Equivalences of Pushdown Systems Are Hard

Petr Jančar*

Dept Comp. Sci., FEI, Techn. Univ. of Ostrava (VŠB-TUO),
17. Listopadu 15, 70833 Ostrava, Czech Rep.
petr.jancar@vsb.cz
http://www.cs.vsb.cz/jancar

Abstract. Language equivalence of deterministic pushdown automata (DPDA) was shown to be decidable by Sénizergues (1997, 2001); Stirling (2002) then showed that the problem is primitive recursive.

Sénizergues (1998, 2005) also generalized his proof to show decidability of bisimulation equivalence of (nondeterministic) PDA where ε-rules can be only deterministic and popping; this problem was shown to be nonelementary by Benedikt, Göller, Kiefer, and Murawski (2013), even for PDA with no ε-rules.

Here we refine Stirling's analysis and show that DPDA equivalence is in TOWER, i.e., in the "least" nonelementary complexity class. The basic proof ideas remain the same but the presentation and the analysis are simplified, in particular by using a first-order term framework.

The framework of (nondeterministic) first-order grammars, with term root-rewriting rules, is equivalent to the model of PDA with restricted ε-rules to which Sénizergues's decidability proof applies. We show that bisimulation equivalence is here Ackermann-hard, and thus not primitive recursive.

1 Introduction

Pushdown automata (PDA) are a standard and widely used model in computer science; we recall that a PDA is a (generally nondeterministic) finite automaton equipped with an (unbounded) stack, i.e., with a (LIFO) linear list accessible at only one end (at the "top"). PDA are naturally used to model (sequential) programs with recursive procedure calls, and they also form a basis for (automated) verification of some program properties. A traditional area that employs deterministic PDA (DPDA) is the syntactic analysis of programming languages.

Given such "devices" or "systems" (like PDA or DPDA), it is standard to ask if their (functional or behavioural) equivalence can be effectively, or even efficiently, checked. A classical equivalence is *language equivalence*, asking if two given systems accept the same sequences of (external) *input symbols*, or from another viewpoint, if they can perform the same sequences of *actions*.

While language equivalence of PDA was quickly recognized to be undecidable, the decidability question for DPDA had been a famous open problem since

* Supported by the Grant Agency of the Czech Rep., project GAČR:P202/11/0340.

A. Muscholl (Ed.): FOSSACS 2014, LNCS 8412, pp. 1–28, 2014.

1960s. The decidability was finally shown by Sénizergues [21], who got the Gödel Prize in 2002 for this achievement. Stirling [26] then showed that the problem is primitive recursive. We note that the known complexity lower bound for DPDA equivalence is just P-hardness (derived from P-hardness of the emptiness problem for context-free languages).

A fundamental behavioural equivalence is *bisimulation equivalence*, also called *bisimilarity*; roughly speaking, two states of a system (or of two systems) are bisimilar if performing an action a in one state can be matched by performing an action with the same name a in the other state so that the resulting states are also bisimilar. For deterministic systems this equivalence in principle coincides with language equivalence, but for nondeterministic ones it is finer.

Sénizergues [22] generalized his proof for DPDA to show decidability of bisimulation equivalence of (nondeterministic) PDA where ε-rules (internally changing the current state) can be only deterministic and stack-popping. (If we allow the option that a popping ε-rule can apply at the same time when an "external-action" rule can also apply, then bisimilarity becomes undecidable [14].) For this more general decidable problem no complexity upper bound has been shown. Regarding the lower bound, the previously known ExpTime-hardness [18] has been recently shifted: Benedikt, Göller, Kiefer, and Murawski [2] showed that the problem is nonelementary, even for "real-time" PDA (RT-PDA), i.e. PDA with no ε-rules.

Contribution of this paper is summarized in the following points 1, 2.

1. Equivalence of DPDA is in TOWER. We refine Stirling's analysis (from [26]) and show that DPDA equivalence is in TOWER, i.e., in the least (reasonably defined) nonelementary complexity class. We note that TOWER $= \mathbf{F}_3$ in the hierarchy of fast-growing complexity classes recently described by Schmitz [19]. The basic proof ideas remain the same (as in [26]) but the presentation and the analysis are simplified, in particular by using a first-order term framework, called *deterministic first-order grammars* (detFOG).

2. Bisimilarity of first-order grammars is Ackermann-hard. The general framework of (nondeterministic) first-order grammars (FOG), i.e. of finite sets of term *root-rewriting* rules, is equivalent to PDA with deterministic and popping ε-rules, i.e. to the model where decidability of bisimilarity was shown by Sénizergues [22]. We show that bisimulation equivalence is here even Ackermann-hard, and thus not primitive recursive. The proof is given by a reduction from the control-state reachability problem for reset (or lossy) counter machines for which Ackermann-hardness was shown by Schnoebelen (see [20] and references therein, and [28] for an independent proof related to relevance logic).

Further Comments. Some advantages of using first-order terms, i.e. a formalism with which (not only) every computer scientist is intimately familiar, were already demonstrated in [15]. Though the close relationship between (D)PDA and first-order schemes has been long known (see, e.g., [7]), the perspective viewing (D)PDA as (deterministic) finite sets of term *root-rewriting* rules seems not to have been fully exploited so far. We note that the decidability proof in [15] (for deterministic grammars) was constructed so that it constitutes a

good basis for extending the decidability proof to the nondeterministic case (to match [22]). Here we concentrate on bounding the length of shortest words witnessing nonequivalence, which cannot be easily generalized, as is now also indicated by the gap between TOWER and the Ackermann-hardness.

We summarize the relevant known results in the following table.

	DPDA = detFOG	RT-PDA	PDA∼FOG	PDA
Lang-Eq	P-hard **in TOWER**	Undecidable	Undecidable	Undecidable
Bisi-Eq	P-hard **in TOWER**	TOWER-hard Decidable	**ACK-hard** Decidable	Undecidable

In the column PDA∼FOG we refer to those PDA that are equivalent to first-order grammars, i.e., to PDA with only deterministic and popping ε-rules. PDA in the last column can have unrestricted ε-rules.

The results in the boldface are shown in this paper. The TOWER-membership is only one result, since language equivalence and bisimilarity in principle coincide for DPDA. As already mentioned, this result is derived by a finer look at Stirling's approach [26], where only a primitive recursive upper bound is claimed. The TOWER-hardness for RT-PDA is, in fact, a slight extrapolation of the result presented in [2]. The authors only claim nonelementary complexity but their proof can be adjusted to show TOWER-hardness in the sense of [19]. (This is the usual case with proofs showing nonelementary complexity lower bounds, as is also explained in [19]; in the particular case of RT-PDA this was also confirmed by a personal communication with S. Kiefer.)

We also note that the proof of Ackermann-hardness for first-order grammars presented here uses the feature (namely varying lengths of branches of syntactic trees of terms) corresponding to (restricted) ε-rules; hence TOWER-hardness remains the best known lower complexity bound for bisimilarity of RT-PDA.

The problem for reset counter machines, used in the hardness proof here, is in fact Ackermann-complete (or ACK-complete, or $\mathbf{F}_\omega$-complete in the hierarchy of [19]); the upper bound was shown in [9]. The question of a similar upper bound for the bisimilarity problem is not discussed here.

Related Work and Some Open Questions. Here we only briefly mention some results and questions that are very close to the above discussed equivalence problems, with no attempt to give any sort of a full account. (For an updated survey of a specific area, namely bisimilarity checking of infinite-state systems, we can refer to [24].)

The main challenge is still to clarify the complexity status of DPDA equivalence, which is still far from being understood. The practical experiments by Henry and Sénizergues [12] have strengthened the feeling that the TOWER bound is indeed too large. More pleasant upper bounds were shown for subclasses of DPDA. A co-NP upper bound is known for finite-turn DPDA [23]. For simple grammars (real-time DPDA with a single control state), a polynomial algorithm deciding equivalence was shown in [13] (see [8] for a recent upper bound); it is worth to note that the *language inclusion* problem is *undecidable*

even in this simple case [10]. A recent result also shows NL-completeness of equivalence of deterministic one-counter automata [3] (answering the forty-year old polynomiality question posed by Valiant and Paterson).

A natural subclass of PDA are visibly pushdown automata, with ExpTime-complete language equivalence problem (Alur and Madhusudan [1]); for bisimilarity the ExpTime-completeness was shown by Srba [25], who also used Walukiewicz's result on model checking pushdown systems [29]. For real-time one-counter automata bisimilarity is PSPACE-complete [4]. An interesting subclass is BPA (Basic Process Algebra), corresponding to real-time PDA with a single control state. In the so called normed case bisimilarity is polynomial (see the above mentioned [13], [8]), but in general bisimilarity is in 2-ExpTime (claimed in [6] and explicitly proven in [16]) and ExpTime-hard [17].

We can also mention the higher-order case. The decidability question for higher-order DPDA equivalence remains an open problem; some progress in this direction was made by Stirling in [27]. Bisimilarity of second-order RT-PDA is undecidable [5]. Another generalization of pushdown systems are ground term (or tree) rewrite systems (where rules replace subterms with subterms). Here the decidability of bisimilarity is open [11].

2 Preliminaries

In this section we define the basic notions; some standard definitions might be given in restricted forms when we do not need the full generality.

By $\mathbb{N}$ we denote the set $\{0, 1, 2, \dots\}$ of nonnegative integers; we put $[i, j] = \{i, i+1, \dots, j\}$. For a set $\mathcal{A}$, by $\mathrm{CARD}(\mathcal{A})$ we denote its cardinality (i.e. the number of elements when $\mathcal{A}$ is finite). By $\mathcal{A}^*$ we denote the set of finite sequences of elements of $\mathcal{A}$, which are also called *words* (over $\mathcal{A}$). By $|w|$ we denote the *length* of $w \in \mathcal{A}^*$. If $w = uv$ then u is a *prefix* of w, and v is a *suffix* of w. By ε we denote the *empty sequence* ($|\varepsilon| = 0$).

2.1 Bisimulation Equivalence in LTSs and in Deterministic LTSs

Labelled Transition Systems. A *labelled transition system* (an LTS) is a tuple $\mathcal{L} = (\mathcal{S}, \Sigma, (\stackrel{a}{\longrightarrow})_{a \in \Sigma})$ where $\mathcal{S}$ is a finite or countable set of *states*, Σ is a finite set of *actions* (or *letters*), and $\stackrel{a}{\longrightarrow} \subseteq \mathcal{S} \times \mathcal{S}$ is a set of *a-transitions* (for each $a \in \Sigma$). We write $s \stackrel{a}{\longrightarrow} s'$ instead of $(s, s') \in \stackrel{a}{\longrightarrow}$. By $s \stackrel{w}{\longrightarrow} s'$, where $w = a_1 a_2 \dots a_n \in \Sigma^*$, we denote that there is a *path* $s = s_0 \stackrel{a_1}{\longrightarrow} s_1 \stackrel{a_2}{\longrightarrow} \cdots \stackrel{a_n}{\longrightarrow} s_n = s'$; if $s \stackrel{w}{\longrightarrow} s'$, then s' is *reachable from* s. By writing $s \stackrel{w}{\longrightarrow}$ we mean that s *enables* w, i.e., $s \stackrel{w}{\longrightarrow} s'$ for some s'.

Deterministic LTSs. An *LTS* $\mathcal{L} = (\mathcal{S}, \Sigma, (\stackrel{a}{\longrightarrow})_{a \in \Sigma})$ is *deterministic*, a *det-LTS* for short, if for each pair $s \in \mathcal{S}$, $a \in \Sigma$ there is at most one s' such that $s \stackrel{a}{\longrightarrow} s'$. Hence if w is enabled by s then there is precisely one s' such that $s \stackrel{w}{\longrightarrow} s'$. Here we also use expressions like "the path $s \stackrel{w}{\longrightarrow} s'$" or "the path $s \stackrel{w}{\longrightarrow}$" since the respective path is unique.

Bisimilarity. We assume a (general) LTS $\mathcal{L} = (\mathcal{S}, \Sigma, (\xrightarrow{a})_{a\in\Sigma})$. A set $\mathcal{B} \subseteq \mathcal{S}\times\mathcal{S}$ is a *bisimulation* if for any $(s,t) \in \mathcal{B}$ we have: for any $s \xrightarrow{a} s'$ there is $t \xrightarrow{a} t'$ such that $(s',t') \in \mathcal{B}$, and for any $t \xrightarrow{a} t'$ there is $s \xrightarrow{a} s'$ such that $(s',t') \in \mathcal{B}$. States $s,t \in \mathcal{S}$ are *bisimulation equivalent*, or *bisimilar*, written $s \sim t$, if there is a bisimulation $\mathcal{B}$ containing (s,t). In fact, $\sim \subseteq \mathcal{S} \times \mathcal{S}$ is the maximal bisimulation, the union of all bisimulations.

Trace Equivalence and Eq-levels in Det-LTSs. It is easy to check that in det-LTSs bisimulation equivalence coincides with so called *trace equivalence*, i.e., in any *deterministic* LTS $\mathcal{L} = (\mathcal{S}, \Sigma, (\xrightarrow{a})_{a\in\Sigma})$ we have

$$s \sim t \text{ iff } \forall w \in \Sigma^* : s \xrightarrow{w} \Leftrightarrow t \xrightarrow{w}.$$

Hence s, t are equivalent iff they enable the same set of words, also called traces.

This suggests the following natural stratification of $\sim$; it can be defined in the general (nondeterministic) case as well, but the (technically easier) deterministic case is sufficient for us. We thus assume a given det-LTS $\mathcal{L} = (\mathcal{S}, \Sigma, (\xrightarrow{a})_{a\in\Sigma})$.

For any $k \in \mathbb{N}$ we put $\Sigma^{\leq k} = \{w \,;\, |w| \leq k\}$, and

$$s \sim_k t \text{ if } \forall w \in \Sigma^{\leq k} : s \xrightarrow{w} \Leftrightarrow t \xrightarrow{w}.$$

We note that $\sim_0 = \mathcal{S} \times \mathcal{S}$, and $\sim_0 \supseteq \sim_1 \supseteq \sim_2 \supseteq \cdots$. Since our assumed $\mathcal{L}$ is deterministic, it is also obvious that $\bigcap_{k\in\mathbb{N}} \sim_k = \sim$.

We use the notation $s \sim_k t$ also for $k = \omega$, identifying $s \sim_\omega t$ with $s \sim t$. For each pair (s,t) of states we define its *equivalence level*, its *eq-level* for short:

$$\textsc{EqLv}(s,t) = \max\{\, k \in \mathbb{N} \cup \{\omega\} \mid s \sim_k t \,\}.$$

We stipulate that $k < \omega$ and $\omega - k = \omega + k = \omega$ for any $k \in \mathbb{N}$. (Hence, e.g., $s \sim_{e-5} t$ means $s \sim t$ when $e = \omega$.)

Witnesses for Nonequivalent Pairs in Det-LTSs. Given a det-LTS $\mathcal{L} = (\mathcal{S}, \Sigma, (\xrightarrow{a})_{a\in\Sigma})$, we note that $s \not\sim t$ implies that any *shortest* word wa ($a \in \Sigma$) witnessing their nonequivalence (i.e., enabled by precisely one of s,t) satisfies $|w| = \textsc{EqLv}(s,t)$. For technical convenience we introduce the following definition:

$$\text{a word } w \in \Sigma^* \text{ is a } \textit{witness for } (s,t)$$

if it is a shortest word such that for some $a \in \Sigma$ we have that wa is enabled by precisely one of s,t. (A witness w for (s,t) is thus a shortest word satisfying $s \xrightarrow{w} s'$, $t \xrightarrow{w} t'$ where $\textsc{EqLv}(s',t') = 0$, i.e. $s' \not\sim_1 t'$.) The *witness set* for (s,t) is the set of all witnesses for (s,t) (which is empty iff $s \sim t$).

We state some obvious facts in the next proposition. The crux is that by performing the same action $a \in \Sigma$ from s and t in a det-LTS the eq-level drops by at most one (if at all), and it does drop for some action when $\omega > \textsc{EqLv}(s,t) > 0$.

Proposition 1. *In any det-LTS* $\mathcal{L} = (\mathcal{S}, \Sigma, (\xrightarrow{a})_{a\in\Sigma})$ *we have:*

1. *If* $\textsc{EqLv}(s,s') > \textsc{EqLv}(s,t)$, *then* $\textsc{EqLv}(s,t) = \textsc{EqLv}(s',t)$ *and the witness sets for* (s,t) *and* (s',t) *are the same.*

2. *If u is a witness for (s,t) and $u = wu'$, then u' is a witness for (s',t') where $s \xrightarrow{w} s'$, $t \xrightarrow{w} t'$. Hence also* $\text{EQLV}(s',t') = \text{EQLV}(s,t) - |w|$.
3. *If $s \xrightarrow{v} s''$ and $t \xrightarrow{v} t''$, then* $\text{EQLV}(s'',t'') \geq \text{EQLV}(s,t) - |v|$.

2.2 First-Order Terms, FO Grammars, and Det-FO Grammars

We aim to look at LTSs in which states are first-order terms. Transitions in such an LTS will be determined by a finite set of *root-rewriting* rules. We start with definitions of these ingredients.

First-Order (Regular) Terms. The terms are built from some specified function symbols, using *variables* from a fixed set $\text{VAR} = \{x_1, x_2, x_3, \dots\}$.

A *finite term* is either a variable x_i, or $A(G_1, \dots, G_m)$ where A is a function symbol with arity m and G_j are finite terms. Each finite term has its (rooted, finite, ordered) *syntactic tree*: for x_i it is just the root labelled with x_i; for $A(G_1, \dots, G_m)$, the root is labelled with A, and the ordered root-successors are the trees corresponding to $G_1, \dots, G_m$, respectively. The height of a finite term E, denoted $\text{HEIGHT}(E)$, is the length of the longest branch of its syntactic tree.

Given a syntactic tree of a term, if we allow redirecting the arcs (i.e. changing their target nodes) arbitrarily, then we get a finite *graph presentation* (with a designated root but with possible cycles) of a *regular term*; its syntactic tree might be infinite, but the term has only finitely many *subterms*, where a subterm can have infinitely many *occurrences*, in arbitrarily large *depths*. By $size(E)$ we mean the size (the number of nodes, say) of the smallest graph presentation of a regular term E. (At the level of our later analysis, the details of such definitions are unimportant.)

In what follows, by a "term" we mean a "regular term" if we do not say explicitly that the term is finite. We reserve symbols E, F, G, H, and also T, U, V, W, for denoting (regular) terms.

Substitutions, and Their Associative Composition. By $\text{TERMS}_{\mathcal{N}}$ we denote the set of all (regular) terms over a (finite) set $\mathcal{N}$ of function symbols (called "nonterminals" later). A *substitution* σ is a mapping $\sigma : \text{VAR} \to \text{TERMS}_{\mathcal{N}}$ whose *support* $\text{SUPP}(\sigma) = \{x_i \mid \sigma(x_i) \neq x_i\}$ is *finite*; we reserve the symbol σ for substitutions. By $\text{RANGE}(\sigma)$ we mean the set $\{\sigma(x_i) \mid x_i \in \text{SUPP}(\sigma)\}$. The finite-support restriction allows us to present any σ as a finite set of pairs. E.g., $\{(x_i, H)\}$ where $H \neq x_i$ is a substitution with the one-element support $\{x_i\}$.

By *applying a substitution* σ to a term E we get the term $E\sigma$ that arises from E by replacing each occurrence of x_i with $\sigma(x_i)$; given graph presentations, in the graph of E we just redirect each arc leading to x_i towards the root of $\sigma(x_i)$ (which includes the special "root-designating arc" when $E = x_i$). For $E = x_i$ we thus have $E\sigma = x_i\sigma = \sigma(x_i)$.

The natural *composition of substitutions* (where $\sigma = \sigma_1\sigma_2$ satisfies $\sigma(x_i) = (\sigma_1(x_i))\sigma_2$) is obviously associative. We thus write simply $E\sigma_1\sigma_2$ when meaning $(E\sigma_1)\sigma_2$ or $E(\sigma_1\sigma_2)$. For future use we might note that $\{(x_i, H)\}\sigma$ is the substitution arising from σ by replacing $\sigma(x_i)$ with $H\sigma$.

First-Order Grammars, and Det-FO Grammars. A *first-order grammar*, an *FO grammar* or just a *grammar* for short, is a tuple $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$ where $\mathcal{N} = \{A_1, A_2, \dots\}$ is a finite set of ranked *nonterminals*, viewed as function symbols with arities, $\Sigma = \{a_1, a_2, \dots\}$ is a finite set of *actions* (or letters), and $\mathcal{R}$ is a finite set of (root rewriting) *rules* of the form

$$A(x_1, x_2, \dots, x_m) \xrightarrow{a} E \tag{1}$$

where $A \in \mathcal{N}$, $arity(A) = m$, $a \in \Sigma$, and E is a *finite* term over $\mathcal{N}$ in which each occurring variable is from the set $\{x_1, x_2, \dots, x_m\}$. (We exemplify the rules by $A(x_1, x_2, x_3) \xrightarrow{b} C(D(x_3, B), x_2)$, $A(x_1, x_2, x_3) \xrightarrow{b} x_2$, $D(x_1, x_2) \xrightarrow{a} A(D(x_2, x_2), x_1, B)$; here the arities of A, B, C, D are $3, 0, 2, 2$, respectively.)

A grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$ is a *det-FO grammar* (deterministic first-order grammar) if for each pair $A \in \mathcal{N}$, $a \in \Sigma$ there is at most one rule of the type (1).

2.3 LTSs of Grammars, Equivalence Problem, Relation to PDA

LTSs Associated with Grammars and Det-FO Grammars. A grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$ defines the LTS $\mathcal{L}_\mathcal{G} = (\textsc{Terms}_\mathcal{N}, \Sigma, (\xrightarrow{a})_{a \in \Sigma})$ in which each rule

$$A(x_1, \dots, x_m) \xrightarrow{a} E \text{ induces transitions } (A(x_1, \dots, x_m))\sigma \xrightarrow{a} E\sigma$$

for all substitutions $\sigma : \textsc{Var} \to \textsc{Terms}_\mathcal{N}$.
(Examples of transitions induced by the previously given rules are $A(x_1, x_2, x_3) \xrightarrow{b} C(D(x_3, B), x_2)$, $A(x_5, x_5, x_2) \xrightarrow{b} C(D(x_2, B), x_5)$, $A(U_1, U_2, U_3) \xrightarrow{b} C(D(U_3, B), U_2)$, $A(U_1, U_2, U_3) \xrightarrow{b} U_2$, etc.)

We *complete the definition of* $\mathcal{L}_\mathcal{G}$ *by stipulating* that

$$\textsc{EqLv}(x_i, H) = 0 \text{ if } H \neq x_i \text{ (in particular, } x_i \not\sim_1 x_j \text{ for } i \neq j).$$

To stay in the realm of pure LTSs, we could imagine that each used variable $x \in \textsc{Var}$ is equipped with a fresh unique action a_x and with the rule $x \xrightarrow{a_x} x$. But we never consider such "transitions" in our reasoning, and we handle x_i as "dead terms" (not enabling any action). We thus (often tacitly) use the fact that $F \xrightarrow{w} G$ implies $F\sigma \xrightarrow{w} G\sigma$ (though not vice-versa in general).

Since the rhs (right-hand sides) in the rules (1) are finite terms, all *terms reachable from a finite term* are *finite*. (It turns out technically convenient to have the rhs finite while taking the set $\textsc{Terms}_\mathcal{N}$ of all regular terms as the state set of $\mathcal{L}_\mathcal{G}$; the other options are in principle equivalent.)

We also observe that *the LTS* $\mathcal{L}_\mathcal{G}$ *is deterministic iff* $\mathcal{G}$ *is a det-FO grammar.*

Equivalence Problems. By *the bisimilarity problem* (or the bisimulation equivalence problem) *for FO grammars* we mean the decision problem that asks, given a grammar $\mathcal{G}$ and terms T_0, U_0, whether $T_0 \sim U_0$ in $\mathcal{L}_\mathcal{G}$.

By *the equivalence problem for det-FO grammars* we mean the restriction of the bisimilarity problem to deterministic first-order grammars.

Relation of Grammars and Pushdown Automata. We have mentioned the relationship between (D)PDA and first-order schemes (see, e.g., [7]). A concrete transformation of a PDA (or of a DPDA) to an FO grammar (or to a det-FO grammar) can be found in [15]. We just sketch the idea, though it is not important here, and can be skipped; it suffices just to accept the main message mentioned afterwards.

A configuration $q_iY_1Y_2\dots Y_k\bot$ of a PDA, where $\bot$ is the bottom-of-the-stack symbol and the control states are $q_1, q_2, \dots, q_m$, can be viewed as the term $\mathcal{T}(q_iY_1Y_2\dots Y_k\bot)$ defined inductively as follows: $\mathcal{T}(q_i\bot) = \bot$, $\mathcal{T}(q_iY\alpha) = [q_iY](\mathcal{T}(q_1\alpha), \mathcal{T}(q_2\alpha), \dots, \mathcal{T}(q_m\alpha))$. Hence we view each pair (q_i, Y) of a control state and a stack symbol as a nonterminal $[q_iY]$ with arity m; a special case is $\bot$ with arity 0. A pushdown rule $q_iY \stackrel{a}{\longrightarrow} q_j\beta$ is rewritten to $q_iYx \stackrel{a}{\longrightarrow} q_j\beta x$ for a special formal symbol x, and the rule is transformed to $\mathcal{T}(q_iYx) \stackrel{a}{\longrightarrow} \mathcal{T}(q_j\beta x)$, where we define $\mathcal{T}(q_jx) = x_j$; hence $\mathcal{T}(q_iYx) = [q_iY](x_1, x_2, \dots, x_m)$. In fact, we still modify the operator $\mathcal{T}$ when *deterministic popping ε-rules* $q_iY \stackrel{\varepsilon}{\longrightarrow} q_j$ are present. (In this case no other rule of the form $q_iY \stackrel{\cdots}{\longrightarrow} ..$ can be present.) For each such rule we do not create the grammar rule $\mathcal{T}(q_iYx) \stackrel{\varepsilon}{\longrightarrow} \mathcal{T}(q_jx)$ (recall that we have no ε-rules in our grammars) but we put $\mathcal{T}(q_iY\alpha) = \mathcal{T}(q_j\alpha)$. (Hence the branches of the syntactic tree of $\mathcal{T}(q\alpha)$ can have varying lengths.)

The main message is that the classical language equivalence of DPDA is easily inter-reducible with the equivalence of det-FO grammars. (This also uses the fact that trace equivalence is a version of language equivalence to which the classical accepting-state equivalence can be easily reduced. Another standard fact is that in DPDA we can w.l.o.g. assume that all ε-rules are deterministic and popping.) A similar inter-reducibility holds for the bisimilarity problem for PDA and FO grammars, with the proviso that we restrict ourselves to (nondeterministic) PDA where all ε-rules (if any are present) are deterministic and popping.

2.4 Complexity Classes TOWER and ACK (Ackermann)

We now recall the notions needed for stating our complexity results. A hierarchy of "hard" complexity classes (where TOWER $= \mathbf{F}_3$ and ACK $= \mathbf{F}_\omega$) as well as more details can be found in [19].

An *elementary function* $\mathbb{N}^k \to \mathbb{N}$ arises by a finite composition of constants, the elementary operations $+, -, \cdot, div$ and the exponential operator $\uparrow$, where $m \uparrow n = m^n$. E.g., the triple-exponential function $f(n) = 2^{2^{2^n}}$ is elementary.

Tower-Bounded Functions, Class TOWER. Function $\mathsf{Tower} : \mathbb{N} \to \mathbb{N}$ defined by $\mathsf{Tower}(0) = 1$ and $\mathsf{Tower}(n{+}1) = 2^{\mathsf{Tower}(n)}$ is nonelementary. We say that a *function* $f : \mathbb{N} \to \mathbb{N}$ is Tower-*bounded* if there is an elementary function g such that $f(n) \leq \mathsf{Tower}(g(n))$. By TOWER we denote the class of decision problems solvable by Turing machines with Tower-bounded time (or space).

Class ACK, and Ackermann-Hardness. Let the family $f_0, f_1, f_2, \dots$ of functions be defined by putting $f_0(n) = n{+}1$ and $f_{k+1}(n) = f_k(f_k(\dots f_k(n)\dots))$ where f_k is applied $n{+}1$ times. By (our version of) the *Ackermann function* we

mean the function f_A defined by $f_A(n) = f_n(n)$; it is a "simplest" non-primitive recursive function. A decision problem belongs to the class ACK if it is solvable in time (or space) $f_A(g(n))$ where g is a primitive recursive function. It is the ACK-hardness, called Ackermann-hardness, what is important here. We refer to [19] for a full discussion, but for our use the following restricted version suffices.

We define HP-Ack as the problem that asks, given a Turing machine M, an input w, and some $n \in \mathbb{N}$, whether M halts on w within $f_A(n)$ steps. We say that a *problem* $\mathcal{P}$ is *Ackermann-hard* if HP-Ack is reducible to $\mathcal{P}$, or to the complementary problem co-$\mathcal{P}$, by a standard polynomial many-one reduction.

3 Equivalence of Det-FO Grammars Is in TOWER

In this section we show that the lengths of witnesses for pairs of nonequivalent terms in LTSs associated with det-FO grammars are Tower-bounded. It is then obvious that the equivalence problem for det-FO grammars is in TOWER.

To make this precise, we define the function MaxFEL : $\mathbb{N} \to \mathbb{N}$ ("Maximal Finite Eq-Level") as given below. We stipulate $\max \emptyset = 0$, and by $size(\mathcal{G}, T, U)$ we mean the size of a standard presentation of grammar $\mathcal{G}$ and terms T, U. For any $j \in \mathbb{N}$, we put

$$\text{MaxFEL}(j) = \max\{\, e \mid \text{there are a det-FO grammar } \mathcal{G} \text{ and terms } T, U \text{ such that } T \not\sim U \text{ in } \mathcal{L}_{\mathcal{G}},\ size(\mathcal{G}, T, U) \leq j, \text{ and } \text{EqLv}(T, U) = e \,\}.$$

Theorem 2. *Function* MaxFEL *(for det-FO grammars) is* Tower*-bounded.*

Corollary 3. *The equivalence problem for det-FO grammars, as well as for DPDA, is in* TOWER*.*

In Section 3.2 we describe a proof of Theorem 2 at a partly informal level. Some more formalized proof parts are then given in Section 3.3. Before starting with the proof, we first observe some important facts in Section 3.1. These facts are more or less straightforward, and an "impatient" reader might thus have only a quick look at Section 3.1 and read Section 3.2 immediately, returning to Section 3.1 if/when needed.

3.1 Compositionality of Terms, and Safe Changes of Substitutions

Given a det-FO grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$, the LTS $\mathcal{L}_{\mathcal{G}}$ is *deterministic*, and it thus has the properties captured by Prop. 1. We now note some other properties, using the *structure of states* of $\mathcal{L}_{\mathcal{G}}$, i.e. the structure of terms. (Recall that by a "term" we mean a "regular term", unless we explicitly say a "finite term".)

It is useful to extend the relations $\sim_k$ and $\sim$ to substitutions. For $\sigma, \sigma' : \text{Var} \to \text{Terms}_{\mathcal{N}}$ and $k \in \mathbb{N} \cup \{\omega\}$ we put

$$\sigma \sim_k \sigma' \text{ if } \sigma(x_i) \sim_k \sigma'(x_i) \text{ for all } x_i \in \text{Var}.$$

We also put $\text{EqLv}(\sigma, \sigma') = \max\{\, k \mid \sigma \sim_k \sigma' \}$.

Congruence Properties, and the Limit of a Repeated Substitution
The items 1, 2 in the next proposition (Prop. 4) state the simple fact that $\sim_k$ are congruences (for all $k \in \mathbb{N} \cup \{\omega\}$) in our term-setting. The item 3 states a basic fact underpinning our use of *regular* terms.

The point where and why we come to regular terms even when starting with finite terms can be *roughly* described as follows: If we can replace subterms $\sigma(x_i)$ in a pair $(E\sigma, F\sigma)$ with $H\sigma$, while keeping the same eq-level, i.e. having $\text{EQLV}(E\sigma, F\sigma) = \text{EQLV}(E\{(x_i, H)\}\sigma, F\{(x_i, H)\}\sigma)$, then we can repeat such replacing of σ with $\{(x_i, H)\}\sigma$ forever, keeping the same eq-level all the time. The limit of this process is the pair $(E'\sigma, F'\sigma)$ where $E' = E\{(x_i, H)\}\{(x_i, H)\}\cdots$ and $F' = F\{(x_i, H)\}\{(x_i, H)\}\cdots$. In $(E'\sigma, F'\sigma)$ the value $\sigma(x_i)$ is irrelevant, and we can thus "remove" x_i from the support of σ (as described below).

We note that the limit $H' = H\{(x_i, H)\}\{(x_i, H)\}\{(x_i, H)\}\cdots$ is a well-defined regular term, when H is a (regular) term. If $H = x_i$, then $H' = x_i$, and otherwise H' is the unique term satisfying $H' = H\{(x_i, H')\}$; we note that $H' = H$ if x_i does not occur in H, which includes the case $H = x_j$ for $j \neq i$. (A graph presentation of H' arises from a graph presentation of H by redirecting any arc leading to x_i towards the root.)

We note that $H \neq x_i$ implies that x_i does not occur in H'. We also observe that if x_i does not occur in a term G then $\sigma(x_i)$ plays no role in the composition $\{(x_i, G)\}\sigma$. In this case $\{(x_i, G)\}\sigma = \{(x_i, G)\}\sigma_{[-x_i]}$ where $\sigma_{[-x_i]}$ arises from σ by removing x_i from the support (if it is there), i.e.,

$$\sigma_{[-x_i]}(x_i) = x_i \text{ and } \sigma_{[-x_i]}(x_j) = \sigma(x_j) \text{ for } j \neq i.$$

Proposition 4

1. *If $E \sim_k F$, then $E\sigma \sim_k F\sigma$. Hence* $\text{EQLV}(E, F) \leq \text{EQLV}(E\sigma, F\sigma)$.
2. *If $\sigma \sim_k \sigma'$ then $E\sigma \sim_k E\sigma'$. Hence* $\text{EQLV}(\sigma, \sigma') \leq \text{EQLV}(E\sigma, E\sigma')$. *Moreover, if $\sigma \not\sim \sigma'$ and $E \notin \text{VAR}$ then* $\text{EQLV}(\sigma, \sigma') < \text{EQLV}(E\sigma, E\sigma')$.
3. *If $\sigma(x_i) \sim_k H\sigma$ and $H \neq x_i$, then $\sigma \sim_k \{(x_i, H')\}\sigma_{[-x_i]}$ where $H' = H\{(x_i, H)\}\{(x_i, H)\}\cdots$.*

Proof. The points 1 and 2 can be easily shown by induction on k; for $k = \omega$ we use the fact that $\sim = \bigcap_{j\in\mathbb{N}} \sim_j$. It is also obvious that $\text{EQLV}(E\sigma, E\sigma') \geq |w| + \text{EQLV}(\sigma, \sigma')$ if w is a shortest word such that $E \xrightarrow{w} x_j$ for some $x_j \in \text{VAR}$; if there is no such w, then $E\sigma \sim E\sigma'$.

3. Suppose $\sigma(x_i) \sim_k H\sigma$ and $H \neq x_i$, and put $\sigma' = \{(x_i, H')\}\sigma_{[-x_i]}$; hence $\sigma' = \{(x_i, H')\}\sigma$ since x_i does not occur in H'. We need to show $\sigma \sim_k \sigma'$.

Since $\sigma'(x_j) = \sigma(x_j)$ for $j \neq i$, we have $\text{EQLV}(\sigma, \sigma') = \text{EQLV}(\sigma(x_i), \sigma'(x_i)) = \text{EQLV}(\sigma(x_i), H'\sigma)$. Hence it suffices to show $\text{EQLV}(\sigma(x_i), H'\sigma) \geq k$; we assume $\sigma(x_i) \not\sim H'\sigma$, since otherwise we are done.

Since $H' = H\{(x_i, H')\}$, we deduce that $H'\sigma = H\sigma'$. Using 2 (and the fact $\sigma'(x_j) = \sigma(x_j)$ when $H = x_j$), we deduce that $\text{EQLV}(H\sigma, H\sigma') > \text{EQLV}(\sigma, \sigma') = \text{EQLV}(\sigma(x_i), H'\sigma)$.

Hence $\text{EQLV}(\sigma(x_i), H'\sigma) = \text{EQLV}(\sigma(x_i), H\sigma) \geq k$ (by using Prop. 1(1)). □

Getting an "Equation" $\sigma(x_i) \sim_k H\sigma$
The (technical) item 1 in the next proposition (Prop. 5) is trivial. The item 2 "completes" the item 1 in Prop. 4: Roughly speaking, if we can "increase" $\text{EQLV}(E, F)$ by applying some σ to both E and F, then the reason is that any witness w for (E, F) reaches some x_i on one side and $H \neq x_i$ on the other side. We have $\text{EQLV}(x_i, H) = 0$ but $\text{EQLV}(\sigma(x_i), H\sigma)$ might be larger.

Proposition 5

1. *If* $E \xrightarrow{w} x_i$, $F \xrightarrow{w} H$ *or* $E \xrightarrow{w} H$, $F \xrightarrow{w} x_i$ *where* $H \neq x_i$, *then for any* σ *we have* $\text{EQLV}(\sigma(x_i), H\sigma) \geq \text{EQLV}(E\sigma, F\sigma) - |w|$.
2. *If* $\text{EQLV}(E, F) < \text{EQLV}(E\sigma, F\sigma)$ *for some* σ, *then for any witness* w *for* (E, F) *there are some* $x_i \in \text{SUPP}(\sigma)$ *and* $H \neq x_i$ *such that* $E \xrightarrow{w} x_i$, $F \xrightarrow{w} H$ *or* $E \xrightarrow{w} H$, $F \xrightarrow{w} x_i$.

Proof. 1. Since $E \xrightarrow{w} x_i$ implies $E\sigma \xrightarrow{w} \sigma(x_i)$ and $F \xrightarrow{w} H$ implies $F\sigma \xrightarrow{w} H\sigma$, the claim follows from Prop. 1(3).

2. By induction on $e = \text{EQLV}(E, F)$. We cannot have $(E, F) = (x_i, x_i)$ since $E \not\sim F$; if $\{E, F\} = \{x_i, H\}$ for $H \neq x_i$, then we are done: $e = 0$, $w = \varepsilon$, and if $x_i \notin \text{SUPP}(\sigma)$ then $\{E\sigma, F\sigma\} = \{x_i, H\sigma\}$, in which case $\text{EQLV}(x_i, H\sigma) > 0$ implies that $H = x_j$ (for $j \neq i$) and $\sigma(x_j) = x_i$, whence $x_j \in \text{SUPP}(\sigma)$.

We thus assume that both $\text{ROOT}(E)$ and $\text{ROOT}(F)$ are nonterminals. If $e = 0$ (i.e., the roots enable different sets of actions), then $\text{EQLV}(E\sigma, F\sigma) = 0$ – a contradiction; hence $e > 0$. Then for each $a \in \Sigma$ where $E \xrightarrow{a} E'$, $F \xrightarrow{a} F'$ and $\text{EQLV}(E', F') = e-1$ (hence for each a "starting" a witness for (E, F)) we have $\text{EQLV}(E'\sigma, F'\sigma) \geq \text{EQLV}(E\sigma, F\sigma)-1 > e-1$. By the induction hypothesis any witness w' for (E', F') satisfies the claim, and thus any witness aw' for (E, F) satisfies the claim as well. □

Safe Changes of Subterms in a Pair of Terms (Keeping the Eq-Level)
The form of the next proposition is tailored to match our later use. The crux of the item 1 (of Prop. 6) trivially follows from the already established facts: we can "safely" replace a subterm V in one term of a pair (T, U) with another subterm V' if $\text{EQLV}(V, V') > \text{EQLV}(T, U)$. By "safely" we mean that the eq-level does not change: if the arising pair is (T', U), say, then $\text{EQLV}(T', U) = \text{EQLV}(T, U)$; moreover, even the witness sets for (T, U) and (T', U) are the same.

The item 2 is slightly subtler: if we want to safely replace $(E\sigma, F\sigma)$ with $(E\sigma', F\sigma')$ (thus replacing the relevant subterms on both sides simultaneously), then a (substantially) weaker condition for $\text{EQLV}(\sigma, \sigma')$ suffices. Roughly speaking, from $(E\sigma', F\sigma')$ we should first perform a word at least as long as a witness for (E, F) before the change of substitutions might matter (regarding the eq-level). For a safe replacing, i.e. for guaranteeing $\text{EQLV}(E\sigma', F\sigma') = \text{EQLV}(E\sigma, F\sigma)$, it thus suffices that $\text{EQLV}(\sigma, \sigma') > \text{EQLV}(E\sigma, F\sigma) - \text{EQLV}(E, F)$.

Proposition 6

1. *If* $\text{EQLV}(\sigma, \sigma') > \text{EQLV}(G\sigma, U)$, *then the witness sets for* $(G\sigma, U)$ *and* $(G\sigma', U)$ *are the same (and* $\text{EQLV}(G\sigma, U) = \text{EQLV}(G\sigma', U)$*).*

2. *Assume* $\text{EQLV}(E,F) = k < \omega$ *and* $\text{EQLV}(E\sigma, F\sigma) = e$ *(hence* $k \le e$*). If* $\sigma \sim_{e-k+1} \sigma'$ *then* $\text{EQLV}(E\sigma', F\sigma') = e$.

Proof
1. Since $\text{EQLV}(G\sigma, G\sigma') \ge \text{EQLV}(\sigma, \sigma')$, the claim follows from Prop. 1(1).
2. If $e = \omega$, then we have $\sigma \sim \sigma'$, and thus $E\sigma' \sim E\sigma \sim F\sigma \sim F\sigma'$. Suppose now a counterexample with the minimal e; hence $\text{EQLV}(E\sigma', F\sigma') = e' > e$. (If $e' < e$, then swapping σ, σ' contradicts our minimality assumption.)

If $\{E,F\} = \{x_i, H\}$, then $H \ne x_i$, $k = 0$, and $\text{EQLV}(\sigma(x_i), H\sigma) = e$. Since $\sigma \sim_{e+1} \sigma'$, we have $\text{EQLV}(\sigma'(x_i), H\sigma') = e$, a contradiction. Otherwise (when $\{E,F\} \ne \{x_i, H\}$, and thus $\text{ROOT}(E)$ and $\text{ROOT}(F)$ are nonterminals) we must have $1 \le k \le e < e'$, and there is $a \in \Sigma$ such that $E \xrightarrow{a} E'$, $F \xrightarrow{a} F'$ and $\text{EQLV}(E'\sigma, F'\sigma) = e{-}1$. We thus have

$$k{-}1 \le k' = \text{EQLV}(E', F') \le \text{EQLV}(E'\sigma, F'\sigma) = e{-}1 < \\ < e'{-}1 \le \text{EQLV}(E'\sigma', F'\sigma').$$

Since $\sigma \sim_{e-1-k'+1} \sigma'$, we contradict our minimality assumption. □

We now state a simple corollary of the previous facts; its form will be particularly useful in an inductive argument based on decreasing the support of certain substitutions.

Corollary 7. *Suppose* $\text{EQLV}(E\sigma, F\sigma) > \text{EQLV}(E\sigma'\sigma, F\sigma'\sigma)$. *Then* $E \not\sim F$ *and for any witness* w *for* (E,F) *there are* x_i, $H \ne x_i$ *such that* $E \xrightarrow{w} x_i$, $F \xrightarrow{w} H$, *or vice versa. For any such* x_i, H *we have*

$$\text{EQLV}\,(\,E\sigma'\sigma, F\sigma'\sigma\,) = \text{EQLV}\,(\,E\sigma'\{(x_i,H')\}\sigma_{[-x_i]}, F\sigma'\{(x_i,H')\}\sigma_{[-x_i]}\,),$$

where $H' = H\{(x_i,H)\}\{(x_i,H)\}\cdots$.

Proof. Let the assumption hold. We can thus write

$$k = \text{EQLV}(E,F) \le \text{EQLV}(E\sigma'\sigma, F\sigma'\sigma) = e' < e = \text{EQLV}(E\sigma, F\sigma).$$

Hence $E \not\sim F$. Let us fix a witness w for (E,F); it has the associated x_i, H by Prop 5(2), and $|w| = k$. We deduce $\sigma(x_i) \sim_{e-k} H\sigma$ (by Prop 5(1)), and $\sigma \sim_{e-k} \{(x_i,H')\}\sigma_{[-x_i]}$ (by Prop. 4(3)).

Since $k \le k' = \text{EQLV}(E\sigma', F\sigma') \le e' < e$, we have $\sigma \sim_{e'-k'+1} \{(x_i,H')\}\sigma_{[-x_i]}$. The claim thus follows from Prop. 6(2). □

3.2 Proof of Theorem 2 (Partly Informal)

Convention (on Small Numbers, and on (A,i)-Sink Words)
In the following reasoning we assume a fixed det-FO grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$. To ease the presentation, we also use informally-sounding words like "small" or "short". Nevertheless, we will *not further formalize* such usages, since the respective expressions have rigorous meanings: we view a *number* as *small* if it is bounded by an elementary function of $size(\mathcal{G})$ (independently of any initial terms

T_0, U_0). I.e., we (implicitly) claim in each such case that there is an elementary function $f : \mathbb{N} \to \mathbb{N}$, independent of our concrete grammar $\mathcal{G}$, such that the value $f(size(\mathcal{G}))$ is an upper bound for our "small number". When we say that a *word*, or a *sequence* in general, is *short*, then we mean that its length is small. A *set* is *small* if its cardinality is small. A *term* is *small* when its (presentation) size is small.

E.g., we can easily check that if U is shortly reachable from W, i.e., if there is a short w such that $W \xrightarrow{w} U$, then there are small *finite* terms H, F such that $W = H\sigma$, $U = F\sigma$ (F being shortly reachable from H) where $\text{RANGE}(\sigma)$ contains only subterms of W occurring in small depths in W. This is based on the fact that performing one transition from a term W results in $E\sigma$ where E is the rhs of a rule in $\mathcal{R}$ and $\text{RANGE}(\sigma)$ contains only depth-1 subterms of W; we note that $size(E)$ is bounded by the small number TRANSINC ("Increase by a Transition") that is equal to the maximal size of the right-hand sides of the rules in $\mathcal{R}$.

We can explicitly note that $W \xrightarrow{w} U$ implies $size(U) \leq size(W) + |w| \cdot \text{TRANSINC}$; if W is finite then also $\text{HEIGHT}(U) \leq \text{HEIGHT}(W) + |w| \cdot \text{TRANSINC}$.

A useful exercise is to note that if there is a word $w \in \Sigma^*$ such that $A(x_1, \ldots, x_m) \xrightarrow{w} x_i$, called an (A, i)-*sink word*, then a shortest such word is short. (More details for this claim are in Section 3.3.) We say that any (A, i)-sink word *exposes* the ith *root-successor* in any term $A(V_1, \ldots, V_m)$. If there is no (A, i)-sink word, then the ith root-successor V_i in $A(V_1, \ldots, V_m)$ is *non-exposable*, and plays no role (i.e., by any change of V_i another equivalent term arises). In fact, we will assume that *all root-successors are exposable* (since the grammar $\mathcal{G}$ can be harmlessly adjusted to satisfy this), and

$$\text{we fix a shortest } (A, i)\text{-sink word } w_{[A,i]}$$

for all $A \in \mathcal{N}$, $i \in [1, arity(A)]$. We also define the following small number:

$$M_0 = 1 + \max\{\, |w_{[A,i]}|; A \in \mathcal{N}, i \in [1, arity(A)] \,\}. \tag{2}$$

Start of the Proof of Theorem 2

We have fixed a det-FO grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$, and we now assume a witness u_0 for (T_0, U_0) where T_0, U_0 are *finite* terms; hence $T_0 \not\sim U_0$ in $\mathcal{L}_\mathcal{G}$ and $|u_0| = \text{EQLV}(T_0, U_0)$. (The restriction to *finite* T_0, U_0 is here technically convenient, not really crucial.) If $u_0 = a_1 a_2 \ldots a_k$, then $(T_0, U_0), u_0$ *generate the sequence*

$$(T_0, U_0) \xrightarrow{a_1} (T_1, U_1) \xrightarrow{a_2} \ldots \xrightarrow{a_k} (T_k, U_k) \tag{3}$$

where we write $(T, U) \xrightarrow{a} (T', U')$ instead of $T \xrightarrow{a} T'$, $U \xrightarrow{a} U'$. We note that $\text{EQLV}(T_i, U_i) = k - i$; the *sequence* $(T_0, U_0), (T_1, U_1), \ldots, (T_k, U_k)$ is thus *eqlevel-decreasing*, i.e., it satisfies $\text{EQLV}(T_0, U_0) > \text{EQLV}(T_1, U_1) > \cdots > \text{EQLV}(T_k, U_k)$. Moreover, $a_{i+1} a_{i+2} \ldots a_k$ is a witness for (T_i, U_i).

We prove Theorem 2 by deriving a Tower-bounded function (of $size(\mathcal{G}, T_0, U_0)$) that bounds the length k of (3); we use two macro-steps described below.

First Macro-step: Controlled Balancing. Both paths $T_0 \xrightarrow{u_0} T_k$, $U_0 \xrightarrow{u_0} U_k$ in (3) can have "sinking" and "non-sinking" segments distributed quite differently. By sinking we mean "exposing subterms" (corresponding to popping, i.e. stack-height decreasing, in DPDA), by non-sinking we mean the opposite (the stack-height can only increase). We will now show particular "balancing steps" that allow us to stepwise modify the pairs (T_j, U_j) in (3) so that the component terms in the final modified $(\overline{T}_j, \overline{U}_j)$ are more "balanced" (i.e., more "close to each other") while the eq-levels are kept unchanged (i.e., $\text{EQLV}(\overline{T}_j, \overline{U}_j) = \text{EQLV}(T_j, U_j)$). We will have $(\overline{T}_0, \overline{U}_0) = (T_0, U_0)$, and all $\overline{T}_j, \overline{U}_j$ will be finite terms.

A slight control of balancing steps will give rise to a certain subsequence $(\overline{T}_{i_0}, \overline{U}_{i_0}), (\overline{T}_{i_1}, \overline{U}_{i_1}), \ldots, (\overline{T}_{i_\ell}, \overline{U}_{i_\ell})$ of the sequence of pairs in the "balanced version" of (3), where $i_0 = 0$. The subsequence satisfies that each pair $(\overline{T}_{i_j}, \overline{U}_{i_j})$ is "close" to a certain finite term W_j, called a "(balancing) pivot"; this also entails that the heights of $\overline{T}_{i_j}$ and $\overline{U}_{i_j}$ are bounded by $\text{HEIGHT}(W_j) + x$ where x is a small number.

Moreover, the sequence $W_0, W_1, \ldots, W_\ell$ of pivots will be "sufficiently representative" in the sense that for any $r \in [0, k]$ we can bound $\text{HEIGHT}(\overline{T}_r)$ and $\text{HEIGHT}(\overline{U}_r)$ by $\text{HEIGHT}(W_j) + y$ where y is a small number and j is the largest such that $i_j \leq r$. In other words, the heights of terms in the segment $(\overline{T}_{i_j}, \overline{U}_{i_j})$, $(\overline{T}_{i_j+1}, \overline{U}_{i_j+1})$, $(\overline{T}_{i_j+2}, \overline{U}_{i_j+2})$, $\ldots$, $(\overline{T}_r, \overline{U}_r)$, where $r = i_{j+1}-1$ if $j < \ell$ and $r = k$ if $j = \ell$, are bounded by $\text{HEIGHT}(W_j) + z$ for a small number z.

Hence it suffices to bound ℓ (the length of the subsequence) by a Tower-bounded function (of $size(\mathcal{G}, T_0, U_0)$); this will be achieved by the second macro-step, by using the "pivot-path" $W_0 \xrightarrow{w_1} W_1 \xrightarrow{w_2} \cdots \xrightarrow{w_\ell} W_\ell$ that will be precisely defined at the end of this first macro-step.

Balancing Steps

We say that a *path* $V \xrightarrow{u}$ is *root-performable* if $A(x_1, \ldots, x_m) \xrightarrow{u}$ where $A = \text{ROOT}(V)$. For technical convenience we define a *non-sink segment* as a root-performable path $V \xrightarrow{w} V'$ where $|w| = M_0$. (Thus $V \xrightarrow{w} V'$ might expose a root-successor in V, but only by the last step if at all; so $V \xrightarrow{w} V'$ surely "misses" the possibility to expose some root-successor in V as quickly as possible.)

Let $T \xrightarrow{w} T'$ be a non-sink segment for a part

$$(T, U) \xrightarrow{w} (T', U')$$

of (3); hence $(T, U) = (T_{r-M_0}, U_{r-M_0})$, $(T', U') = (T_r, U_r)$, and $w = a_{r-M_0+1}\, a_{r-M_0+2} \ldots a_r$, for some $r \in [M_0, k]$. Then $T = A(V_1, \ldots, V_m)$, $A(x_1, \ldots, x_m) \xrightarrow{w} G$, and $T \xrightarrow{w_{[A,j]}} V_j$ for each $j \in [1, m]$. We recall that $|w_{[A,j]}| < |w| = M_0$.

Since $T \sim_{M_0} U$, we must have $U \xrightarrow{w_{[A,j]}} V'_j$ for some V'_j, and $\text{EQLV}(V_j, V'_j) > \text{EQLV}(T', U')$ (by Prop. 1(2,3)).

We thus have $T' = G\sigma$ where $\sigma(x_j) = V_j$ for $j \in [1, m]$, and we can

$$\text{replace the pair } (T', U') = (G\sigma, U') \text{ with } (T'', U') = (G\sigma', U')$$

where $\sigma'(x_j) = V'_j$. After replacing (T', U') with (T'', U'), the whole respective suffix of (3) changes: we generate it by $(T'', U'), u'$ where $u' = a_{r+1}\, a_{r+2} \cdots a_k$. This is safe in the sense that the witness sets (and the eq-levels) for (T'', U') and (T', U') are the same (by Prop. 6(1)).

In our concrete notation, we have transformed (3) to

$$(T_0, U_0) \xrightarrow{a_1} \cdots (T_{r-1}, U_{r-1}) \xrightarrow{a_r} (\widehat{T}_r, U_r) \xrightarrow{a_{r+1}} (\widehat{T}_{r+1}, U_{r+1}) \ldots \xrightarrow{a_k} (\widehat{T}_k, U_k)$$

where $T_{r-1} \xrightarrow{a_r} \widehat{T}_r$ might be not a valid transition but the eq-levels have not changed, i.e., $\text{EqLv}(\widehat{T}_{r+i}, U_{r+i}) = \text{EqLv}(T_{r+i}, U_{r+i})$ for $i \in [0, k-r]$. (The notation $\widehat{T}_j$ should not be mixed with $\overline{T}_j$ discussed previously; any T_j can undergo several changes, though at most one "balancing", before becoming the "final" $\overline{T}_j$.)

Since w is short ($|w| = M_0$), the pair (T'', U') (i.e. $(\widehat{T}_r, U_r)$) is "balanced" in the sense that both terms are "close" to one term, namely to U: we have $(T'', U') = (G\sigma', U')$ where G is a small finite term (since shortly reachable from $A(x_1, \ldots, x_m)$), and $\text{RANGE}(\sigma')$ contains only terms that are shortly reachable from U, and U' is itself shortly reachable from U. We capture the discussed closeness by the notation

$$U \models (T'', U'),$$

and we also say that U is the *pivot* of our *balancing step*, while (T'', U') is called the *balancing result*.

Analogously to the described *left-balancing step*, where we replace $(G\sigma, U')$ with $(G\sigma', U')$, we define a *right-balancing step*, where ($U \xrightarrow{w} U'$ is a non-sink segment and) we replace $(T', G\sigma)$ with $(T', G\sigma')$; here T is the pivot and we have $T \models (T', G\sigma')$.

An important feature of our (ternary) predicate $\models$ is that

$$W \models (T, U) \text{ implies that } W = H\sigma,\ T = E\sigma,\ U = F\sigma$$

for some small finite terms H, E, F, where $\text{RANGE}(\sigma)$ contains only subterms of W occurring in small depths in W; moreover, at least one of E, F is shortly reachable from H (and thus at least one of T, U is shortly reachable from W). (A precise definition of $W \models (T, U)$ is given in Section 3.3.)

Quadruple-Sequences, and Pivot Paths

We use the described balancing steps as follows. We first artificially create a "pivot" $W_0 = B(T_0, U_0)$ for a (possibly added) nonterminal B so that we have $W_0 \models (T_0, U_0)$. Our aim is to transform (3), by stepwise balancing, into a certain sequence of a different kind, namely to

$$((W_0, T_0, U_0), u_0) \rightsquigarrow ((W_1, T'_1, U'_1), u_1) \rightsquigarrow \cdots \rightsquigarrow ((W_\ell, T'_\ell, U'_\ell), u_\ell) \qquad (4)$$

where also $W_i \models (T'_i, U'_i)$ for $i \in [1, \ell]$ and u_i is a proper suffix of u_{i-1} and a witness for (T'_i, U'_i). (The sequence (T_0, U_0), (T'_1, U'_1), $\ldots$, (T'_ℓ, U'_ℓ) will be the subsequence $(\overline{T}_{i_0}, \overline{U}_{i_0}), (\overline{T}_{i_1}, \overline{U}_{i_1}), \ldots, (\overline{T}_{i_\ell}, \overline{U}_{i_\ell})$ discussed earlier.)

We start with the one-element sequence $((W_0, T_0, U_0), u_0)$. Then we traverse (3) from left to right, and we perform a first possible balancing step (if there is any non-sink segment in $T_0 \xrightarrow{u_0}$ or $U_0 \xrightarrow{u_0}$). We prolong our sequence to $((W_0, T_0, U_0), u_0) \rightsquigarrow ((W_1, T_1', U_1'), u_1)$ where W_1 is the pivot and (T_1', U_1') the balancing result of this step; u_1 is the suffix of u_0 that is a witness for (T_1', U_1').

Now we continue, by traversing the sequence generated by $(T_1', U_1'), u_1$, or by $(T_j', U_j'), u_j$ in general; this generated sequence corresponds to the current version of the respective suffix of the stepwise modified (3). We look for the first possible balancing step in this sequence, with *one proviso*: if we have $W_j \models (G\sigma', U_j')$ after a left-balancing step, then the next balancing step can be a right-balancing (thus changing the pivot-side) only if the "rest-head" G has been in the meantime erased, i.e., some $\sigma'(x_i)$ (that is shortly reachable from W_j) has been exposed. Hence shortly after a left-balancing step corresponding to $W_j \models (G\sigma', U_j')$ either another left-balancing step is performed or G is erased and balancing at both sides is again allowed.

After any right-balancing step, an analogous proviso applies. This guarantees that W_{j+1} is reachable from W_j, by a certain path $W_j \xrightarrow{w_{j+1}} W_{j+1}$ in which at most *a small number of non-sink segments occurs*: either the path $W_j \xrightarrow{w_{j+1}} W_{j+1}$ is short, or it has a short prefix after which no non-sink segment occurs. (More details are in Section 3.3.)

We finish creating the sequence (4) when the paths $T_\ell' \xrightarrow{u_\ell}$, $U_\ell' \xrightarrow{u_\ell}$ do not allow further balancing; it is clear that these paths then must be either short or sinking all the time (in which case the heights of terms in the paths $T_\ell' \xrightarrow{u_\ell}$, $U_\ell' \xrightarrow{u_\ell}$ are successively decreasing).

To summarize, from the sequence (3), whose length k we want to bound, we have come to the sequence (4) that also has the associated *pivot-path*

$$W_0 \xrightarrow{w_1} W_1 \xrightarrow{w_2} \cdots \xrightarrow{w_\ell} W_\ell, \tag{5}$$

where the number of non-sink segments in each subpath $W_{j-1} \xrightarrow{w_j} W_j$ is small.

Second Step: Deriving a Tower-bound from a Pivot Path. As we made clear, for each term W there is only a small number of pairs (T, U) such that $W \models (T, U)$. Since $(T_0, U_0), (T_1', U_1'), (T_2', U_2'), \ldots, (T_\ell', U_\ell')$ in (4) is an eqlevel-decreasing sequence, we have no repeat of a pair here, and thus the number of occurrences of each concrete W_j in the sequence $W_0, W_1, \ldots, W_\ell$ is small.

We can intuitively note that if the sequence (3) is "very long" (w.r.t. $size(\mathcal{G}, T_0, U_0)$) then the maximal $\textsc{Height}(W_j)$ is "much larger" than $\textsc{Height}(W_0)$. In this case the pivot path (5) must have "long increasing segments". In a long increasing segment the pivots W_j are "frequent" since each (sub)path $W_{j-1} \xrightarrow{w_j} W_j$ has at most a small number of non-sink segments.

We formalize this intuition by help of "stair sequences" that correspond to the standard "stair-growing" stack-contents in a path from one PDA-configuration to a larger configuration.

Stair Sequences. Let us present (5) as

$$W_0 \xrightarrow{w'_1} V_1 \xrightarrow{w''_1} W_1 \xrightarrow{w'_2} V_2 \xrightarrow{w''_2} W_2 \cdots \xrightarrow{w'_\ell} V_\ell \xrightarrow{w''_\ell} W_\ell \tag{6}$$

where V_j is the term in the path $W_{j-1} \xrightarrow{w_j} W_j$ for which the respective w'_j is the shortest possible such that the path $V_j \xrightarrow{w''_j} W_j$ does not expose any root-successor in V_j (and is thus also root-performable). There is always such V_j; in some cases we might have $V_j = W_{j-1}$ ($w'_j = \varepsilon$) or $V_j = W_j$ ($w''_j = \varepsilon$).

Hence $V_j = A(x_1, \ldots, x_m)\sigma$ and $W_j = G\sigma$ where $A(x_1, \ldots, x_m) \xrightarrow{w''_j} G$ and $\text{RANGE}(\sigma)$ contains root-successors in V_j; since the number of non-sink segments in $W_{j-1} \xrightarrow{w_j} W_j$ is small, G is a small finite term. We say that a subsequence

$$i_1 < i_2 < \cdots < i_r \tag{7}$$

of the sequence $1, 2, \ldots, \ell$ is a *stair sequence* if for each $j \in [1, r-1]$ the subpath

$$V_{i_j} \xrightarrow{w''_{i_j}} \xrightarrow{w'_{i_j+1}} \cdots \xrightarrow{w''_{i_r-1}} \xrightarrow{w'_{i_r}} V_{i_r}$$

of (6) does not expose any root-successor in V_{i_j} (and is thus root-performable). Moreover, for convenience we require that the sequence is maximal in the sense that we would violate the above condition by inserting any i where $i_j < i < i_{j+1}$ for some $j \in [1, r-1]$. This requirement guarantees the *small-stair property*: for any $j \in [1, r-1]$ we have $V_{i_j} = A(x_1, \ldots, x_m)\sigma$ and $V_{i_{j+1}} = F\sigma$ where $A(x_1, \ldots, x_m) \xrightarrow{w''_{i_j}} \xrightarrow{w'_{i_j+1}} \cdots \xrightarrow{w''_{i_{j+1}-1}} \xrightarrow{w'_{i_{j+1}}} F$ and F is a small finite term (though *not all* terms in the path $A(x_1, \ldots, x_m) \longrightarrow \cdots \longrightarrow F$ are claimed to be small).

It Suffices to Get a Tower-bound on the Lengths of Stair Sequences Later we show that the lengths of stair sequences are bounded by a Tower-bounded function f of $size(\mathcal{G})$, independently of T_0, U_0; now we assume such f. For any pivot W_j (in the sequence (5)) we then obviously have

$$\text{HEIGHT}(W_j) \leq \text{HEIGHT}(W_0) + \mathsf{stair} \cdot (1 + f(size(\mathcal{G})))$$

where stair is an appropriate small number.

Since the number of (finite) terms with the height at most $\mathrm{H} \in \mathbb{N}$ is bounded by $y \uparrow (m \uparrow \mathrm{H})$ for some small numbers m, y (recall that $\uparrow$ is the exponential operator), we easily deduce that the number of elements of the sequence (4) is bounded by a Tower-bounded function (of $size(\mathcal{G}, T_0, U_0)$).

For $j \in [0, \ell]$ we consider $((W_j, T'_j, U'_j), u_j)$ in (4), where we put $(T'_0, U'_0) = (T_0, U_0)$. Let $u_j = wu_{j+1}$; we put $w = u_\ell$ when $j = \ell$. It is easy to check that the heights of terms on the paths $T'_j \xrightarrow{w}$ and $U'_j \xrightarrow{w}$ are bounded by $\text{HEIGHT}(W_j)+x$ where x is a small number. Since the sequence generated by $(T'_j, U'_j), w$ is eqlevel-decreasing, and thus has no repeat, we get that $|w|$ is bounded by the value $(y \uparrow (m \uparrow (\text{HEIGHT}(W_j) + x)))^2$ for some small m, x, y.

We thus routinely derive a Tower-bounded function (of $size(\mathcal{G}, T_0, U_0)$) that bounds the length of u_0 and thus $\text{EQLV}(T_0, U_0)$. Hence after we show the

promised Tower-bounded function f (of $size(\mathcal{G})$) that bounds the lengths of stair sequences, the proof of Theorem 2 will be finished.

A Tower-bound on the Lengths of Stair Sequences
Let us fix a stair sequence $i_1 < i_2 < \cdots < i_r$, referring to the notation around (7). We now show that $r \leq f(size(\mathcal{G}))$ for a Tower-bounded function f (independent of $\mathcal{G}$).

By our definition, which also implies the small-stair property, we can easily observe that the sequence V_{i_1}, V_{i_2}, V_{i_3}, ..., V_{i_r} can be presented as

$$A_1\sigma',\ A_2\sigma'_1\sigma',\ A_3\sigma'_2\sigma'_1\sigma',\ \ldots,\ A_r\sigma'_{r-1}\sigma'_{r-2}\cdots\sigma'_1\sigma'$$

where we write shortly A_j instead of $A_j(x_1,\ldots,x_{m_j})$ and where $A_j(x_1,\ldots,x_{m_j}) \xrightarrow{w} A_{j+1}(x_1,\ldots,x_{m_{j+1}})\sigma'_j$ for the respective w from (6), i.e. $w = w''_{i_j}\cdots w'_{i_{j+1}}$. Hence the supports of σ' and σ'_j are subsets of $\{x_1,x_2,\ldots,x_m\}$ where m is the maximal arity of nonterminals, and $\text{RANGE}(\sigma'_j)$ contains the root-successors in the small finite term F such that $A_j(x_1,\ldots,x_{m_j}) \xrightarrow{w} F$. Hence σ'_j are small, i.e., they are small sets of small pairs of the form (x_i,E).

By the definition of (6), we can present W_{i_1}, W_{i_2}, W_{i_3}, ..., W_{i_r} as

$$G_1\sigma', G_2\sigma'_1\sigma', G_3\sigma'_2\sigma'_1\sigma', \ldots, G_r\sigma'_{r-1}\sigma'_{r-2}\cdots\sigma'_1\sigma' \tag{8}$$

where G_j are small finite terms. Recalling the discussion around the introduction of $W \models (T,U)$ (namely the form $W = H\sigma$, $T = E\sigma$, $U = F\sigma$), it is useful to rewrite (8) as

$$H_1\sigma,\ H_2\rho_1\sigma,\ H_3\rho_2\rho_1\sigma,\ \ldots,\ H_r\rho_{r-1}\rho_{r-2}\cdots\rho_1\sigma$$

where H_j are also small finite terms, but maybe with larger heights than G_j, guaranteeing that the sequence (T'_{i_1},U'_{i_1}), (T'_{i_2},U'_{i_2}), ..., (T'_{i_r},U'_{i_r}) (extracted from (4)) can be presented as

$$(E_1\sigma,F_1\sigma),(E_2\rho_1\sigma,F_2\rho_1\sigma),\ldots,(E_r\rho_{r-1}\rho_{r-2}\cdots\rho_1\sigma,F_r\rho_{r-1}\rho_{r-2}\cdots\rho_1\sigma) \tag{9}$$

where all E_j, F_j are small. This forces us to increase the supports of σ and ρ_j comparing to σ' and σ'_j (since $\text{RANGE}(\sigma)$ contains deeper subterms of V_{i_1}) but it is obvious that we can take the supports of σ and of all ρ_j as subsets of

$$\text{SUP}_0 = \{x_1,x_2,\ldots,x_{n_0}\} \tag{10}$$

where n_0 is small. Moreover, all terms in $\text{RANGE}(\rho_j)$ are small (but this is not claimed for σ). (We use the symbol "ρ" instead of a variant of "σ" to stress the small size of all ρ_j.) Given σ, (9) is fully determined by the sequence of triples

$$(E_1,F_1,\rho_0)\,(E_2,F_2,\rho_1)\,\cdots\,(E_r,F_r,\rho_{r-1}) \tag{11}$$

which can be viewed as a word in a small alphabet AL (consisting of the respective triples); we use ρ_0 for uniformity, defining it as the empty-support substitution.

We now note a useful combinatorial fact. Let us define a function $h:\mathbb{N}\to\mathbb{N}$ inductively as follows:

$$h(0) = 1, \text{ and } h(j{+}1) = h(j) \cdot (1 + q^{h(j)}) \text{ where } q = \text{CARD}(\mathsf{AL}).$$

Viewing q as a constant, h is obviously a Tower-bounded function. The pigeonhole principle implies that in any $h(1)$-long segment of (11), i.e. in any segment with length $h(1) = 1{+}\text{CARD}(\mathsf{AL})$, there are two different occurrences of one element of AL. Moreover, in any $h(j{+}1)$-long segment there are two different, non-overlapping, occurrences of one $h(j)$-long segment.

We aim to show that $r \leq h(n_0{+}1)$, where r is the length of our stair sequence, as well as of the sequences (9) and (11), and $n_0 = \text{CARD}(\text{SUP}_0)$ is introduced in (10). Since n_0 and $q = \text{CARD}(\mathsf{AL})$ are small numbers (bounded by elementary functions of $size(\mathcal{G})$), the value $h(n_0{+}1)$ is obviously bounded by $g(size(\mathcal{G}))$ for a Tower-bounded function g. Hence after we show $r \leq h(n_0{+}1)$ (in the following last part of this section), the proof of Theorem 2 will be finished.

Length r of Any Stair Sequence Is Less Than $h(n_0{+}1)$

We first introduce certain "recurrent-pattern" sequences, now using *general regular* terms and substitutions as ingredients, with *no size-restrictions*.

We define (n, ℓ)-presentations where $n, \ell \in \mathbb{N}$ (not to be mixed with ℓ in (5)). Each (n, ℓ)-presentation presents a sequence with 2^ℓ elements where an element is a pair of (regular) terms. A sequence that can be presented by an (n, ℓ)-presentation is called an (n, ℓ)-*sequence*. An (n, ℓ)-*presentation* consists of

- a set $\text{SUP} \subseteq \text{VAR}$ where $\text{CARD}(\text{SUP}) \leq n$,
- a pair (E, F) of (regular) terms, and
- substitutions $\sigma_1, \sigma_2, \ldots, \sigma_\ell$ and σ whose supports are subsets of SUP.

If $\ell = 0$, then the presented $(n, 0)$-sequence is the one-element sequence $(E\sigma, F\sigma)$. If $\ell > 0$, then the presented sequence (with 2^ℓ elements) arises by concatenating two $(n, \ell{-}1)$-sequences: the first half (with $2^{\ell-1}$ elements) is presented by SUP, (E, F), $\sigma_1, \sigma_2, \ldots, \sigma_{\ell-1}$, σ, and the second half by SUP, (E, F), $\sigma_1, \sigma_2 \ldots, \sigma_{\ell-1}$, and $\sigma' = \sigma_\ell \sigma$.

An example of an $(n, 2)$-sequence is

$$(E\sigma, F\sigma), (E\sigma_1\sigma, F\sigma_1\sigma), (E\sigma_2\sigma, F\sigma_2\sigma), (E\sigma_1\sigma_2\sigma, F\sigma_1\sigma_2\sigma) \tag{12}$$

if the supports of $\sigma, \sigma_1, \sigma_2$ are subsets of SUP with $\text{CARD}(\text{SUP}) \leq n$. If also $\text{SUPP}(\sigma_3) \subseteq \text{SUP}$, and we replace σ in (12) with $\sigma_3\sigma$, we get

$$(E\sigma_3\sigma, F\sigma_3\sigma), (E\sigma_1\sigma_3\sigma, F\sigma_1\sigma_3\sigma), (E\sigma_2\sigma_3\sigma, F\sigma_2\sigma_3\sigma), (E\sigma_1\sigma_2\sigma_3\sigma, F\sigma_1\sigma_2\sigma_3\sigma).$$

Put together, the above 8 pairs constitute an $(n, 3)$-sequence, presented by SUP, (E, F), σ_1, σ_2, σ_3, and σ.

We now prove the next claim, which also implies that any $h(n_0{+}1)$-long segment of the (eqlevel-decreasing) sequence (9) contains an $(n_0, n_0{+}1)$-subsequence (i.e. a subsequence that is an $(n_0, n_0{+}1)$-sequence).

Claim. *Any $h(\ell)$-long segment of (9) contains an (n_0, ℓ)-subsequence.*

Proof. We give an inductive definition (based on ℓ) of so called *good* (n_0, ℓ)-*presentations*. It will be guaranteed that each $h(\ell)$-long segment of (9) has an (n_0, ℓ)-subsequence with a *good* (n_0, ℓ)-*presentation*.

Any $h(0)$-long segment is an element $(E_i\rho_{i-1}\rho_{i-2}\dots\rho_1\sigma, F_i\rho_{i-1}\rho_{i-2}\dots\rho_1\sigma)$; it trivially constitutes an $(n_0,0)$-sequence, and we define its good $(n_0,0)$-presentation as SUP, (E,F), $\overline{\sigma}$ where $(E,F)=(E_i,F_i)$ and $\overline{\sigma}=\rho_{i-1}\rho_{i-2}\dots\rho_1\sigma$.

For a $h(\ell{+}1)$-long segment S of (9) we define a respective good $(n_0,\ell{+}1)$-presentation as follows. We take two $h(\ell)$-long non-overlapping subsegments S_1, S_2 of S such that the respective "images" of S_1 and S_2 in (11) are the same. (This is possible by the definition of h.)

Let SUP, (E,F), σ_1, σ_2, …, σ_ℓ, $\overline{\sigma}$ be a good (n_0,ℓ)-presentation of a subsequence of S_1, starting at (the relative) position p_1^{rel} inside S_1 and at (the absolute) position p_1^{abs} in (9); let p_2^{abs} denote the position in (9) that corresponds to the relative position p_1^{rel} in S_2. Our (inductive) construction of good presentations guarantees that $(E,F)=(E_{p_1^{abs}},F_{p_1^{abs}})$ and $\overline{\sigma}=\rho_{p_1^{abs}-1}\rho_{p_1^{abs}-2}\cdots\rho_1\sigma$. Another (inductive) property of good presentations is that SUP, (E,F), σ_1, σ_2, …, σ_ℓ, $\sigma_{\ell+1}\overline{\sigma}$, where where we put $\sigma_{\ell+1}=\rho_{p_2^{abs}-1}\rho_{p_2^{abs}-2}\cdots\rho_{p_1^{abs}}$, is a good (n_0,ℓ)-presentation of a subsequence of S_2 (since the images of S_1 and S_2 in (11) are the same). (Our definition in the case $\ell=0$ indeed guarantees these two properties.)

Then SUP, (E,F), σ_1, σ_2, …, $\sigma_{\ell+1}$, $\overline{\sigma}$ is defined to be a good $(n_0,\ell{+}1)$-presentation, presenting a subsequence of the $h(\ell{+}1)$-long segment S. The above used properties of good (n_0,ℓ)-presentations are obviously guaranteed for the defined good $(n_0,\ell{+}1)$-presentation as well. □

We observe that all elements of a $(0,\ell)$-sequence are the same. There is thus no eqlevel-decreasing $(0,\ell)$-sequence for $\ell>0$. The next claim shows that an eqlevel-decreasing (n,ℓ)-sequence with $\ell>0$ gives rise to an eqlevel-decreasing $(n{-}1,\ell{-}1)$-sequence (arising from the original even-index elements by subterm replacing that does not change the respective eq-levels). This implies that

$$\text{there is no eqlevel-decreasing } (n,\ell)\text{-sequence where } n<\ell;$$

in particular, there is no eqlevel-decreasing $(n_0,n_0{+}1)$-sequence. Using the previous claim, we deduce that there is no $h(n_0{+}1)$-long segment in (9); this implies $r<h(n_0{+}1)$, and the proof of Theorem 2 is finished.

Claim. *Let e_j denote the eq-level of the jth element of an (n,ℓ)-sequence, and assume $\ell>0$ and $e_1>e_2>\dots>e_{2\ell}$. Then $n>0$ and there is an $(n{-}1,\ell{-}1)$-sequence in which the eq-level of its jth element is e_{2j}.*

Proof. Let Seq be an (n,ℓ)-sequence, presented by SUP, (E,F), $\sigma_1,\sigma_2,\dots,\sigma_\ell$, and σ, where $\ell>0$ and $e_1>e_2>\dots>e_{2^\ell}$ are the eq-levels of the elements of Seq in the respective order. We must obviously have $n>0$.

Since $\text{EQLV}(E\sigma,F\sigma)=e_1>e_2=\text{EQLV}(E\sigma_1\sigma,F\sigma_1\sigma)$, we can fix some $x_i\in$ SUP and $H\neq x_i$, where $E\xrightarrow{w}x_i$ and $F\xrightarrow{w}H$, or vice versa, for a witness w for (E,F). (This follows from Prop. 5(2).) We put

$$H'=H\{(x_i,H)\}\{(x_i,H)\}\cdots,\ \text{and}\ \sigma'_j=\sigma_j\,\{(x_i,H')\}\ (\text{for all } j\in[1,\ell]).$$

We note that $\sigma'_j\overline{\sigma}=\sigma'_j\overline{\sigma}_{[-x_i]}$ for any $\overline{\sigma}$ (where $\overline{\sigma}_{[-x_i]}$ arises from $\overline{\sigma}$ by putting $\overline{\sigma}_{[-x_i]}(x_i)=x_i$). We say that the above (n,ℓ)-*presentation* of Seq (accompanied) *with x_i,H gives rise* to the $(n{-}1,\ell{-}1)$-sequence Seq' presented by

$$\textsc{Sup} \setminus \{x_i\}, (E\sigma'_1, F\sigma'_1), (\sigma'_2)_{[-x_i]}, (\sigma'_3)_{[-x_i]}, \ldots, (\sigma'_\ell)_{[-x_i]}, \sigma_{[-x_i]}. \tag{13}$$

We now show that Seq' satisfies the desired condition, i.e., the eq-levels of its elements are $e_2 > e_4 > e_6 > \cdots > e_{2^\ell}$. We prove this by induction on ℓ.

By Cor. 7 we deduce that $\textsc{EqLv}\,(\, E\sigma'_1\sigma_{[-x_i]}, F\sigma'_1\sigma_{[-x_i]}\,) = e_2$, since

$$\textsc{EqLv}\,(\, E\sigma_1\sigma, F\sigma_1\sigma\,) = \textsc{EqLv}\,(\, E\sigma_1\{(x_i, H')\}\sigma_{[-x_i]}, F\sigma_1\{(x_i, H')\}\sigma_{[-x_i]}\,).$$

If $\ell = 1$, then we are done.

If $\ell > 1$, then the $(n, \ell{-}1)$-presentation $\textsc{Sup}$, (E, F), $\sigma_1, \sigma_2, \ldots, \sigma_{\ell-1}$, σ (presenting the first half of Seq) with x_i, H gives rise to the $(n{-}1, \ell{-}2)$-sequence Seq'_1 that is the first half of Seq'; hence Seq'_1 is presented by $\textsc{Sup} \setminus \{x_i\}$, $(E\sigma'_1, F\sigma'_1)$, $(\sigma'_2)_{[-x_i]}$, $(\sigma'_3)_{[-x_i]}$, …, $(\sigma'_{\ell-1})_{[-x_i]}$, $\sigma_{[-x_i]}$. By the induction hypothesis, the eq-levels in Seq'_1 are $e_2, e_4, \ldots, e_{2^{\ell-1}}$.

The $(n, \ell{-}1)$-presentation $\textsc{Sup}$, (E, F), $\sigma_1, \sigma_2, \ldots, \sigma_{\ell-1}$, $\sigma_\ell\sigma$ (presenting the second half of Seq) with x_i, H gives rise to the $(n{-}1, \ell{-}2)$-sequence Seq''_2 presented by $\textsc{Sup} \setminus \{x_i\}$, $(E\sigma'_1, F\sigma'_1)$, $(\sigma'_2)_{[-x_i]}$, $(\sigma'_3)_{[-x_i]}$, …, $(\sigma'_{\ell-1})_{[-x_i]}$, $(\sigma_\ell\sigma)_{[-x_i]}$. By the induction hypothesis, the eq-levels in Seq''_2 are $e_{2^{\ell-1}+2}, e_{2^{\ell-1}+4}, \ldots, e_{2^\ell}$.

By another (repeated) use of Cor. 7, the eq-levels in Seq''_2 do not change when we replace $(\sigma_\ell\sigma)_{[-x_i]}$ with $(\sigma_\ell\{(x_i, H')\}\sigma_{[-x_i]})_{[-x_i]} = (\sigma'_\ell)_{[-x_i]}\sigma_{[-x_i]}$ in the presentation. By this replacing we get the sequence Seq'_2 that is the second half of Seq', in fact. The proof is thus finished. □

3.3 Formalizing Informal Parts of the Proof of Theorem 2

In this section we just add more formal details to some parts of the proof in Section 3.2 where the reasoning might look too informal. We assume a given det-FO grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$.

Sink Words, Normal-Form Grammars, Small Number M_0
For every $A \in \mathcal{N}$ and $i \in [1, m]$ where $m = \mathit{arity}(A)$ we say that $w \in \Sigma^*$ is an (A, i)-*sink word* if $A(x_1, \ldots, x_m) \xrightarrow{w} x_i$. We can compute a shortest (A, i)-sink word $w_{[A,i]}$ for each (A, i) for which such a word exists. The lengths of some $w_{[A,i]}$ can be exponential in $\mathit{size}(\mathcal{G})$, but we can compute these lengths (and concise presentations of $w_{[A,i]}$) by a polynomial algorithm based on dynamic programming. The essential fact is that the length of a shortest (A, i)-sink word is $1{+}|v|$ where v is a shortest word such that $E \xrightarrow{v} x_i$ for the right-hand side E of a rule $A(x_1, \ldots, x_m) \xrightarrow{a} E$ in $\mathcal{R}$. If $E = x_i$ then $v = \varepsilon$; otherwise v can be composed from the (A, i)-sink words corresponding to a respective branch in the syntactic tree of E.

We assume that the *grammar* $\mathcal{G}$ is *in* the *normal form*, i.e., there is $w_{[A,i]}$ for each (A, i). This is harmless: if there is no such word for a concrete pair A, i, then we can decrease the arity of A and modify the rules in $\mathcal{R}$ accordingly, while the LTS $\mathcal{L}_\mathcal{G}$ remains unchanged, in fact.

It is now also quite clear that M_0 defined by (2) is indeed a small number.

Closeness Predicate $\models$ Defined via $\models_B$, $\models_R$, $\models_L$
We recall the small number TRANSINC (the maximal size of the rhs of a rule in $\mathcal{R}$) that also bounds the term-height increase caused by performing one transition, and we define another small number:

$$M_1 = \text{TRANSINC} \cdot (M_0)^2 + 2 \cdot M_0.$$

The points 1-4 below define (a technically convenient form of) the predicate $W \models (T, U)$, divided into three (not necessarily disjoint) cases $\models_B$ ("both sides reachable"), $\models_R$ ("right-hand side reachable"), and $\models_L$ ("left-hand side reachable"). We say that a term T is *k-reachable* from W if $W \xrightarrow{w} T$ where $|w| \leq k$.

1) $W \models_B (T, U)$ if each of T, U is M_1-reachable from W.
2) $W \models_R (T, U)$ if U is M_0-reachable from W and $T = G\sigma$ where $\text{HEIGHT}(G) \leq M_0 \cdot \text{TRANSINC}$ and $\sigma(x_i)$ is M_0-reachable from W for each x_i in G.
3) $W \models_L (T, U)$ if T is M_0-reachable from W and $U = G\sigma$ where $\text{HEIGHT}(G) \leq M_0 \cdot \text{TRANSINC}$ and $\sigma(x_i)$ is M_0-reachable from W for each x_i in G.
4) $W \models (T, U)$ if $W \models_B (T, U)$ or $W \models_L (T, U)$ or $W \models_R (T, U)$.

We can easily verify a claim from Section 3.2: if $W \models (T, U)$, then $W = H\sigma$, $T = E\sigma$, $U = F\sigma$ for some small finite terms H, E, F, where $\text{RANGE}(\sigma)$ contains only subterms of W occurring in small depths in W.

Relation $\rightsquigarrow$ on the Set of Quadruples $((W, T, U), u)$
Now we define the relation $\rightsquigarrow$ (used in (4)), by the following (deduction) rules divided into the cases a), b), c). In fact, formally we introduce the relations $\overset{\text{BAL}}{\rightsquigarrow}$ ("balance") and $\overset{\text{POST}}{\rightsquigarrow}$ ("postpone") where $\rightsquigarrow = \overset{\text{BAL}}{\rightsquigarrow} \cup \overset{\text{POST}}{\rightsquigarrow} \cdot \overset{\text{BAL}}{\rightsquigarrow}$. We assume that

$$W \models (T, U) \text{ and } u \text{ is a witness for } (T, U),$$

and we describe $((W', T', U'), u')$ such that $((W, T, U), u) \overset{\text{BAL}}{\rightsquigarrow} ((W', T', U'), u')$ or $((W, T, U), u) \overset{\text{POST}}{\rightsquigarrow} ((W', T', U'), u')$; at the same time we verify that $W' \models (T', U')$ and that u' is a proper suffix of u and a witness for (T', U').

a) $W \models_B (T, U)$.
Suppose that u has the *shortest* prefix w such that one of the paths $T \xrightarrow{w}$, $U \xrightarrow{w}$ finishes by a non-sink segment.
   i) Suppose $w = u_1 u_2$ where $T \xrightarrow{u_1} T_1 \xrightarrow{u_2} T_2$ and $T_1 \xrightarrow{u_2} T_2$ is a non-sink segment; we thus have $|u_2| = M_0$, $T_1 = A(V_1, \ldots, V_m)$, and $T_2 = G\sigma$ where $A(x_1, \ldots, x_m) \xrightarrow{u_2} G$ and $\sigma(x_i) = V_i$ (for $i \in [1, m]$).
   Let $U \xrightarrow{u_1} U_1 \xrightarrow{u_2} U_2$ and $u = u_1 u_2 u'$. Then we deduce
   $$((W, T, U), u) \overset{\text{BAL}}{\rightsquigarrow} ((U_1, G\sigma', U_2), u')$$
   putting $\sigma'(x_i) = V_i'$ where $U_1 \xrightarrow{w_{(A,i)}} V_i'$.
   We can verify that $U_1 \models_R (G\sigma', U_2)$. Moreover, u' is a witness for $(G\sigma', U_2)$ (by using Prop. 6(1)).
   ii) If $w = u_1 u_2$ where $U \xrightarrow{u_1} U_1 \xrightarrow{u_2} U_2$ and $U_1 \xrightarrow{u_2} U_2$ is a non-sink segment, then we proceed symmetrically and deduce

$$((W,T,U),u) \overset{\text{BAL}}{\rightsquigarrow} ((T_1,T_2,G\sigma'),u').$$

Here $T_1 \models_L (T_2, G\sigma')$, and u' is a witness for $(T_2, G\sigma')$.

b) $W \not\models_B (T,U)$ and $W \models_R (T,U)$.
Hence $W \models_R (G\sigma,U)$ where $G\sigma$ is a presentation of T in the form of the definition of $\models_R$ (in the point 2); thus $\text{HEIGHT}(G) \leq M_0 \cdot \text{TRANSINC}$, and G is not a variable since $W \not\models_B (T,U)$. Suppose now that u has the shortest prefix w such that one of the following two conditions holds:
i) $G \xrightarrow{w} x_i$ (hence $G\sigma \xrightarrow{w} \sigma(x_i)$ where $\sigma(x_i)$ is M_0-reachable from W), or
ii) $w = u_1u_2$ where $G \xrightarrow{u_1} G_1 \xrightarrow{u_2} G_2$ and $G_1 \xrightarrow{u_2} G_2$ is a non-sink segment.
We note that if there is no such w, then u is short. Here we assume $u = wu'$.
In the case i), $G \xrightarrow{w} x_i$, we take U' such that $U \xrightarrow{w} U'$ and we deduce

$$((W,G\sigma,U),u) \overset{\text{POST}}{\rightsquigarrow} ((W,\sigma(x_i),U'),u').$$

We note that $W \models_B (\sigma(x_i),U')$: since $|w| \leq (1 + M_0 \cdot \text{TRANSINC}) \cdot M_0$ and U is M_0-reachable from W, we have that U' is M_1-reachable from W; and $\sigma(x_i)$ is even M_0-reachable from W.
In the case ii) we proceed as in a), i.e., for $U \xrightarrow{u_1} U_1 \xrightarrow{u_2} U_2$ we deduce

$$((W,G\sigma,U),u) \overset{\text{BAL}}{\rightsquigarrow} ((U_1,G'\sigma',U_2),u')$$

accordingly (where $G_1\sigma = A'(x_1,\ldots,x_{m'})\sigma'' \xrightarrow{u_2} G'\sigma'' = G_2\sigma$ is the non-sink segment). Here $U_1 \models_R (G'\sigma',U_2)$ and u' is a witness for $(G'\sigma',U_2)$.

c) $W \not\models_B (T,U)$ and $W \models_L (T,G\sigma)$.
This case is analogous to b). We can here deduce

$$((W,T,G\sigma),u) \overset{\text{POST}}{\rightsquigarrow} ((W,T',\sigma(x_i)),u')$$

where $W \models_B (T',\sigma(x_i))$ and u' is a witness for $(T',\sigma(x_i))$, or

$$((W,T,G\sigma),u) \overset{\text{BAL}}{\rightsquigarrow} ((T_1,T_2,G'\sigma'),u')$$

where $T_1 \models_L (T_2,G'\sigma')$ and u' is a witness for $(T_2,G'\sigma')$.

We recall that $\rightsquigarrow = \overset{\text{BAL}}{\rightsquigarrow} \cup \overset{\text{POST}}{\rightsquigarrow} \cdot \overset{\text{BAL}}{\rightsquigarrow}$, and we note that $\overset{\text{POST}}{\rightsquigarrow} \cdot \overset{\text{POST}}{\rightsquigarrow}$ is empty.

We can easily verify the next claims:

1. If $((W,T,U),u) \overset{\text{BAL}}{\rightsquigarrow} ((W',T',U'),u')$ by a), then W' is M_0-reachable from a subterm of a term (T or U) that is M_1-reachable from W.
2. If $((W,T,U),u) \overset{\text{POST}}{\rightsquigarrow} ((W',T',U'),u')$ then $W' = W$.
3. If $((W,T,U),u) \overset{\text{BAL}}{\rightsquigarrow} ((W',T',U'),u')$ by b) or c), then W' is M_1-reachable from W.

Hence $((W,T,U),u) \rightsquigarrow ((W',T',U'),u')$ implies that $W \xrightarrow{v_1} W_1' \xrightarrow{v_2} W_2' \xrightarrow{v_3} W'$ for some v_1, v_2, v_3 and W_1', W_2' where $|v_1| \leq M_1$, $|v_3| \leq M_0$, and $W_1' \xrightarrow{v_2} W_2'$ sinks from W_1' to its subterm W_2': we take v_2 as the sequence of the appropriate (A,i)-sink words $w_{[A,i]}$ (along the respective branch in the syntactic tree of W_1').

We fix such a path $W \xrightarrow{w} W'$, where $w = v_1v_2v_3$, for each $((W,T,U),u) \rightsquigarrow ((W',T',U'),u')$. It is obvious that $W \xrightarrow{w} W'$ contains at most a small number of non-sink segments.

We have thus formalized deriving a pivot-path (5) from the sequence (4).

4 Bisimilarity of FO Grammars Is Ackermann-Hard

Here we prove the next theorem, referring to the notions discussed in Section 2.4.

Theorem 8. *Bisimilarity of first-order grammars is Ackermann-hard.*

We do not give a direct reduction from HP-Ack, since using Lemma 9 below is much more convenient. We define the necessary notions first.

Reset Counter Machines (RCMs). An *RCM* is a tuple $\mathcal{M} = (d, Q, \delta)$ where d is the *dimension*, yielding d nonnegative *counters* $c_1, c_2, \ldots, c_d$, Q is a finite set of *(control) states*, and $\delta \subseteq Q \times \mathrm{Op} \times Q$ is a finite set of *instructions*, where the set Op of *operations* contains $\textsc{inc}(c_i)$ (increment c_i), $\textsc{dec}(c_i)$ (decrement c_i), and $\textsc{reset}(c_i)$ (set c_i to 0), for $i = 1, 2, \ldots, d$. We view $Q \times \mathbb{N}^d$ as the set Conf of *configurations* of $\mathcal{M}$. The *transition relation* $\longrightarrow \subseteq \mathrm{Conf} \times \mathrm{Conf}$ is induced by δ in the obvious way: If $(p, op, q) \in \delta$ then we have $(p, (n_1, \ldots, n_d)) \longrightarrow (q, (n'_1, \ldots, n'_d))$ in the following cases:

- $op = \textsc{inc}(c_i)$, $n'_i = n_i + 1$, and $n'_j = n_j$ for all $j \neq i$; or
- $op = \textsc{dec}(c_i)$, $n_i > 0$, $n'_i = n_i - 1$, and $n'_j = n_j$ for all $j \neq i$; or
- $op = \textsc{reset}(c_i)$, $n'_i = 0$, and $n'_j = n_j$ for all $j \neq i$.

By $\longrightarrow^*$ we denote the reflexive and transitive closure of $\longrightarrow$.

Control-State Reachability Problem for RCMs
We define the *RCM control-state reachability problem* in the following form: given an RCM $\mathcal{M} = (d, Q, \delta)$ and (control) states p_{IN}, p_{F}, we ask if p_{F} is reachable from $(p_{\text{IN}}, (0, 0, \ldots, 0))$, i.e., if there are $m_1, m_2, \ldots, m_d \in \mathbb{N}$ such that $(p_{\text{IN}}, (0, 0, \ldots, 0)) \longrightarrow^* (p_{\text{F}}, (m_1, m_2, \ldots, m_d))$.

Lemma 9. *[20] RCM control-state reachability problem is Ackermann-hard.*

We mentioned this problem, and its ACK-completeness, in Section 1. (The crux of the hardness proof is an efficient construction that, given $n \in \mathbb{N}$, provides RCMs $\mathcal{M}_1$, $\mathcal{M}_2$ that "weakly compute" the function f_n and its inverse, for f_n used in the definition of the Ackermann function. By "weakly" we mean that some computations can also return smaller values than expected.)

RCM Control-State Reachability Reduces to First-Order Bisimilarity
We finish by proving the next lemma; this establishes Theorem 8, by using Lemma 9. The given reduction is obviously polynomial. (In fact, it can be checked to be a logspace reduction, but this is a minor point in the view of the fact that even a primitive-recursive reduction would suffice here.)

Lemma 10. *The RCM control-state reachability problem is polynomially reducible to the complement of the bisimilarity problem for first-order grammars.*

Proof. Let us consider an instance $\mathcal{M} = (d, Q, \delta)$, p_{IN}, p_{F} of the RCM control-state reachability problem, and imagine the following game between Attacker (he) and Defender (she). This is the first version of a game that will be afterwards

implemented as a standard bisimulation game. Attacker aims to show that p_{F} is reachable from $(p_{\text{IN}}, (0, 0, \ldots, 0))$, while Defender opposes this.

The game uses $2d$ *game-counters*, which are never decremented; each counter c_i of $\mathcal{M}$ yields two game-counters, namely c_i^I and c_i^D, for counting the numbers of Increments and Decrements of c_i, respectively, since the last reset or since the beginning if there has been no reset of c_i so far. The *initial position* is $(p_{\text{IN}}, ((0, 0), \ldots, (0, 0)))$, with all $2d$ game-counters (organized in pairs) having the value 0.

A game *round* from position $(p, ((n_1, n_1'), \ldots, (n_d, n_d')))$ proceeds as described below. It will become clear that it suffices to consider only the cases $n_i \geq n_i'$; the position then corresponds to the $\mathcal{M}$'s configuration $(p, (n_1 - n_1', \ldots, n_d - n_d'))$.

If $p = p_{\text{F}}$, then Attacker wins; if $p \neq p_{\text{F}}$ and there is no instruction $(p, op, q) \in \delta$, then Defender wins. Otherwise Attacker chooses $(p, op, q) \in \delta$, and the continuation depends on op as follows:

1. If $op = \text{INC}(c_i)$, then the next-round position arises (from the previous one) by replacing p with q and by performing $c_i^I := c_i^I + 1$ (the counter of increments of c_i is incremented, i.e., n_i is replaced with $n_i + 1$).
2. If $op = \text{RESET}(c_i)$, then the next-round position arises by replacing p with q and by performing $c_i^I := 0$ and $c_i^D := 0$ (hence both n_i and n_i' are replaced with 0).
3. If $op = \text{DEC}(c_i)$, then *Defender chooses* one of the following options:
   (a) the next-round position arises by replacing p with q and by performing $c_i^D := c_i^D + 1$ (the counter of decrements of c_i is incremented, i.e., n_i' is replaced with $n_i' + 1$), or
   (b) (Defender claims that this *decrement* is *illegal* since $n_i = n_i'$ and) the next position becomes just (n_i, n_i'). In this case a (deterministic) check if $n_i = n_i'$ is performed, by successive synchronized decrements at both sides. If indeed $n_i = n_i'$ (the counter-bottoms are reached at the same moment), then Defender wins; otherwise (when $n_i \neq n_i'$) Attacker wins.

If $(p_{\text{IN}}, (0, 0, \ldots, 0)) \longrightarrow^* (p_{\text{F}}, (m_1, m_2, \ldots, m_d))$ for some $m_1, m_2, \ldots, m_d$, i.e., if the answer to RCM control-state reachability is YES, then Attacker has a winning strategy: he just follows the corresponding sequence of instructions. He thus also always chooses $\text{DEC}(c_i)$ *legally*, i.e. only in the cases where $n_i > n_i'$, and Defender loses if she ever chooses 3(b). If the answer is NO (p_{F} is not reachable), and Attacker follows a legal sequence of instructions, then he either loses in a "dead" state or the play is infinite; if Attacker chooses an illegal decrement, then in the first such situation we obviously have $n_i = n_i'$ for the respective counter c_i, and Defender can force her win via 3(b).

Since the game-counters can be only incremented or reset, it is a routine to implement the above game as a bisimulation game in the grammar-framework (using a standard method of "Defender's forcing" for implementing the choice in 3). We now describe the corresponding grammar $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R})$.

The set $\mathcal{N}$ of nonterminals will include a unary nonterminal I, a nullary nonterminal $\perp$, and the nonterminals with arity $2d$ that are induced by control

states of $\mathcal{M}$ as follows: each $p \in Q$ induces $A_p, A_{(p,i)}, B_p, B_{(p,i,1)}, B_{(p,i,2)}$, where $i = 1, 2, \ldots, d$.

We intend that a game-position $(p, ((n_1, n'_1), \ldots, (n_d, n'_d)))$ corresponds to the pair of terms

$$\Big(A_p(I^{n_1}\bot, I^{n'_1}\bot, \ldots, I^{n_d}\bot, I^{n'_d}\bot),\ B_p(I^{n_1}\bot, I^{n'_1}\bot, \ldots, I^{n_d}\bot, I^{n'_d}\bot)\Big) \qquad (14)$$

where $I^k\bot$ is a shorthand for $I(I(\ldots I(\bot)\ldots))$ with I occurring k times; we put $I^0\bot = \bot$. The RCM control-state reachability instance $\mathcal{M}, p_{\text{IN}}, p_{\text{F}}$ will be reduced to the (non)bisimilarity-problem instance $\mathcal{G}$, $A_{p_{\text{IN}}}(\bot, \ldots, \bot)$, $B_{p_{\text{IN}}}(\bot, \ldots, \bot)$.

We put $\Sigma = \delta \uplus \{a, b\}$, i.e., the actions of $\mathcal{G}$ correspond to the instructions (or instruction names) of $\mathcal{M}$, and we also use auxiliary (fresh) actions a, b.

The set of rules $\mathcal{R}$ contains a sole rule for I, namely $I(x_1) \stackrel{a}{\longrightarrow} x_1$, and no rule for $\bot$; hence $I^n\bot \sim I^{n'}\bot$ iff $n = n'$. Each instruction $\text{INS} = (p, op, q) \in \delta$ induces the rules in $\mathcal{R}$ as follows:

1. If $op = \text{INC}(c_i)$, then the induced rules are
   $A_p(x_1, \ldots, x_{2d}) \stackrel{\text{INS}}{\longrightarrow} A_q(x_1, \ldots, x_{2(i-1)}, I(x_{2i-1}), x_{2i}, \ldots, x_{2d})$, and
   $B_p(x_1, \ldots, x_{2d}) \stackrel{\text{INS}}{\longrightarrow} B_q(x_1, \ldots, x_{2(i-1)}, I(x_{2i-1}), x_{2i}, \ldots, x_{2d})$.
2. If $op = \text{RESET}(c_i)$, then the induced rules are
   $A_p(x_1, \ldots, x_{2d}) \stackrel{\text{INS}}{\longrightarrow} A_q(x_1, \ldots, x_{2(i-1)}, \bot, \bot, x_{2i+1}, \ldots, x_{2d})$,
   $B_p(x_1, \ldots, x_{2d}) \stackrel{\text{INS}}{\longrightarrow} B_q(x_1, \ldots, x_{2(i-1)}, \bot, \bot, x_{2i+1}, \ldots, x_{2d})$.
3. If $op = \text{DEC}(c_i)$, then the induced rules are below; here we use the shorthand $A \stackrel{a}{\longrightarrow} B$ when meaning $A(x_1, \ldots, x_{2d}) \stackrel{a}{\longrightarrow} B(x_1, \ldots, x_{2d})$:
   $A_p \stackrel{\text{INS}}{\longrightarrow} A_{(q,i)}$, $A_p \stackrel{\text{INS}}{\longrightarrow} B_{(q,i,1)}$, $A_p \stackrel{\text{INS}}{\longrightarrow} B_{(q,i,2)}$, $B_p \stackrel{\text{INS}}{\longrightarrow} B_{(q,i,1)}$, $B_p \stackrel{\text{INS}}{\longrightarrow} B_{(q,i,2)}$,
   $A_{(q,i)}(x_1, \ldots, x_{2d}) \stackrel{a}{\longrightarrow} A_q(x_1, \ldots, x_{2i-1}, I(x_{2i}), x_{2i+1}, \ldots, x_{2d})$,
   $B_{(q,i,1)}(x_1, \ldots, x_{2d}) \stackrel{a}{\longrightarrow} B_q(x_1, \ldots, x_{2i-1}, I(x_{2i}), x_{2i+1}, \ldots, x_{2d})$,
   $B_{(q,i,2)}(x_1, \ldots, x_{2d}) \stackrel{a}{\longrightarrow} A_q(x_1, \ldots, x_{2i-1}, I(x_{2i}), x_{2i+1}, \ldots, x_{2d})$,
   $A_{(q,i)}(x_1, \ldots, x_{2d}) \stackrel{b}{\longrightarrow} x_{2i-1}$, $B_{(q,i,1)}(x_1, \ldots, x_{2d}) \stackrel{b}{\longrightarrow} x_{2i-1}$,
   $B_{(q,i,2)}(x_1, \ldots, x_{2d}) \stackrel{b}{\longrightarrow} x_{2i}$.

Moreover, $\mathcal{R}$ contains $A_{p_{\text{F}}}(x_1, \ldots, x_{2d}) \stackrel{a}{\longrightarrow} \bot$ (but *not* $B_{p_{\text{F}}}(x_1, \ldots, x_{2d}) \stackrel{a}{\longrightarrow} \bot$).

Now we recall the standard (turn-based) *bisimulation game*, starting with the pair $(A_{p_{\text{IN}}}(\bot, \ldots, \bot), B_{p_{\text{IN}}}(\bot, \ldots, \bot))$. In the round starting with (T_1, T_2), Attacker chooses a transition $T_j \stackrel{a}{\longrightarrow} T'_j$ and then Defender chooses $T_{3-j} \stackrel{a}{\longrightarrow} T'_{3-j}$ (for the same $a \in \Sigma$); the next round starts with the pair (T'_1, T'_2). If a player gets stuck, then (s)he loses; an infinite play is a win of Defender. It is obvious that Defender has a winning strategy in this game iff $A_{p_{\text{IN}}}(\bot, \ldots, \bot) \sim B_{p_{\text{IN}}}(\bot, \ldots, \bot)$.

We now easily check that this bisimulation game indeed implements the above described game; a game-position $(p, ((n_1, n'_1), \ldots, (n_d, n'_d)))$ is implemented as the pair (14). The points 1 and 2 directly correspond to the previous points 1 and 2. If Attacker chooses an instruction $\text{INS} = (p, \text{DEC}(c_i), q)$, then he must

use the respective rule $A_p \xrightarrow{\text{Ins}} A_{(q,i)}$ in 3, since otherwise Defender installs syntactic equality, i.e. a pair (T, T). It is now Defender who chooses $B_p \xrightarrow{\text{Ins}} B_{(q,i,1)}$ (corresponding to the previous 3(a)) or $B_p \xrightarrow{\text{Ins}} B_{(q,i,2)}$ (corresponding to 3(b)). Attacker then must choose action a in the first case, and action b in the second case; otherwise we get syntactic equality. The first case thus results in the pair $(A_q(\dots), B_q(\dots))$ corresponding to the next game-position (where c_i^D has been incremented), and the second case results in the pair $(I^{n_i}\bot, I^{n_i'}\bot)$; we have already observed that $I^{n_i}\bot \sim I^{n_i'}\bot$ iff $n_i = n_i'$.

Finally we observe that in any pair $(A_{p_F}(\dots), B_{p_F}(\dots))$ Attacker wins immediately (since the transition $A_{p_F}(\dots) \xrightarrow{a} \bot$ can not be matched).

We have thus established that p_F is reachable from $(p_{IN}, (0, \dots, 0))$ if, and only if, $A_{p_{IN}}(\bot, \dots, \bot) \not\sim B_{p_{IN}}(\bot, \dots, \bot)$. □

Acknowledgement. The Ackermann-hardness result was achieved during my visit at LSV ENS Cachan, and I am grateful to Sylvain Schmitz and Philippe Schnoebelen for fruitful discussions.

References

1. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: Proc. STOC 2004, pp. 202–211. ACM (2004)
2. Benedikt, M., Göller, S., Kiefer, S., Murawski, A.S.: Bisimilarity of pushdown automata is nonelementary. In: Proc. LICS 2013, pp. 488–498. IEEE Computer Society (2013)
3. Böhm, S., Göller, S., Jančar, P.: Equivalence of deterministic one-counter automata is NL-complete. In: Proc. STOC 2013, pp. 131–140. ACM (2013)
4. Böhm, S., Göller, S., Jančar, P.: Bisimulation equivalence and regularity for real-time one-counter automata. J. Comput. Syst. Sci. 80, 720–743 (2014), http://dx.doi.org/10.1016/j.jcss.2013.11.003
5. Broadbent, C.H., Göller, S.: On bisimilarity of higher-order pushdown automata: Undecidability at order two. In: FSTTCS 2012. LIPIcs, vol. 18, pp. 160–172. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2012)
6. Burkart, O., Caucal, D., Steffen, B.: An elementary bisimulation decision procedure for arbitrary context-free processes. In: Hájek, P., Wiedermann, J. (eds.) MFCS 1995. LNCS, vol. 969, pp. 423–433. Springer, Heidelberg (1995)
7. Courcelle, B.: Recursive applicative program schemes. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, vol. B, pp. 459–492. Elsevier, MIT Press (1990)
8. Czerwinski, W., Lasota, S.: Fast equivalence-checking for normed context-free processes. In: Proc. of FSTTCS 2010. LIPIcs, vol. 8, pp. 260–271. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2010)
9. Figueira, D., Figueira, S., Schmitz, S., Schnoebelen, P.: Ackermannian and primitive-recursive bounds with Dickson's lemma. In: Proc. LICS 2011, pp. 269–278. IEEE Computer Society (2011)
10. Friedman, E.P.: The inclusion problem for simple languages. Theor. Comput. Sci. 1(4), 297–316 (1976)

11. Göller, S., Lohrey, M.: The First-Order Theory of Ground Tree Rewrite Graphs. Logical Methods in Computer Science 10(1) (2014), http://dx.doi.org/10.2168/LMCS-10(1:7)2014
12. Henry, P., Sénizergues, G.: LALBLC A program testing the equivalence of dpda's. In: Konstantinidis, S. (ed.) CIAA 2013. LNCS, vol. 7982, pp. 169–180. Springer, Heidelberg (2013)
13. Hirshfeld, Y., Jerrum, M., Moller, F.: A polynomial algorithm for deciding bisimilarity of normed context-free processes. Theor. Comput. Sci. 158(1&2), 143–159 (1996)
14. Jančar, P., Srba, J.: Undecidability of bisimilarity by Defender's forcing. J. ACM 55(1) (2008)
15. Jančar, P.: Decidability of DPDA language equivalence via first-order grammars. In: Proc. LICS 2012, pp. 415–424. IEEE Computer Society (2012)
16. Jančar, P.: Bisimilarity on basic process algebra is in 2-ExpTime (an explicit proof). Logical Methods in Computer Science 9(1) (2013)
17. Kiefer, S.: BPA bisimilarity is EXPTIME-hard. Inf. Process. Lett. 113(4), 101–106 (2013)
18. Kučera, A., Mayr, R.: On the complexity of checking semantic equivalences between pushdown processes and finite-state processes. Inf. Comput. 208(7), 772–796 (2010)
19. Schmitz, S.: Complexity hierarchies beyond elementary. CoRR abs/1312.5686 (2013)
20. Schnoebelen, P.: Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 616–628. Springer, Heidelberg (2010)
21. Sénizergues, G.: L(A)=L(B)? Decidability results from complete formal systems. Theoretical Computer Science 251(1-2), 1–166 (2001); a preliminary version appeared at Degano, P., Gorrieri, R., Marchetti-Spaccamela, A. (eds.): ICALP 1997. LNCS, vol. 1256. Springer, Heidelberg (1997)
22. Sénizergues, G.: The bisimulation problem for equational graphs of finite outdegree. SIAM J. Comput. 34(5), 1025–1106 (2005) (a preliminary version appeared at FOCS 1998)
23. Sénizergues, G.: The equivalence problem for t-turn DPDA is co-NP. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 478–489. Springer, Heidelberg (2003)
24. Srba, J.: Roadmap of infinite results. in: Current Trends In Theoretical Computer Science, The Challenge of the New Century, vol. 2, pp. 337–350. World Scientific Publishing Co. (2004), updated version at http://users-cs.au.dk/srba/roadmap/
25. Srba, J.: Beyond language equivalence on visibly pushdown automata. Logical Methods in Computer Science 5(1) (2009)
26. Stirling, C.: Deciding DPDA equivalence is primitive recursive. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 821–832. Springer, Heidelberg (2002)
27. Stirling, C.: Second-order simple grammars. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 509–523. Springer, Heidelberg (2006)
28. Urquhart, A.: The complexity of decision procedures in relevance logic II. J. Symb. Log. 64(4), 1774–1802 (1999)
29. Walukiewicz, I.: Pushdown processes: Games and model-checking. Inf. Comput. 164(2), 234–263 (2001)