput. Sci., vol. 8573. Springer, 2014, pp. 232–243.

[19] ——, "Equivalences of pushdown systems are hard," in *Proc. FoSSaCS'14*, ser. Lect. Notes in Comput. Sci., vol. 8412. Springer, 2014, pp. 1–28.

[20] ——, "Deciding semantic finiteness of pushdown processes and first-order grammars w.r.t. bisimulation equivalence," in *Proc. MFCS'16*, ser. Leibniz Int. Proc. Inf., vol. 58. LZI, 2016, pp. 52:1–52:13, full version available as arXiv:1305.0516 [cs.LO].

[21] ——, "Equivalence of pushdown automata via first-order grammars," Preprint, arXiv:1812.03518 [cs.LO], 2018, submitted to a journal.

[22] S. Kiefer, "BPA bisimilarity is EXPTIME-hard," *Inform. Proc. Letters*, vol. 113, no. 4, pp. 101–106, 2013.

[23] M. H. Löb and S. S. Wainer, "Hierarchies of number theoretic functions, I," *Arch. Math. Logic*, vol. 13, pp. 39–51, 1970.

[24] R. Mayr, "Undecidability of weak bisimulation equivalence for 1-counter processes," in *Proc. ICALP'03*, ser. Lect. Notes in Comput. Sci., vol. 2719. Springer, 2003, pp. 570–583.

[25] R. Milner, *A Calculus of Communicating Systems*, ser. Lect. Notes in Comput. Sci. Springer, 1980, vol. 92.

[26] D. M. R. Park, "Concurrency and automata on infinite sequences," in *Proc. GI TCS'81*, ser. Lect. Notes in Comput. Sci., vol. 104. Springer,

TABLE II
GRAMMATICAL CONSTANTS DEFINED IN [21]

| Constant | Ref. in [21] | Ref. in this paper | Growth in $\lvert\mathcal{G}\rvert$ |
|---|---|---|---|
| $m = \max_{A\in\mathcal{N}} r(A)$ | (7) | (2) | linear |
| $\text{HINC} = \max_{E\in\text{RHS}} \text{HEIGHT}(E) - 1$ | (4) | (3) | linear |
| $\text{SINC} = \max_{E\in\text{RHS}} \text{NTSIZE}(E)$ | (5) | (4) | linear |
| $d_0 = 1 + \max_{A\in\mathcal{N}, 1\le i\le r(A)} \lvert w_{[A,i]}\rvert$ | (6) | (5) | exponential |
| $d_1 = 2\lvert\mathcal{N}\rvert(\max\{d_0, \lvert\mathcal{R}\rvert^{d_0}\})^{m+2}$ | (13) | | doubly exponential |
| $d_2 = d_0 + (1 + d_0\text{HINC})(d_0 - 1)$ | (19) | | exponential |
| $d_3 = (\max\{d_0, \lvert\mathcal{R}\rvert^{d_0}\})^2$ | (21) | | doubly exponential |
| $n = m^{d_0}$ | (24) | (6) | doubly exponential |
| $s = m^{d_0+1} + (m+2)d_0\text{SINC} + (d_2 + d_0 - 1)\text{SINC}$ | (25) | | doubly exponential |
| $g = (d_2 + d_0 - 1)\text{SINC}$ | (26) | | exponential |
| $d_4 = d_1(1 + \sum_{E\in\text{RHS}} \text{NTSIZE}(E))^{d_2+d_0-1}$ | (23) | | doubly exponential |
| $d_5 = (d_2 + d_0 - 1)(1 + (d_0 - 1)\text{HINC})$ | (31) | | doubly exponential |
| $c = \max\{d_3, 2d_4d_5\}$ | (38) | | doubly exponential |


1981, pp. 167–183.

[27] Ph. Schnoebelen, "Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets," in *Proc. MFCS'10*, ser. Lect. Notes in Comput. Sci., vol. 6281. Springer, 2010, pp. 616–628.

[28] S. Schmitz, "Complexity bounds for ordinal-based termination," in *Proc. RP 2014*, ser. Lect. Notes in Comput. Sci., vol. 8762. Springer, 2014, pp. 1–19.

[29] ——, "Complexity hierarchies beyond ELEMENTARY," *ACM Trans. Comput. Theory*, vol. 8, no. 1, pp. 3:1–3:36, 2016.

[30] ——, "Algorithmic complexity of well-quasi-orders," Habilitation thesis, École Normale Supérieure Paris-Saclay, 2017.

[31] G. Sénizergues, "The equivalence problem for deterministic pushdown automata is decidable," in *Proc. ICALP'97*, ser. Lect. Notes in Comput. Sci., vol. 1256. Springer, 1997, pp. 671–681.

[32] ——, "Decidability of bisimulation equivalence for equational graphs of finite out-degree," in *Proc. FOCS'98*. IEEE, 1998, pp. 120–129.

[33] ——, "$L(A) = L(B)$? Decidability results from complete formal systems," *Theor. Comput. Sci.*, vol. 251, no. 1–2, pp. 1–166, 2001.

[34] ——, "The bisimulation problem for equational graphs of finite out-degree," *SIAM J. Comput.*, vol. 34, no. 5, pp. 1025–1106, 2005.

[35] J. Srba, "Roadmap of infinite results," in *Current Trends in Theoretical Computer Science*. World Scientific Publishing, 2004, vol. 2, pp. 337–350. [Online]. Available: http://people.cs.aau.dk/~srba/roadmap/

[36] ——, "Beyond language equivalence on visibly pushdown automata," *Logic. Meth. in Comput. Sci.*, vol. 5, no. 1, pp. 2:1–2:22, 2009.

[37] C. Stirling, "Deciding DPDA equivalence is primitive recursive," in *Proc. ICALP'02*, ser. Lect. Notes in Comput. Sci., vol. 2380. Springer, 2002, pp. 821–832.

[38] S. S. Wainer, "Ordinal recursion, and a refinement of the extended Grzegorczyk hierarchy," *J. Symb. Log.*, vol. 37, no. 2, pp. 281–292, 1972.

[39] Q. Yin, Y. Fu, C. He, M. Huang, and X. Tao, "Branching bisimilarity checking for PRS," in *Proc. ICALP'14*, ser. Lect. Notes in Comput. Sci., vol. 8573. Springer, 2014, pp. 363–374.