Decidability of DPDA Language Equivalence via First-Order Grammars

Petr Jančar
Techn. Univ. Ostrava, Czech Republic (www.cs.vsb.cz/jancar)

***Abstract*—The decidability of language equivalence of deterministic pushdown automata (DPDA) was established by G. Sénizergues (1997, 2001), who thus solved a famous long-standing open problem. A simplified proof, also providing a primitive recursive complexity upper bound, was given by C. Stirling (2002). In this paper, the decidability is re-proved in the framework of first-order terms and grammars (given by finite sets of root-rewriting rules). The proof is based on the abstract ideas used in the previous proofs, but the chosen framework seems to be more natural for the problem and allows a short presentation which should be transparent for a general computer science audience. The approach can be also easily adapted to provide the mentioned complexity bound, and to extend the decidability to bisimulation equivalence of (nondeterministic) pushdown automata, i.e. to show a result which was also established by G. Sénizergues (1998, 2005).**

I. Introduction

Language equivalence of deterministic pushdown automata (DPDA) is a famous problem in language theory. The decidability question for this problem was posed in the 1960s [7], then a series of works solving various subcases followed, until the question was answered positively by Sénizergues in 1997, with the full journal version [12]. G. Sénizergues was awarded the Gödel prize in 2002 for this significant achievement.

Later Stirling [16] and also Sénizergues [13] provided simpler proofs than the original proof. A modified version, showing also a (nonelementary) primitive recursive complexity upper bound, appeared as a conference paper by Stirling in 2002 [17]; Sénizergues showed a “more reasonable” upper bound for a subclass in [15]. Sénizergues also generalised the decidability result to bisimulation equivalence over a class of nondeterministic pushdown automata [14]. Unfortunately, even the simpler proofs seem rather long and technical, which does not ease further research regarding, e.g., the complexity. (No nontrivial lower bound for DPDA-equivalence seems to be known; the general bisimilarity problem is ExpTime-hard [9].)

The algorithms are based on the following key points. If two configurations are nonequivalent then there is a shortest word witnessing this fact, an *sw-word* for short. If two configurations are equivalent then any attempt to (stepwise) build a potential sw-word can be contradicted: an (algorithmically verifiable) proof of a contradiction is produced after a (sufficiently long) prefix of the potential sw-word has been constructed.

This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), and by the Czech Grant Agency project GAČR:P202/11/0340.

One reason why the DPDA problem turned out so intricate seems to be the lack of structure of configurations (strings of symbols), which calls for a richer framework. This is also discussed by Stirling [16] who refers to the algebraic theory of linear combinations of boolean rational series built by Sénizergues, and replaces it by a process calculus whose processes are derived from determinising strict grammars. Another difficulty is that providing a proof of equivalence for a given pair of states (i.e. structured objects representing configurations) seems to require some measures and conditional rules in the respective logical systems, to keep their soundness. Stirling used (simpler) bisimulation approximants instead of Sénizergues’s system of weights.

We can view the transformations T_B, T_C in [12] as two main means for contradicting that a particular word is a prefix of an sw-word; Stirling [16] uses the rules BAL and CUT. BAL (“balancing”) aims at making the (structured) states in the pairs along the supposed sw-word close to each other, i.e. having “bounded (different) heads” and the same (maybe large) “tails.” CUT aims at “cutting away” large tails soundly.

Here we re-prove the decidability of DPDA language equivalence in the framework of (deterministic) first-order grammars, i.e. systems of first-order terms with finitely many root-rewriting rules. The framework of *regular terms* (possibly infinite terms with only finitely many different subterms), completed *with substitutions*, seems to be a more natural and simpler substitute of the algebraic structures in the previous works. In fact, close relations between the frameworks of (D)PDA, strict deterministic grammars and first-order schemes were recognized long ago (see, e.g., references in [5] and [12]).

The overall strategy of the presented proof is not new, it is close to [16] in particular, but the proof is no direct translation of a previous proof into another framework. The previous logical systems are replaced with a *Prover-Refuter game* whose soundness is obvious. The CUT rule is replaced with a notion of a *finite basis*, generating the equivalence relation in a certain sense. The chosen framework and the new ingredients allow to present a direct, short, and easily understandable proof. Moreover, the presentation is tailored so that the complexity result [17] and the generalization to bisimilarity [14] can be added smoothly. (See http://arxiv.org/abs/1010.4760 for complexity; the author plans a different arxiv-paper on bisimilarity.)

The related research is rich and active. There are comprehensive references in Sénizergues’s and Stirling’s papers to the prior research; for recent research on related complexity

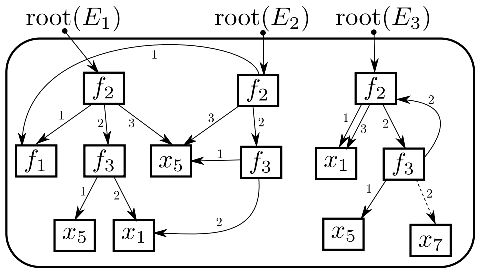

Fig. 1. Graph presentations of terms

questions and on higher-order schemes, we can refer, e.g., to [2], [3], [6], [8], [10], [11] and the references therein.

The (sub)section titles show the structure of the paper. Section II sets "the stage." Section III describes the (obviously sound) Prover-Refuter game. Section IV shows that Prover has a strategy which is complete (enables to prove equivalence for all equivalent pairs). Hence searching for an sw-word and for an equivalence proof in parallel gives a desired algorithm.

II. Basic Notions and Simple Observations

$\mathbb{N}$ denotes the set $\{0, 1, 2, \dots\}$ of natural numbers. For a set $\mathcal{A}$, by $\text{CARD}(\mathcal{A})$ we denote its cardinality (i.e. the number of elements when $\mathcal{A}$ is finite). $\mathcal{A}^*$ denotes the set of finite sequences of elements of $\mathcal{A}$, 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. The *empty sequence* is denoted ε (thus $|\varepsilon| = 0$).

First-Order Regular Terms and Substitutions

We recall the standard notion of *first-order terms*, assuming a fixed countable set $\text{VAR} = \{x_1, x_2, x_3, \dots\}$ of *variables*. Given a set $\mathcal{F} = \{f_1, f_2, \dots, f_k\}$ of *function symbols* where $arity(f)$ denotes the arity of $f \in \mathcal{F}$, an example of a *term over* $\mathcal{F}$ is $E_1 = f_2(f_1, f_3(x_5, x_1), x_5)$, assuming $k \geq 3$ and $arity(f_1) = 0$, $arity(f_2) = 3$, $arity(f_3) = 2$. We can recognize the syntactic tree of E_1 in the left part of Fig. 1.

Formally we view *terms as* the *partial mappings*

$$E : \mathbb{N}^* \to \mathcal{F} \cup \text{VAR}$$

where $\gamma\, i$ ($\gamma \in \mathbb{N}^*$, $i \in \mathbb{N}$) belongs to $\text{DOM}(E)$ iff $\gamma \in \text{DOM}(E)$ and $1 \leq i \leq arity(E(\gamma))$; we stipulate $arity(x_j) = 0$ for all $x_j \in \text{VAR}$. The expressions like $f_2(f_1, f_3(x_5, x_1), x_5)$ or x_{17} are thus viewed as representing partial mappings $\mathbb{N}^* \to \mathcal{F} \cup \text{VAR}$ (e.g., $E_1(\varepsilon) = f_2$, $E_1(\langle 2, 1\rangle) = x_5$, etc.).

Given a term E and $\gamma \in \text{DOM}(E)$, *the term* E_γ where $\text{DOM}(E_\gamma) = \{\delta \mid \gamma\delta \in \text{DOM}(E)\}$ and $E_\gamma(\delta) = E(\gamma\delta)$ is a *subterm* of E, *occurring in* E *at* γ, *at depth* $|\gamma|$. A *term* E is *finite* if $\text{DOM}(E)$ is finite. A *term* is *regular* if it has only finitely many subterms.

A *graph presentation* GP (of some *regular* terms) is a *finite graph* whose nodes are labelled with elements of $\mathcal{F} \cup \text{VAR}$. Each node labelled with f has m outgoing arcs labelled with $1, 2, \dots, m$ where $m = arity(f)$; the nodes labelled with variables x_i (and the nodes labelled with nullary function symbols) have no outgoing arcs. A term E is presented by a graph GP when a node n is specified as the *root* of E; we refer to such a graph as to GP_E. A term can have more than one presentation. E.g., E_1, E_2 presented in Fig. 1 are obviously the same terms; E_3 in Fig. 1 (where we now ignore the dotted arc leading to x_7) is an infinite regular term.

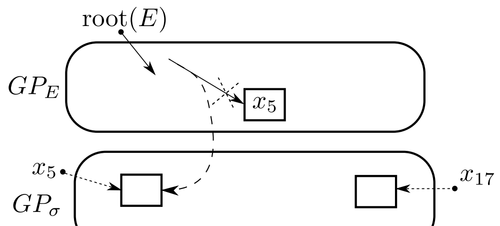

Fig. 2. Creating $\text{GP}_{E\sigma}$ from GP_E and GP_σ

Convention. By "terms" we further mean "regular terms" (i.e. those having finite graph presentations). We do not consider the empty term with the empty domain; we might use a special nullary (function) symbol $\perp$ instead.

We note that the equality of two (regular) terms can be efficiently checked by standard partition-refinement techniques: Given a graph presentation GP, for each node n of GP we consider the (sub)term rooted in n. We first partition all these terms according to the root-labels, and then we are refining the partition according to the (current) partition-classes of root-successors, until the stable partition (the fixpoint) is found.

By $\text{TERMS}_\mathcal{F}$ we denote the set of all (regular) terms over $\mathcal{F}$. A *substitution* σ is a mapping $\sigma : \text{VAR} \to \text{TERMS}_\mathcal{F}$ whose *support* $\text{SUPP}(\sigma) = \{x_i \mid \sigma(x_i) \neq x_i\}$ is *finite*. The finite-support restriction allows us to present any substitution σ by a finite graph GP where each $x_i \in \text{SUPP}(\sigma)$ has an associated node in GP, namely the root of $\sigma(x_i)$; we refer to GP_σ then. For a term E and a substitution σ, we define $E\sigma$ (the *term resulting from* E *by applying* σ) as expected: $\gamma \in \text{DOM}(E\sigma)$ iff either $\gamma \in \text{DOM}(E)$ and $E(\gamma) \notin \text{VAR}$, in which case $(E\sigma)(\gamma) = E(\gamma)$, or $\gamma = \gamma_1\gamma_2$, $\gamma_1 \in \text{DOM}(E)$, $E(\gamma_1) = x_j$, $\gamma_2 \in \text{DOM}(\sigma(x_j))$, in which case $(E\sigma)(\gamma) = (\sigma(x_j))(\gamma_2)$.

Fig. 2 illustrates how to get a presentation of $E\sigma$ from presentations GP_E, GP_σ of E and σ, respectively: each arc leading to (a node labelled with) $x_i \in \text{SUPP}(\sigma)$ in GP_E is redirected to the node associated with x_i in GP_σ. Note that if $E = x_i$ then $E\sigma = x_i\sigma = \sigma(x_i)$.

By $E\sigma_1\sigma_2$ we mean $(E\sigma_1)\sigma_2$, but we also define the *composition of substitutions* $\sigma_1 \circ \sigma_2$, denoted just $\sigma_1\sigma_2$: for $\sigma = \sigma_1\sigma_2$ and $x_i \in \text{VAR}$ we have $\sigma(x_i) = (\sigma_1(x_i))\,\sigma_2$; thus $\text{SUPP}(\sigma_1\sigma_2) \subseteq \text{SUPP}(\sigma_1) \cup \text{SUPP}(\sigma_2)$. We can easily check $E\sigma_1\sigma_2 = (E\sigma_1)\sigma_2 = E\sigma$ where $\sigma = \sigma_1\sigma_2$; more generally, the *composition* is *associative*, i.e., $(\sigma_1\sigma_2)\sigma_3 = \sigma_1(\sigma_2\sigma_3)$.

Finally we note that E_3 in Fig. 1 can be viewed as arising from (a finite term) $E = f_2(x_1, f_3(x_5, x_7), x_1)$ (represented like E_3 but using *the dotted arc* in Fig. 1) by applying the substitution $\sigma' = \{(x_7, E)\}$, i.e. σ' where $\text{SUPP}(\sigma') = \{x_7\}$ and $\sigma'(x_7) = E$, repeatedly forever; hence $E_3 = E\sigma'\sigma'\sigma' \dots$. (To get GP_{E_3} from GP_E, each arc leading to x_7 is redirected to the root.) Note that the auxiliary variable x_7 could be replaced with any x_i not occuring in E_3.

Later we also refer to the *presentation size* $\text{PRESSIZE}(E)$, by which we mean the *size* of the *smallest graph presentation*

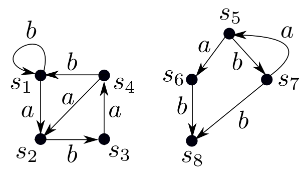

Fig. 3. A (finite) deterministic labelled transition system

of E; similarly for $\textsc{PresSize}(\sigma)$. We can use any natural notion of *size* which takes also the indices of variables into account; e.g., we can take the number of nodes and arcs plus the bit-size of all labels. We thus have only finitely many terms E with $\textsc{PresSize}(E) \leq b$, for any given bound $b \in \mathbb{N}$. Another natural property which we assume is $\textsc{PresSize}(E\sigma) \leq \textsc{PresSize}(E) + \textsc{PresSize}(\sigma)$.

Labelled Transition Systems and Trace Equivalence

A *labelled transition system*, an *LTS* for short, is a tuple $\mathcal{L} = (\mathcal{S}, \mathcal{A}, (\xrightarrow{a})_{a\in\mathcal{A}})$ where $\mathcal{S}$ is the set of *states*, $\mathcal{A}$ the set of *actions* and $\xrightarrow{a} \subseteq \mathcal{S} \times \mathcal{S}$ is the set of *transitions labelled with* $a \in \mathcal{A}$, called a*-transitions*. Fig. 3 shows a finite LTS (in fact, a det-LTS as defined later). The relations $\xrightarrow{w} \subseteq \mathcal{S} \times \mathcal{S}$ for $w \in \mathcal{A}^*$ are defined as expected: $s \xrightarrow{\varepsilon} s$; if $s \xrightarrow{a} s'$ and $s' \xrightarrow{u} s''$ then $s \xrightarrow{au} s''$. In Fig. 3 we have, e.g., $s_1 \xrightarrow{bab} s_3$.

By writing $s \xrightarrow{w}$ we mean that s *enables* (a trace) $w \in \mathcal{A}^*$, i.e., $s \xrightarrow{w} s'$ for some s'. *Trace equivalence* $\sim$ on $\mathcal{S}$, and its "strata" $\sim_0$, $\sim_1$, $\sim_2$, ..., are defined as follows:

$$s \sim t \text{ if } \forall w \in \mathcal{A}^* : s \xrightarrow{w} \Leftrightarrow t \xrightarrow{w},$$
$$\text{and for } k \in \mathbb{N}\text{: } s \sim_k t \text{ if } \forall w \in \mathcal{A}^{\leq k} : s \xrightarrow{w} \Leftrightarrow t \xrightarrow{w},$$

where $\mathcal{A}^{\leq k} = \{w \in \mathcal{A}^* \mid |w| \leq k\}$.

Fact 1. *(1)* $\sim$ *and all* $\sim_k$ *are equivalence relations. (2)* $\sim_0 = \mathcal{S} \times \mathcal{S}$*. (3)* $\sim_0 \supseteq \sim_1 \supseteq \sim_2 \supseteq \cdots$*. (4)* $\cap_{k\in\mathbb{N}} \sim_k = \sim$*.*

This (trivial) fact suggests to define the *equivalence-level* (*eq-level*) for each pair of states:

$$\textsc{EqLv}(s,t) = k \ (k \in \mathbb{N}) \text{ if } s \sim_k t \text{ and } s \not\sim_{k+1} t\,;$$
$$\textsc{EqLv}(s,t) = \omega \text{ if } s \sim t, \text{ also written as } s \sim_\omega t.$$

In Fig. 3 we have, e.g., $\textsc{EqLv}(s_1, s_2) = 0$, $\textsc{EqLv}(s_1, s_5) = 2$, $\textsc{EqLv}(s_1, s_4) = \omega$. We take ω as an infinite number satisfying $n < \omega$ and $\omega - n = \omega + n = \omega$ for any $n \in \mathbb{N}$.

The next fact (for $k \in \mathbb{N}$) will be particularly useful.

Proposition 2. *If* $\textsc{EqLv}(s,t) = k$ *and* $\textsc{EqLv}(s,s') \geq k+1$ *then* $\textsc{EqLv}(s',t) = k$ *(since* $s' \sim_k s \sim_k t$ *and* $s' \sim_{k+1} s \not\sim_{k+1} t$*).*

Of special interest for us are *deterministic* LTSs, *det-LTSs* for short: $\mathcal{L} = (\mathcal{S}, \mathcal{A}, (\xrightarrow{a})_{a\in\mathcal{A}})$ is deterministic if for each $s \in \mathcal{S}$ and each $a \in \mathcal{A}$ there is at most one s' such that $s \xrightarrow{a} s'$. (Fig. 3 depicts a finite det-LTS.)

Fact 3. *In any det-LTS, if* $w = a_1 a_2 \dots a_k$ *and* $s \xrightarrow{w}$ *then there is a unique* path $s \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} s_k$*.*

For det-LTSs we easily observe that by performing the same action $a \in \mathcal{A}$ from s, t the eq-level can drop by at most one, and it does drop for some action when $\omega > \textsc{EqLv}(s,t) > 0$:

$$r_1 : Ax_1 \xrightarrow{a} ABx_1 \qquad r_2 : Ax_1 \xrightarrow{b} x_1$$
$$r_3 : Bx_1 \xrightarrow{a} BAx_1 \qquad r_4 : Bx_1 \xrightarrow{b} x_1$$

Fig. 4. A det-first-order grammar $\mathcal{G} = (\{A, B\}, \{a, b\}, \{r_1, r_2, r_3, r_4\})$

Proposition 4. *Given a* deterministic *LTS:*
(1) If $s \xrightarrow{w} s'$, $t \xrightarrow{w} t'$ *then* $\textsc{EqLv}(s',t') \geq \textsc{EqLv}(s,t) - |w|$*.*
(2) If $\textsc{EqLv}(s,t) = k \in \mathbb{N}$ *then there is* $w = a_1 a_2 \dots a_k$ *such that* $s \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \cdots \xrightarrow{a_k} s_k$ *and* $t \xrightarrow{a_1} t_1 \xrightarrow{a_2} t_2 \xrightarrow{a_3} \cdots \xrightarrow{a_k} t_k$ *where* $\textsc{EqLv}(s_j, t_j) = k - j$ *for* $j = 1, 2, \dots, k$*.*

In Point (2) we have $s_k \not\sim_1 t_k$, hence there is $a \in \mathcal{A}$ such that $s_k \xrightarrow{a}$, $\neg(t_k \xrightarrow{a})$ or vice versa; the word wa is then a *shortest nonequivalence-witness word* for the pair s, t.

(Det-)First-Order Grammars as Generators of (Det-)LTSs

We now introduce LTSs whose *states* are not "black dots" as in Fig. 3 but *(regular) terms*; transitions $E_1 \xrightarrow{a} E_2$ will be determined by a finite set of root-rewriting rules.

Definition 5. *A* first-order grammar *is a tuple* $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ *where* $\mathcal{N}$ *is a finite set of ranked* nonterminals*, i.e. (function) symbols with arities,* $\mathcal{A}$ *is a finite set of* actions *(or terminals), and* $\mathcal{R}$ *is a finite set of* (root rewriting) rules r *of the form*

$$r : Yx_1x_2 \dots x_m \xrightarrow{a} E \qquad (1)$$

where $Y \in \mathcal{N}$*,* $arity(Y) = m$*,* $a \in \mathcal{A}$*, and* E *is a* finite *term over* $\mathcal{N}$ *in which each occurring variable is from the set* $\{x_1, x_2, \dots, x_m\}$*. (*$E = x_i$*, where* $1 \leq i \leq m$*, is an example.) We put* $\textsc{act}(r) = a$*, thus defining the mapping* $\textsc{act} : \mathcal{R} \to \mathcal{A}$*.* $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ *is* deterministic*, a* det-first-order grammar*, if there is at most one rule (1) for each pair* $Y \in \mathcal{N}$*,* $a \in \mathcal{A}$*.*

Remark on notation. In the previous (classical term) notation, the rules would be written $r : f(x_1, x_2, \dots, x_m) \xrightarrow{a} E$. Now $\mathcal{N}$ plays the role of former $\mathcal{F}$; we use Y to range over $\mathcal{N}$, and we omit parentheses. We might also use A, B for nonterminals, but E, F, G, H and T, U, V, W will always range over $\textsc{Terms}_\mathcal{N}$ (using our fixed $\textsc{Var} = \{x_1, x_2, \dots\}$). $YG_1G_2 \dots G_m$, $(Ax_1x_2x_3)\sigma = A\,\sigma(x_1)\sigma(x_2)\sigma(x_3)$, $F'\sigma_1\sigma_2$ are examples of the notation which we use for terms (where σ's are substitutions). We consider $\bot$ as a special nullary nonterminal, with no rules; we use it in the example in Fig. 9.

Fig. 4 shows an example of a det-first-order grammar $\mathcal{G}$. This $\mathcal{G}$ is, in fact, very simple, we have $arity(Y) = 1$ for all $Y \in \mathcal{N}$ and the rules are thus of the form $Yx_1 \xrightarrow{a} Y_1Y_2 \dots Y_\ell\, x_1$. (A more general example will be illustrated in Fig. 7.) Our example grammar is thus, in fact, a context-free grammar in Greibach normal form, with no special starting symbol and with only left derivations allowed, as the next definition shows (due to using the rules as *root-rewriting*). In fact, the definition takes all *(regular) terms as states*, though we allowed only *finite right-hand sides* (rhs) E *in rules* (1) for technical convenience.

Definition 6. *A grammar* $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ *generates (the* rule based*) LTS* $\mathcal{L}^{\mathrm{R}}_\mathcal{G} = (\textsc{Terms}_\mathcal{N}, \mathcal{R}, (\xrightarrow{r})_{r\in\mathcal{R}})$*: for each rule* $r : Yx_1x_2 \dots x_m \xrightarrow{a} E$ *(recall (1)) we have*

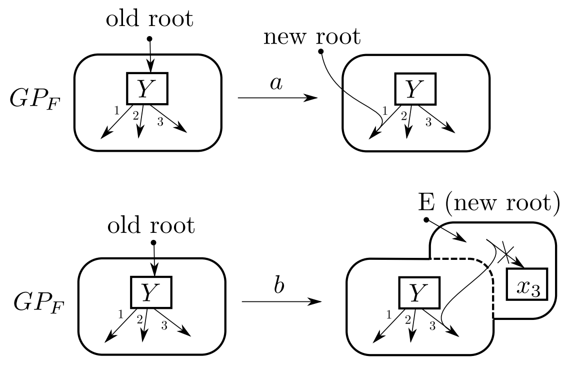

Fig. 5. Applying rules $Yx_1x_2x_3 \xrightarrow{a} x_1$ and $Yx_1x_2x_3 \xrightarrow{b} E$ to GP_F

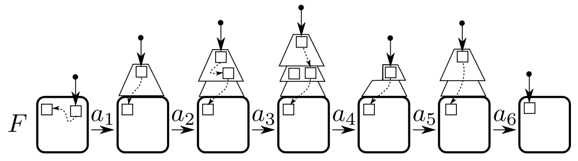

Fig. 6. A path in $\mathcal{L}^{\text{A}}_{\mathcal{G}}$

$F \xrightarrow{r} H$ *if there is a substitution* σ *such that* $F = (Yx_1 \dots x_m)\sigma$ *and* $H = E\sigma$.

(Note that σ *with* $\text{SUPP}(\sigma) = \emptyset$ *yields* $Yx_1 \dots x_m \xrightarrow{r} E$.*)* *For (the* action-based*) LTS* $\mathcal{L}^{\text{A}}_{\mathcal{G}} = (\text{TERMS}_{\mathcal{N}}, \mathcal{A}', (\xrightarrow{a})_{a\in\mathcal{A}'})$ *we define* $\mathcal{A}' = \mathcal{A} \cup \{a_{x_i} \mid x_i \in \text{VAR}\}$ *where* a_{x_i} *is a unique (fresh) action attached to* x_i. *For* $a \in \mathcal{A}'$ *we have* $F \xrightarrow{a} H$ *if* $F \xrightarrow{r} H$ *for some* $r \in \mathcal{R}$ *with* $\text{ACT}(r) = a$ *or if* $F = H = x_i$ *and* $a = a_{x_i}$.

Remark and convention. In $\mathcal{L}^{\text{R}}_{\mathcal{G}}$ the variables x_i are examples of *dead terms* (not enabling any transition), like the term $\perp$. In $\mathcal{L}^{\text{A}}_{\mathcal{G}}$ we have $x_i \xrightarrow{a_{x_i}} x_i$ but we never use these special transitions in our reasoning; we only use the consequence that $x_i \not\sim_1 H$ if $H \neq x_i$ (in particular if $H = x_j$ for $j \neq i$).

Fact 7. *$\mathcal{L}^{\text{R}}_{\mathcal{G}}$ is a det-LTS for any $\mathcal{G}$.*
$\mathcal{L}^{\text{A}}_{\mathcal{G}}$ is a det-LTS iff $\mathcal{G}$ is deterministic.

Fig. 5 shows how the rules can be applied to graph presentations. To apply $r : Yx_1x_2x_3 \xrightarrow{b} E$ to GP_F, we first verify that the root of F is (labelled with) Y. Then we add GP_E (the rhs of r) to GP_F (we "stack GP_E on top of GP_F"), the root of E becomes the new root, and every arc leading to x_i in GP_E is redirected to the i-th successor of the root of F. If $E = x_j$ then the result is that the j-th successor of the (old) root in GP_F becomes the new root (it can be the old root in case of a loop). Fig. 6 depicts a path in $\mathcal{L}^{\text{A}}_{\mathcal{G}}$. We note that even if we successively "stack" many (finite) rhs $E_1, E_2, \dots$ of used rules (or rather subterms of rhs), there can be always root-successors lying "deeply down," even in the initial (regular) term F. Note that the current root is connected to any future root which lies in the current graph.

The next fact holds in both $\mathcal{L}^{\text{R}}_{\mathcal{G}}$ and $\mathcal{L}^{\text{A}}_{\mathcal{G}}$. (Recall Fig. 2).

Fact 8. *If* $E \xrightarrow{w} F$ *then* $E\sigma \xrightarrow{w} F\sigma$; *hence if* $E \xrightarrow{w} x_i$ *then* $E\sigma \xrightarrow{w} \sigma(x_i)$. *If* $E\sigma \xrightarrow{w}$ *but* $\neg(E \xrightarrow{w})$ *then* $w = uv$ *where* $E \xrightarrow{u} x_i$ *for some* $x_i \in \text{VAR}$ *and* $E\sigma \xrightarrow{u} \sigma(x_i) \xrightarrow{v}$.

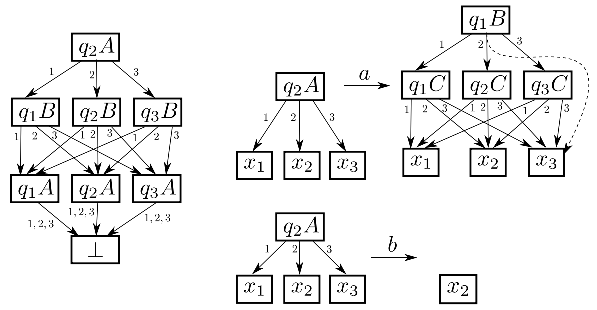

Fig. 7. Term representations of configurations and rules of (D)PDA

Convention. We further refer to $\mathcal{L}^{\text{A}}_{\mathcal{G}}$, if not said otherwise. Hence by writing $E \xrightarrow{w} F$ we mean $w \in \mathcal{A}^*$.

We are interested in the following problem.

Problem TRACE-EQ-DET-G
Input: a det-first-order grammar $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$, and (graph presentations of) two input terms T_{in}, U_{in}.
Question: is $T_{in} \sim U_{in}$ in $\mathcal{L}^{\text{A}}_{\mathcal{G}}$?

(D)PDA from a First-Order Term Perspective

The next lemma could be derived from the papers referred to in [5]; we sketch a direct concise proof, to be self-contained.

Lemma 9. *The DPDA language equivalence problem is (polynomial-time) reducible to* TRACE-EQ-DET-G.

We view a *pushdown automaton* (*PDA*) as a tuple $\mathcal{M} = (Q, \Gamma, \mathcal{A}, \Delta)$ of finite sets of *control states*, *stack symbols*, *actions* (also called *input letters*), and (rewriting) *rules*, respectively. The *term-representation* $\mathcal{T}(q_2ABA)$ *of* the *configuration* $q_2ABA \in Q \times \Gamma^*$, assuming $Q = \{q_1, q_2, q_3\}$, is on the left in Fig. 7; we put $\mathcal{T}(q\varepsilon) = \perp$ and $\mathcal{T}(qA\alpha) = [qA]\mathcal{T}(q_1\alpha)\dots\mathcal{T}(q_k\alpha)$ when $Q = \{q_1, \dots, q_k\}$. So $Q \times \Gamma$ is the set of nonterminals with arity $\text{CARD}(Q)$; $\perp$ is a special "bottom" nullary nonterminal. On the right in Fig. 7 we can see the *term-representations of* two *rules* (from $Q \times \Gamma \times \mathcal{A} \times Q \times \Gamma^*$), one *pushing*, $q_2A \xrightarrow{a} q_1BC$, and one *popping*, $q_2A \xrightarrow{b} q_2$; we ignore the dotted arc for the moment. The term-representation of $qA \xrightarrow{a} q'\beta$ is $\mathcal{T}(qAx) \xrightarrow{a} \mathcal{T}(q'\beta x)$ when we add $\mathcal{T}(q_ix) = x_i$. In PDA semantics, a rule $qA \xrightarrow{a} q'\beta$ implies $qA\alpha \xrightarrow{a} q'\beta\alpha$ for any $\alpha \in \Gamma^*$. We can easily check that this corresponds to the first-order grammar semantics ($qA\alpha \xrightarrow{a} q'\beta\alpha$ iff $\mathcal{T}(qA\alpha) \xrightarrow{a} \mathcal{T}(q'\beta\alpha)$).

We have so far ignored the possible *ε-rules* like $q_2C \xrightarrow{\varepsilon} q_3$; it is standard to assume, w.l.o.g., that all ε-rules are *popping*. A *PDA* is *deterministic*, a *DPDA*, if any rule $qC \xrightarrow{\varepsilon}$ .. excludes the existence of another rule $qC \xrightarrow{a}$ .. for any $a \in \mathcal{A} \cup \{\varepsilon\}$, and there is at most one rule $qA \xrightarrow{a}$ .. for any triple q, A, a; a pair qA is called *unstable* if there is a rule $qA \xrightarrow{\varepsilon}$ .., otherwise qA is *stable*. Def. 5 does not allow to translate a rule $q_2C \xrightarrow{\varepsilon} q_3$ to $[q_2C]x_1x_2x_3 \xrightarrow{\varepsilon} x_3$, but we can adjust the above definition of $\mathcal{T}(q\alpha)$ by putting $\mathcal{T}(qC\beta) = \mathcal{T}(q'\beta)$ when $qC \xrightarrow{\varepsilon} q'$ is a rule. The dotted arc in Fig. 7 illustrates this "ε-contraction" if we have $q_2C \xrightarrow{\varepsilon} q_3$. For a DPDA $\mathcal{M}$ we obviously get: $q\alpha \xrightarrow{w}$ iff $\mathcal{T}(q\alpha) \xrightarrow{w}$ for any $w \in \mathcal{A}^*$.

To prove Lemma 9, we use the following language equivalence problem (w.l.o.g.): given a DPDA $\mathcal{M} = (Q, \Gamma, \mathcal{A}, \Delta)$ and configurations $\mathcal{C}, \mathcal{C}'$, decide if $L(\mathcal{C}) = L(\mathcal{C}')$ where $L(q\alpha) = \{w \in \mathcal{A}^* \mid \exists q' : q\alpha \xrightarrow{w} q'\varepsilon\}$. For any stable pair qA and any $a \in \mathcal{A}$, if there is no rule $qA \xrightarrow{a}$ .. then we add the rule $qA \xrightarrow{a} q_{loop}A$; q_{loop} is a new state with the rules $q_{loop}A \xrightarrow{a} q_{loop}A$ for all $a \in \mathcal{A}$, $A \in \Gamma$. This modification does not affect $L(\mathcal{C})$, $L(\mathcal{C}')$. It is now easy to verify that $L(\mathcal{C}) = L(\mathcal{C}')$ iff $\mathcal{T}(\mathcal{C}) \sim \mathcal{T}(\mathcal{C}')$, assuming the above described transformation of configurations and rules.

Semidecidability of Trace Non-Equivalence

Given $\mathcal{G}$ and a pair $E \not\sim F$, we can find a shortest word witnessing the nonequivalence of E, F by a systematic search. Hence the next lemma is obvious even in the general case, though we now concentrate on the deterministic case.

Lemma 10. *There is an algorithm with the following property: it (halts and) computes* $\text{EqLv}(T_{in}, U_{in})$ *for an instance* $\mathcal{G}, T_{in}, U_{in}$ *of* TRACE-EQ-DET-G *iff* $T_{in} \not\sim U_{in}$ *in* $\mathcal{L}^{\mathrm{A}}_{\mathcal{G}}$.
Thus the complement of TRACE-EQ-DET-G *is semidecidable.*

III. An Algorithm Deciding Trace-Eq-Det-G

We aim to show the semidecidability of TRACE-EQ-DET-G, which will yield the decidability by Lemma 10. III-A shows some simple facts about the equivalences $\sim_k$ and $\sim$, and III-B introduces further technical prerequisites for the Prover-Refuter game (played for an instance $\mathcal{G}$, T_{in}, U_{in}) described in III-C. In III-D we will easily observe the *soundness* of the P-R game, which means that Prover has no winning strategy if $T_{in} \not\sim U_{in}$. It will be also obvious that there is an algorithm which halts for $\mathcal{G}$, T_{in}, U_{in} iff Prover has a winning strategy. Hence the decidability of TRACE-EQ-DET-G will be established once we show the *completeness*, i.e. the existence of a winning strategy of Prover for every $T_{in} \sim U_{in}$; this is done in Sec. IV.

Convention. If not said otherwise, we assume a given det-first-order grammar $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ and refer to the det-LTS $\mathcal{L}^{\mathrm{A}}_{\mathcal{G}}$. (Recall that Fact 3 applies here.) By referring to a *path* $G \xrightarrow{w}$ (or $G \xrightarrow{w} G'$) we mean that w is enabled by G and we also refer to the unique sequence $G \xrightarrow{a_1} G_1 \xrightarrow{a_2} G_2 \xrightarrow{a_3} \cdots \xrightarrow{a_k} G_k$ ($G_k = G'$) where $w = a_1 a_2 \dots a_k$.

A. Some Properties of $\sim_k$ and $\sim$ (in the Det-LTS $\mathcal{L}^{\mathrm{A}}_{\mathcal{G}}$)

For two substitutions $\sigma, \sigma' : \text{Var} \to \text{Terms}_{\mathcal{N}}$ we define

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

The *congruence properties* in Prop. 11 are obvious, by recalling Fact 8 (and Fig. 2, 5, 6).

Proposition 11. *(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 $\text{EqLv}(\sigma, \sigma') \leq \text{EqLv}(E\sigma, E\sigma')$.

Prop. 12 completes Point (1); it follows from the next observation. If $\text{EqLv}(E, F) = k \in \mathbb{N}$ then there is w, $|w| = k$, such that $E \xrightarrow{w} E'$, $F \xrightarrow{w} F'$ where $E' \not\sim_1 F'$. If the roots of E', F' are nonterminals enabling different sets of actions then $\text{EqLv}(E\sigma, F\sigma) = k$ for any σ; another option is that $\{E', F'\} = \{x_i, H\}$ where $H \neq x_i$.

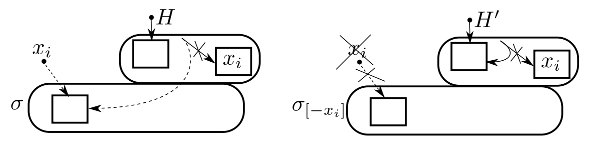

Fig. 8. $H\sigma$ and $H'\sigma_{[-x_i]} = (H\{(x_i, H)\}\{(x_i, H)\}\{(x_i, H)\}\cdots)\sigma$

Proposition 12. *If* $\text{EqLv}(E, F) = k < \ell = \text{EqLv}(E\sigma, F\sigma)$ *($\ell \in \mathbb{N} \cup \{\omega\}$) then there are some $x_i \in$ SUPP(σ), $H \neq x_i$, and a word w, $|w| = k$, such that $E \xrightarrow{w} x_i$, $F \xrightarrow{w} H$ or $E \xrightarrow{w} H$, $F \xrightarrow{w} x_i$; moreover, $\sigma(x_i) \sim_{\ell-k} H\sigma$ (by Prop. 4(1)).*

The next proposition (sketched in Fig. 8) is later useful for decreasing the support of a substitution in an inductive argument (in Fig. 13). We define $\sigma_{[-x_i]}$ as the substitution arising from σ by *removing x_i from the support* (if it is there):

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

(We now might take $E_3 = E\{(x_7, E)\}\{(x_7, E)\}\dots$ in Fig. 1 as an example of $H' = H\sigma'\sigma'\dots$ in Fig. 8.)

Proposition 13. *Assume $H \neq x_i$ and $H' = H\sigma'\sigma'\cdots$ where $\sigma' = \{(x_i, H)\}$. (GP$_{H'}$ arises from GP$_H$ by redirecting all incoming arcs of x_i to the root of H; hence $H' = H$ if x_i does not occur in H, in particular if $H = x_j$, $j \neq i$.)*
If $\sigma(x_i) \sim_k H\sigma$ then $\sigma(x_i) \sim_k H'\sigma_{[-x_i]}$ and thus $\sigma \sim_k \{(x_i, H')\}\,\sigma_{[-x_i]}$.

$H'\sigma = H'\sigma_{[-x_i]}$ since x_i does not occur in H'. It thus suffices to show that $\text{EqLv}(\sigma(x_i), H\sigma) = \text{EqLv}(\sigma(x_i), H'\sigma)$. This follows from Prop. 2, once we note that $H\sigma \not\sim H'\sigma$ implies $\text{EqLv}(H\sigma, H'\sigma) > \text{EqLv}(\sigma(x_i), H'\sigma)$. (Fig. 8 makes clear that any nonequivalence witness w for $H\sigma, H'\sigma$ has a nonempty prefix u such that $H\sigma \xrightarrow{u} \sigma(x_i)$, $H'\sigma \xrightarrow{u} H'\sigma$.)

B. k-Distance Regions (for Deciding $T \sim_k U$)

We have implicitly noted (around Lemma 10) that we can decide whether $T \sim_k U$ (for $k \in \mathbb{N}$); a natural way is to construct the *k-distance region for* (T, U):

$$\text{Reg}(T, U, k) = \{\, (T', U') \mid T \xrightarrow{w} T', U \xrightarrow{w} U' \text{ for some } w, |w| \leq k \,\}.$$

Fig. 9 shows the 2-distance region for $(T, U) = (AB\bot, BA\bot)$, assuming our example grammar in Fig. 4.

Note that $T \not\sim_k U$ iff there is $(T', U') \in \text{Reg}(T, U, k{-}1)$ such that $T' \not\sim_1 U'$. We define the *least eq-level* for a set of pairs of terms (for a region $\text{Reg}(T, U, k)$ in particular):

$$\text{for } R \subseteq \text{Terms}_{\mathcal{N}} \times \text{Terms}_{\mathcal{N}},\ R \neq \emptyset, \text{ we define}$$
$$\text{MinEL}(\mathcal{R}) = \min\{\, \text{EqLv}(T', U') \mid (T', U') \in \mathcal{R} \,\}.$$

The next proposition follows from Prop. 4; it says that any least eq-level pair in $\text{Reg}(T, U, k)$ must be in the bottom row in the figures like Fig. 9 or Fig. 10, if $T \not\sim U$ and $T \sim_k U$.

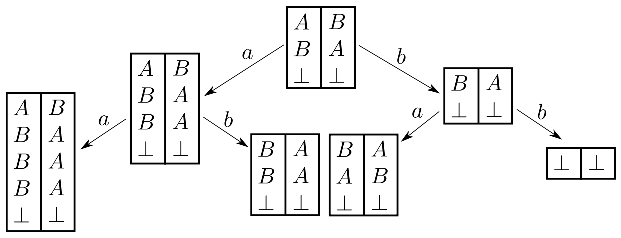
Fig. 9. The 2-distance region $\mathrm{REG}(T,U,2)$ for $(T,U) = (AB\bot, BA\bot)$

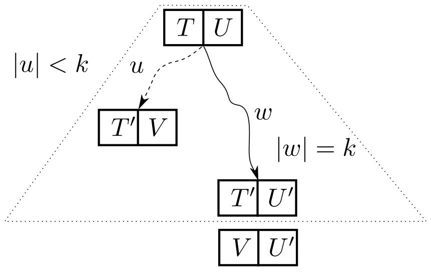

Fig. 10. Case 1 of left-balancing

Proposition 14.
(1.) If $T \sim U$ then $T' \sim U'$ for all $(T',U') \in \mathrm{REG}(T,U,k)$.
(2.) If $T \not\sim U$, $T \sim_k U$ and $(T',U') \in \mathrm{REG}(T,U,k)$ satisfies $\mathrm{EQLV}(T',U') = \mathrm{MINEL}(\mathrm{REG}(T,U,k))$ then $(T',U') \in \mathrm{REG}(T,U,k) \smallsetminus \mathrm{REG}(T,U,k{-}1)$.

By Prop. 14, 11(2) and 2 we easily derive the next proposition. It is useful to look at Fig. 11 (which is fully used later), and imagine $\sigma = \{(x_1,V_1),(x_2,V_2)\}$, $\sigma' = \{(x_1,V_1'),(x_2,V_2')\}$.

Proposition 15. *Suppose that $T \sim_k U$ and for σ,σ' we have $\mathrm{SUPP}(\sigma) = \mathrm{SUPP}(\sigma')$ and $(\sigma(x_i),\sigma'(x_i)) \in \mathrm{REG}(T,U,k{-}1)$ for each $x_i \in \mathrm{SUPP}(\sigma)$.*
If $\mathrm{EQLV}(T',U') = \mathrm{MINEL}(\mathrm{REG}(T,U,k))$ and $T' = G\sigma$ then $\mathrm{EQLV}(G\sigma',U') = \mathrm{EQLV}(T',U')$.

Fig. 10 shows a case with $G = x_1$, $\sigma(x_1) = T'$, $\sigma'(x_1) = V$.

C. Prover-Refuter Game

We describe a game between Prover (she) and Refuter (he). Given an initial pair (T_{in},U_{in}), and finitely many pairs (constituting a "basis") chosen by Prover, Refuter attempts to build a shortest word witnessing that one of the given pairs is non-equivalent; Prover aims to contradict this attempt.

PROVER-REFUTER GAME (P-R GAME)

1) A det-first-order grammar $\mathcal{G} = (\mathcal{N},\mathcal{A},\mathcal{R})$ is given.
2) Prover produces (by "guessing", say) a finite set BASIS of pairs of (graph presentations of regular) terms.
3) An input pair (T_{in},U_{in}) is given.
4) *Refuter* chooses
$(T_0,U_0) \in \mathrm{STARTSET} = \{(T_{in},U_{in})\} \cup \mathrm{BASIS}$,
and *claims* $\mathrm{EQLV}(T_0,U_0) = \mathrm{MINEL}(\mathrm{STARTSET}) < \omega$.
5) For $i = 0,1,2,\ldots$, Phase i is performed, i.e.:
   a) Prover chooses $k > 0$, and $\mathrm{REG}(T_i,U_i,k)$ is constructed; if $T_i \not\sim_k U_i$ then Prover loses (the play ends).
   b) Refuter chooses $(T_i',U_i') \in \mathrm{REG}(T_i,U_i,k) \smallsetminus \mathrm{REG}(T_i,U_i,k{-}1)$ and w_i, $|w_i| = k$, such that $T_i \xrightarrow{w_i} T_i'$, $U_i \xrightarrow{w_i} U_i'$; if there is no such T_i',U_i',w_i (due to dead terms, hence $T_i \sim U_i$), Prover wins. *Refuter claims* that $\mathrm{EQLV}(T_i',U_i') = \mathrm{MINEL}(\mathrm{REG}(T_i,U_i,k))$. (Recall Prop. 14.)
   c) Prover produces (T_{i+1},U_{i+1}) from (T_i',U_i') as follows:
      - either she puts $T_{i+1} = T_i'$, $U_{i+1} = U_i'$ (*no change*),
      - or she *balances* (recall Prop. 15 and Fig. 11): if she finds σ,σ' such that $(\sigma(x_i),\sigma'(x_i)) \in \mathrm{REG}(T,U,k{-}1)$ for all $x_i \in \mathrm{SUPP}(\sigma) = \mathrm{SUPP}(\sigma')$, and she presents T_i' as $G\sigma$ then she can (do a *left-balancing*, namely) put $T_{i+1} = G\sigma'$, and $U_{i+1} = U_i'$; symmetrically, if U_i' is $G\sigma'$ then she can (do a *right-balancing*, namely) put $T_{i+1} = T_i'$, and $U_{i+1} = G\sigma$.

      (Thus $\mathrm{EQLV}(T_{i+1},U_{i+1}) = \mathrm{EQLV}(T_i',U_i')$ if Refuter's claim in 5.b is true. We have $T_{i+1} \sim U_{i+1}$ if $T_i \sim U_i$.)
   d) Prover *either derives a contradiction from Refuter's claims in 4 and 5.b*, by presenting a proof, i.e. a finite algorithmically verifiable sequence of deductions based on Propositions 2, 4, 11, 12, 13, in which case Prover wins, *or lets the play proceed* with Phase $i{+}1$.

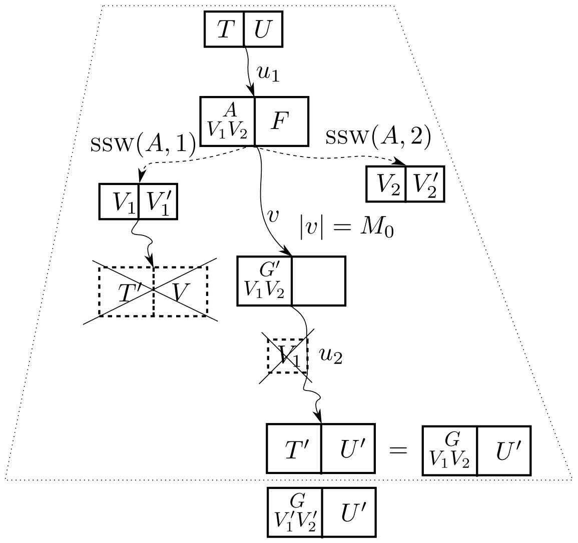

Fig. 11. Case 2 of left-balancing

Fig. 12 (used later) shows an example of Phases i and $i{+}1$.

By switching Points 2) and 3) we get the *weaker form of the game*; a play then starts with a given instance $\mathcal{G}, T_{in}, U_{in}$ of TRACE-EQ-DET-G. We use the above (stronger) form to stress that BASIS is related to the grammar $\mathcal{G}$ (and is independent of T_{in},U_{in}). We note that performing Point 5 in a play gives rise to a (finite or infinite) sequence of pairs

$$(T_1,U_1),(T_2,U_2),(T_3,U_3),\ldots \qquad (2)$$

which is *eq-level decreasing*, by which we mean $\omega > \mathrm{EQLV}(T_1,U_1) > \mathrm{EQLV}(T_2,U_2) > \cdots$, if Refuter's claims are true; we have $T_j \sim U_j$ for all j if $T_0 \sim U_0$.

We can see that BASIS plays no role until possibly used in the final proof contradicting Refuter's claims. E.g., if

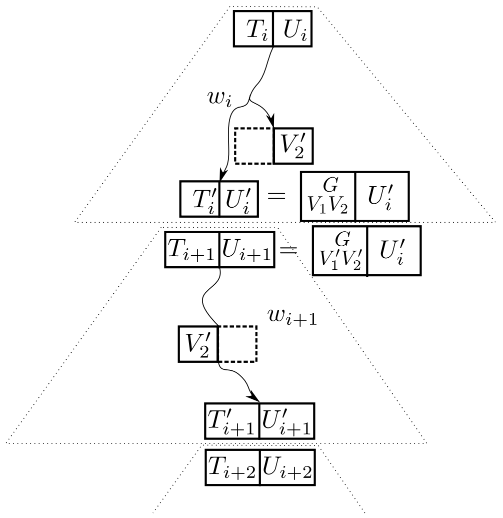

Fig. 12. A left balancing phase i followed by a no-change phase $i{+}1$

(T_i, U_i) for $i > 0$ is shown to be a *basis-instance*, i.e. $(T_i, U_i) = (E\sigma, F\sigma)$ for some $(E, F) \in$ BASIS and some substitution σ, then this is a contradiction, since by Refuter's claims $\text{EQLV}(T_i, U_i) < \text{MINEL}(\text{STARTSET})$ (for $i > 0$) while $\text{EQLV}(E\sigma, F\sigma) \geq \text{EQLV}(E, F) \geq \text{MINEL}(\text{STARTSET})$ (by using Prop. 11(1)). Another simple proof of contradiction is a *repeat*, i.e. getting $(T_j, U_j) = (T_i, U_i)$ for $j > i$.

Remarks. We could make the game more flexible for Prover, adding her other sound possibilities, but the above form suffices for our aims. The name *basis* is inspired by the notion of bisimulation bases in the case of context-free processes; this line of research started with [1] and further developments can be found in [4]. The name reflects the aim to provide a finite set which generates the whole equivalence relation in a certain sense, though this is not formalized here.
As an example of a play of the P-R game, we can assume that $\{(x_1, x_1), (Ax_1, Bx_1)\}$ is chosen as a basis for $\mathcal{G}$ from Fig. 4. If $\text{REG}(T_0, U_0, 2)$ as in Fig. 9 appears in Phase 0 and Refuter chooses $(T_0', U_0') = (ABBB\bot, BAAA\bot)$ then Prover can immediately contradict Refuter's claims: she creates the instance $(AAAA\bot, BAAA\bot)$ of (Ax_1, Bx_1) from (T_0', U_0') by using $(A\bot, B\bot)$, $(AB\bot, BA\bot)$, $(ABB\bot, BAA\bot)$ (with supposedly bigger eq-levels than $\text{EQLV}(T_0', U_0')$) for successive subterm replacements.

D. Soundness of the Prover-Refuter Game

If $\{(T_{in}, U_{in})\} \cup$ BASIS contains a pair of nonequivalent terms then Refuter can be choosing so that his "least eq-level claims" (in 4. and 5.b) are true; then the sequence (2) is eq-level decreasing and Prover loses eventually. This also applies to the weaker form of the P-R game (Points 2 and 3 switched).

Since BASIS is finite and Refuter always has finitely many choices when there is his turn, there is an obvious algorithmic aspect which we also capture in the next (soundness) lemma.

Lemma 16. *There is an algorithm with the following property: given a det-first order grammar $\mathcal{G}$ and T_{in}, U_{in}, it halts iff there is some* BASIS *such that Prover can force her win for $\mathcal{G}$, T_{in}, U_{in} by using* BASIS *(in the weaker form of the game), in which case $T \sim U$ for all $(T, U) \in \{(T_{in}, U_{in})\} \cup$* BASIS.

By combining with Lemma 10 we get an algorithm which decides TRACE-EQ-DET-G, if for each det-first-order grammar $\mathcal{G}$ there exists some BASIS which is sufficient for forcing Prover's win for any $T_{in} \sim U_{in}$. This completeness is shown in Section IV, which will finish a proof of the next theorem; the corollary follows by Lemma 9.

Theorem 17. *Trace equivalence of det-first-order grammars (i.e., the problem* TRACE-EQ-DET-G*) is decidable.*

Corollary 18. *DPDA language equivalence is decidable.*

IV. COMPLETENESS OF THE PROVER-REFUTER GAME

IV-A shows that we get the completeness if there is $n \in \mathbb{N}$, $g : \mathbb{N} \to \mathbb{N}$ for any $\mathcal{G}$ such that Prover has a so-called (n, g)-strategy. IV-B then shows a "balancing strategy" for Prover which turns out to be an (n, g)-strategy.

A. Long (n, g)-Sequences are Sufficient for Prover

We still assume a fixed det-first-order grammar $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ if not said otherwise.
We recall $\text{PRESSIZE}(E)$ (of a regular term E over $\mathcal{N}$), and put $\text{PRESSIZE}(E, F) = \text{PRESSIZE}(E) + \text{PRESSIZE}(F)$, say.

Definition 19.
1) *For $n \in \mathbb{N}$ and a nondecreasing function $g : \mathbb{N} \to \mathbb{N}$, $(T_1, U_1), (T_2, U_2), \ldots$ is an* (n, g)-sequence *if it can be presented as $(E_1\sigma, F_1\sigma)$, $(E_2\sigma, F_2\sigma)$, $\ldots$ where the "heads" satisfy $\text{PRESSIZE}(E_j, F_j) \leq g(j)$ (for $j = 1, 2, \ldots$) and σ satisfies $\text{CARD}(\text{SUPP}(\sigma)) \leq n$.*
2) Prover has an (n, g)-strategy *for $\mathcal{G}$ if she can force that the sequence $(T_1, U_1), (T_2, U_2), (T_3, U_3), \ldots$ arising in the phases $0, 1, 2, \ldots$ (recall (2)) has an infinite subsequence which is an (n, g)-sequence, in each play where $T_0 \sim U_0$ and the play does not finish with Prover's win in Point 5b or with a repeat. (The basis is irrelevant.)*
3) *Stipulating $\max \emptyset = 0$, we define the following finite number (Maximal Finite Equivalence Level) for any $b \in \mathbb{N}$: $\text{MAXFEL}_b = \max\{\, \text{EQLV}(E, F) \mid E \not\sim F$ and $\text{PRESSIZE}(E, F) \leq b \,\}$.*

The essence of the next lemma is the fact that the length of *eq-level decreasing* (n, g)-sequences is bounded by a number depending just on $\mathcal{G}, n, g$ (and independent of σ).

Lemma 20. *If Prover has an (n, g)-strategy for a det-first-order grammar $\mathcal{G}$ then there is some* BASIS *for $\mathcal{G}$ which is sufficient for Prover to force her win for all $T_{in} \sim U_{in}$.*

Proof: We assume $\mathcal{G}$, n, g such that Prover has an (n, g)-strategy for $\mathcal{G}$, and we show that there is some (large) bound $\mathcal{B} \in \mathbb{N}$, determined (somehow) by $\mathcal{G}, n, g$, such that $\text{BASIS} = \{(E, F) \mid E \sim F, \text{PRESSIZE}(E, F) \leq \mathcal{B}\}$ satisfies the claim.

We consider a play of the P-R game in which $\mathcal{G}$ is given, the above BASIS (for some large $\mathcal{B}$) is chosen, and $T_{in} \sim U_{in}$ is given. In Point 4 Refuter necessarily chooses $T_0 \sim U_0$ (though

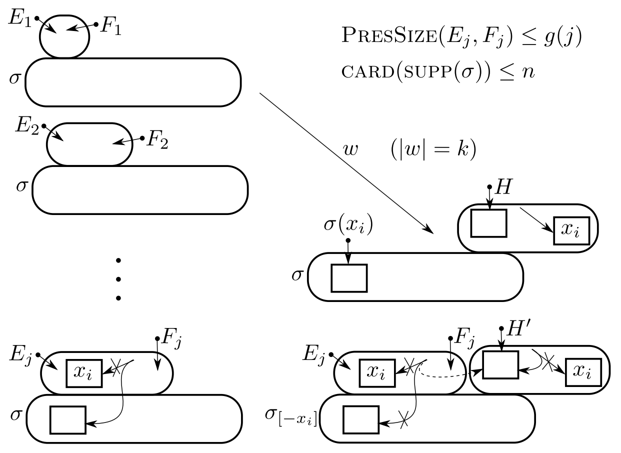

Fig. 13. An (n, g)-(sub)sequence (left); decreasing $\textsc{supp}(\sigma)$ by $\{(x_i, H')\}$

claiming $T_0 \not\sim U_0$). We let Prover use her assumed (n, g)-strategy, and consider a moment (after a number of phases) when the so far constructed sequence $(T_1, U_1), (T_2, U_2), \ldots$ has a "*long*" (n, g)-subsequence $(T_{i_1}, U_{i_1}) = (E_1\sigma, F_1\sigma)$, $(T_{i_2}, U_{i_2}) = (E_2\sigma, F_2\sigma)$, $\ldots$, $(T_{i_\ell}, U_{i_\ell}) = (E_\ell\sigma, F_\ell\sigma)$; let us write ℓ as $\ell_{(n,g)}$.

Prover can derive from Refuter's claims that

$$(E_1\sigma, F_1\sigma), (E_2\sigma, F_2\sigma), \ldots, (E_{\ell_{(n,g)}}\sigma, F_{\ell_{(n,g)}}\sigma) \qquad (3)$$

is eq-level decreasing (though in reality $E_i\sigma \sim F_i\sigma$ for all i).

If $E_1 \sim F_1$ (which must be the case when $n = 0$, so when $\textsc{card}(\textsc{supp}(\sigma)) = 0$) then Prover can claim her win if $\mathcal{B} \ge g(1)$: in this case $(T_{i_1}, U_{i_1}) = (E_1\sigma, F_1\sigma)$ is a basis-instance.

Assume now $\textsc{EqLv}(E_1, F_1) = k \in \mathbb{N}$; note that $k \le \textsc{MaxFEL}_{g(1)}$. Since $E_1\sigma \sim F_1\sigma$, by Prop. 12 we know that Prover can demonstrate $E_1\sigma \xrightarrow{w} \sigma(x_i)$ and $F_1\sigma \xrightarrow{w} H\sigma$ (or vice versa) for $x_i \in \textsc{supp}(\sigma)$, $H \ne x_i$ and $|w| = k$ (see Fig. 13). Moreover, she derives $\textsc{EqLv}(\sigma(x_i), H\sigma) > \textsc{EqLv}(E_{s+1}\sigma, F_{s+1}\sigma) > \textsc{EqLv}(E_{s+2}\sigma, F_{s+2}\sigma) > \cdots$ for (the shift) $s = 1 + \textsc{MaxFEL}_{g(1)}$.

Using (deduction rules based on) Proposition 13, 11(2) and 2, Prover can demonstrate that in the pairs $(E_j\sigma, F_j\sigma)$, for $j = s{+}1, s{+}2, \ldots$, she can replace σ with $\{(x_i, H')\}\sigma_{[-x_i]}$ where $\textsc{gp}_{H'}$ arises from $\textsc{gp}_H$ by redirecting each incoming arc of x_i to the root (see Fig. 13), *without affecting the eq-levels* of these pairs if Refuter's claims are true.

Note that $\textsc{PresSize}(H)$ is surely bounded by $g(1) + \textsc{MaxFEL}_{g(1)} \cdot \textsc{StepInc}$, where $\textsc{StepInc}$ can be taken as the size of the largest rhs in the rules of $\mathcal{G}$; it bounds the possible *one-step increase* of the presentation size when a rule is applied (recall Fig. 5).

Prover thus demonstrates an $(n{-}1, g')$-sequence

$$(E'_1\sigma_{[-x_i]}, F'_1\sigma_{[-x_i]}), (E'_2\sigma_{[-x_i]}, F'_2\sigma_{[-x_i]}), \ldots \qquad (4)$$

of length $\ell_{(n-1,g')} = \ell_{(n,g)} - (1 + \textsc{MaxFEL}_{g(1)})$ where $E'_j = E_{s+j}\{(x_i, H')\}$ and $F'_j = F_{s+j}\{(x_i, H')\}$; we note that $\textsc{PresSize}(E'_j, F'_j)$ is surely bounded (by $g(s{+}j) + 2 \cdot \textsc{PresSize}(H)$ and thus) by

$$g'(j) \text{ defined as} \qquad (5)$$

$$g(1{+}\textsc{MaxFEL}_{g(1)}{+}j) + 2\cdot(g(1) + \textsc{MaxFEL}_{g(1)} \cdot \textsc{StepInc}).$$

We can now reason for the sequence (4) as we did for the sequence (3). If $E'_1 \sim F'_1$ then Prover can claim her win if $\mathcal{B} \ge g'(1)$. If $E'_1 \not\sim F'_1$ then Prover creates an $(n{-}2, g'')$-sequence of length $\ell_{(n-2,g'')} = \ell_{(n-1,g')} - (1{+}\textsc{MaxFEL}_{g'(1)})$, etc. The iteration can happen at most n times, and thus $\mathcal{G}, n, g$ indeed determine some $\mathcal{B}$ which guarantees that the above $\textsc{Basis}$ is sufficient for forcing Prover's win for all $T_{in} \sim U_{in}$. ■

B. A Balancing Strategy which is an (n, g)-Strategy

In this subsection we prove the next lemma, by which a proof of Theorem 17 will be finished (by Lemmas 20, 16, 10).

Lemma 21. *For any det-first-order grammar $\mathcal{G}$, Prover has an (n, g)-strategy (n, g being determined by $\mathcal{G}$).*

Assuming a det-first-order grammar $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ (generating the LTS $\mathcal{L}^{\mathsf{A}}_{\mathcal{G}}$), we now describe a particular *balancing strategy* of Prover in the Prover-Refuter game when $\mathcal{G}$ is given in Point 1; this strategy will turn out to be an (n, g)-strategy. (We use a liberal notion of a strategy; it still leaves some free choice to Prover.) We start with some technical notions.

(Shortest) Sink Words; Root-Performability; Constant M_0

- A word $w \in \mathcal{A}^*$ is a *(Y, j)-sink-word*, where $1 \le j \le m = arity(Y)$, if $Yx_1 \ldots x_m \xrightarrow{w} x_j$ (hence if $YF_1 \ldots F_m \xrightarrow{w} F_j$ for all $F_1, \ldots, F_m$).
- A *path* $F \xrightarrow{u}$ (of length $|u|$) is *root-performable* if the root of F is $Y \in \mathcal{N}$ and $Yx_1 \ldots x_{arity(Y)} \xrightarrow{u}$; hence no proper prefix of u is a (Y, j)-sink word then; if, moreover, u itself is not a (Y, j)-sink word then $F \xrightarrow{u}$ is *strongly root-performable*.
- A *path* $G \xrightarrow{w}$ *sinks into depth k in* $\textsc{dom}(G)$ (recall $G : \mathbb{N}^* \to \mathcal{N} \cup \textsc{Var}$) if it sinks *along* some $\gamma = i_1 i_2 \ldots i_k \in \textsc{dom}(G)$, i.e. if $w = w_1 w_2 \cdots w_k$ and for each ℓ, $1 \le \ell \le k$, we have: w_ℓ is a (Y, i_ℓ)-sink word where $Y = G(i_1 i_2 \ldots i_{\ell-1})$. Hence $G \xrightarrow{\varepsilon}$ sinks into depth 0.

In Fig. 5, a is a $(Y, 1)$-sink-word of length 1. In Fig. 6, if $root(F) = A$ and the arc depicted in $\textsc{gp}_F$ is labelled 2 then $a_1a_2a_3a_4a_5a_6$ is an $(A, 2)$-sink word. $F \xrightarrow{a_1 a_2 \ldots a_j}$ is root-performable for all $j, 0 \le j \le 6$, but it is not strongly root-performable for $j = 6$. The next fact is clear from Fig. 5.

Fact 22. *If w is a (Y, j)-sink-word then $w = av$ and there is a rule $r : Yx_1 \ldots x_m \xrightarrow{a} E$ where $E \xrightarrow{v}$ sinks along some $\gamma \in \textsc{dom}(E)$ for which $E(\gamma) = x_j$ (so $E \xrightarrow{v} x_j$). (A particular case is $E = x_j$, $v = \varepsilon$.)*

It is thus clear that we can efficiently (by standard dynamic programming techniques) fix a *shortest (Y, j)-sink word* $\textsc{ssw}(Y, j)$ for each pair Y, j, $Y{\in}\mathcal{N}$, $1{\le}j{\le}arity(Y)$ (in our assumed $\mathcal{G}$) for which there is such a word. If there is no (Y, j)-sink word (so the j-th successor of Y is nonexposable and thus irrelevant) then we can safely decrease $arity(Y)$ and make the obvious corresponding modifications in the rules of $\mathcal{G}$. Hence we further assume $\textsc{ssw}(Y, j)$ for each Y, j, and put

$$M_0 = 1{+}\max\{\, |\textsc{ssw}(Y, j)| \mid Y{\in}\mathcal{N}, 1{\le}j{\le}arity(Y)\}. \quad (6)$$

Restricted Balancing

The balancing strategy which we are defining obliges Prover to choose $k = M_1$ in Point 5a of each phase of the game, where M_1 is a constant determined by $\mathcal{G}$ (sufficiently larger than M_0 as will be clarified later), and to restrict herself to the following way of balancing (in 5c).

A *left-balancing in Phase* i can only look as follows (we refer to Fig. 10 and 11 where we put $(T,U) = (T_i, U_i)$, $w = w_i$, $|w_i| = k = M_1$, and $(T', U') = (T'_i, U'_i)$):

1) If there is (T'_i, V) in $\text{REG}(T_i, U_i, M_1{-}1)$ for some V (as in Fig. 10) then Prover chooses one such pair and puts $T_{i+1} = V$, $U_{i+1} = U'_i$.
2) If 1) does not apply and there is a root-performable subpath of length M_0 in $T_i \xrightarrow{w_i} T'_i$ (see Fig. 11), then Prover takes *the last* such subpath, in the form $(Ax_1 \ldots x_m)\sigma \xrightarrow{v} G'\sigma$ where $|v| = M_0$, $w_i = u_1 v u_2$, $T_i \xrightarrow{u_1} (Ax_1 \ldots x_m)\sigma \xrightarrow{v} G'\sigma \xrightarrow{u_2} T'_i = G\sigma$, and $Ax_1 \ldots x_m \xrightarrow{v} G' \xrightarrow{u_2} G$. For each j, $1 \leq j \leq m$, Prover finds some $(\sigma(x_j), V'_j) \in \text{REG}(T_i, U_i, M_1{-}1)$ and defines $\sigma'(x_j) = V'_j$; finally she puts $T_{i+1} = G\sigma'$, $U_{i+1} = U'_i$.
3) If none of 1) and 2) applies, no left-balancing is allowed.

Soundness of defining 2): We have $(Ax_1 \ldots x_m)\sigma \xrightarrow{\text{SSW}(A,j)} \sigma(x_j)$ and Prover can thus take V'_j so that $F \xrightarrow{\text{SSW}(A,j)} V'_j$ where F is the right-hand-side counterpart of $(Ax_1 \ldots x_m)\sigma$; the path $F \xrightarrow{\text{SSW}(A,j)} V'_j$ must exist since $T_i \sim_{M_1} U_i$. We also note that there is indeed some G such that $G' \xrightarrow{u_2} G$: if we had $G'\sigma \xrightarrow{u'} \sigma(x_j)$ for a prefix u' of u_2 then $u_2 = u'u''$, $\sigma(x_j) \xrightarrow{u''} T'_i$ and we thus had the case 1), namely $(T'_i, V) \in \text{REG}(T_i, U_i, M_1{-}1)$ where $V'_j \xrightarrow{u''} V$ (as also depicted in Fig. 11).

In both cases 1) and 2), U_i is called the *balancing pivot* and (T_{i+1}, U_{i+1}) the *balancing result* (or the *bal-result*) of this balancing step. The *right balancing steps* are defined symmetrically (T_i is then the pivot).

Switching Balancing Sides is Separated by a No-Change Phase

The strategy obliges Prover to behave as follows in Phase i:

Prover balances, i.e. performs a left balancing step or a right balancing step as defined by 1) and 2) above, if possible but she cannot do a left (right) balancing if a right (left) balancing was done in Phase $i{-}1$; if balancing is (thus) not possible, Prover does no change, i.e. puts $T_{i+1} = T'_i$, $U_{i+1} = U'_i$.

Prover thus *cannot switch balancing sides in two consecutive phases*; such a switch needs a separating no-change phase. To finish the definition of the *balancing strategy*, we now define M_1 (so that the rest-head G in Fig. 12 gets "erased").

We put $\text{DEPTH}(E) = \max\{|\gamma| \mid \gamma \in \text{DOM}(E)\}$ for *finite* terms E, and we define the *maximal one-step depth-increase* STEPDINC (given by the rules in $\mathcal{G}$), and M_1 as follows:

$$\text{STEPDINC} = \max\{\text{DEPTH}(E){-}1 \mid E \text{ is the rhs of a rule}\},$$

$$M_1 = M_0 \cdot (2 + (2M_0{-}1) \cdot \text{STEPDINC}). \qquad (7)$$

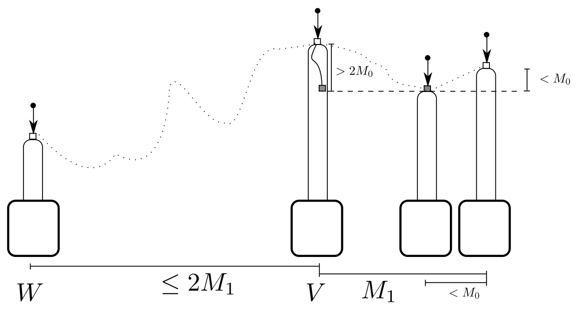

Fig. 14. (A prefix of) the path from a pivot W to the next pivot

Erasing the Rest-Head G when Balancing Sides are Switched

Let us consider a left balancing step in Phase i (as in Fig. 10 or 11). We note that U_{i+1} is reachable from the pivot U_i in M_1 steps. For T_{i+1} we observe that it has "a small finite rest-head G" completed with the tails σ' (like V'_1, V'_2 in Fig. 11) which are reachable from the pivot U_i within M_1 steps. (The rest-head can be even missing, like in Fig. 10.)

We note that $\text{DEPTH}(G') \leq 1 + M_0 \cdot \text{STEPDINC}$ where G' is the finite term referred to in 2) and Fig. 11. Since $G' \xrightarrow{u_2} G$ is M_0*-sinking*, which is a shorthand for saying that there is no root-performable subpath of length M_0 in $G' \xrightarrow{u_2} G$, we can easily check that $\text{DEPTH}(G) \leq \text{DEPTH}(G') + (M_0{-}1) \cdot \text{STEPDINC}$. Hence $\text{DEPTH}(G) \leq 1 + (2M_0{-}1) \cdot \text{STEPDINC}$.

We now consider a case where a left balancing is performed in Phase i and no left balancing is possible in Phase $i{+}1$; we assume $T_{i+1} = G\sigma'$ as above, and in Fig. 12. Phase $i{+}1$ is a no-change phase and the path $T_{i+1} \xrightarrow{w_{i+1}} T'_{i+1} = T_{i+2}$ is M_0-sinking; a prefix of this path thus sinks into the depth $d = M_1 \, div \, M_0$ in $\text{DOM}(T_{i+1})$. Since $d > \text{DEPTH}(G)$, there are u_1, u_2, j such that $w_{i+1} = u_1 u_2$ and $T_{i+1} = G\sigma' \xrightarrow{u_1} \sigma'(x_j) \xrightarrow{u_2} T_{i+2}$. This entails that both sides at the end of Phase $i{+}1$, i.e. both T_{i+2} and U_{i+2}, are reachable from the last pivot U_i within $2M_1$ steps ($U_i \xrightarrow{w_i} U_{i+1} \xrightarrow{w_{i+2}} U_{i+2}$, and $U_i \xrightarrow{v} \sigma'(x_j) \xrightarrow{u_2} T_{i+2}$ for some v, $|v| \leq M_1$).

Pivots of a Play are on a Special "Pivot-Path" in $\mathcal{L}^{\text{A}}_{\mathcal{G}}$

If Prover balances in Phase i and in Phase $i{+}1$ then we have $W \xrightarrow{w_i} W'$, $|w_i| = M_1$, for the respective pivots. ($W = U_i \xrightarrow{w_i} U_{i+1} = W'$ in the case of left balancings, and $W = T_i \xrightarrow{w_i} T_{i+1} = W'$ in the case of right balancings.)

If Prover balances in Phase i with pivot W ($W = U_i$ or $W = T_i$) and does no change in Phase $i{+}1$ then there are words v', v'' of length at most $2M_1$ such that $W \xrightarrow{v'} T_{i+2}$, $W \xrightarrow{v''} U_{i+2}$, as we noted above. If there is the next pivot, U_j or T_j for $j \geq i{+}2$, it is reachable by $w_{i+2} w_{i+3} \ldots w_{i+j-1}$ from T_{i+2} or U_{i+2} (depending on the side of the next pivot).

Thus the next pivot is reachable from the last pivot U_i by a special path: a "starting prefix" of length at most $2M_1$, finishing in some term V (either U_{i+1} or one of T_{i+2}, U_{i+2}), might be followed by a sequence of "follow-up" paths; each of these follow-up paths has length M_1 and is M_0-sinking. Fig. 14 depicts just one follow-up path; we assume $\text{STEPDINC} = 1$ there. (In Fig. 14 the starting prefix gives an impression of term-increasing but this is not true in general.) Our choice of M_1 guarantees "term-sinking" in the follow-up paths; in particular, any path in this follow-up sequence necessarily visits a subterm of V (in the ever greater depth in $\text{DOM}(V)$).

We also observe that if there is no next pivot (i.e. no next balancing) and the play is infinite then both sides (both T_j and U_j) range over finitely many terms, which entails getting a repeat ($(T_{j_1}, U_{j_1}) = (T_{j_2}, U_{j_2})$ for some $j_1 < j_2$).

To summarize, the pivots W_1, W_2, ... (of the balancing steps in a play where Prover adhers to the described balancing strategy) are on a (special) path

$$W_1 \xrightarrow{v_1} W_2 \xrightarrow{v_2} W_3 \xrightarrow{v_3} \cdots \tag{8}$$

where each $W_j \xrightarrow{v_j} W_{j+1}$ is in the above discussed form (as depicted in Fig. 14, which also captures $|v_j| = M_1$).

A Suffix of the Sequence of Bal-Results is an (n, g)-Sequence

We consider an infinite play in which $T_0 \sim U_0$. We have observed that if there are only finitely many balancings then we get a repeat. We thus further assume that there are infinitely many balancing steps in the play.

If a term V' (not only a pivot) is visited infinitely often by the path (8) then any particular visit of V' occurs in the path $W_j \xrightarrow{v_j} W_{j+1}$ for some j (V' is somewhere in the path in Fig. 14), and $\text{PRESSIZE}(W_{j+1})$ can be obviously only boundedly bigger than $\text{PRESSIZE}(V')$. Then one pivot appears infinitely often and the bal-results are infinitely often the same (as can be easily checked by Fig. 10 and 11); we get a repeat.

It remains to consider the case when there is a visit of a term $V = (Yx_1 \ldots x_m)\sigma'$ in (8) (a "stair-base", depicted as the second term in Fig. 15) such that no subterm of V is visited later; thus the rest of (8) is strongly root-performable. There is thus some $k \in \mathbb{N}$ such that (8) can be written as

$$W_1 \xrightarrow{u} V = (Yx_1 \ldots x_m)\sigma' \xrightarrow{u'} H_1\sigma' \xrightarrow{v_{k+1}} H_2\sigma' \xrightarrow{v_{k+2}} \cdots$$

where $(Yx_1 \ldots x_m) \xrightarrow{u'} H_1 \xrightarrow{v_{k+1}} H_2 \xrightarrow{v_{k+2}} \cdots$, and $H_1\sigma' = W_{k+1}$, $H_2\sigma' = W_{k+2}$, ... (By Fig. 14) we can check that

$$\text{DEPTH}(H_j) \leq 1 + j \cdot 2M_1 \cdot \text{STEPDINC}.$$

We now verify that the bal-results with pivots $H_1\sigma'$, $H_2\sigma'$, ... create an (n, g)-sequence $(E_1\sigma, F_1\sigma)$, $(E_2\sigma, F_2\sigma)$, ... (recall the left in Fig. 13), for some n and g determined by the grammar $\mathcal{G}$. To this aim, we present σ' as $\sigma' = \sigma''\sigma$ so that each $\sigma''(x_i)$ $(1 \leq i \leq m)$ is a finite term with $\text{DEPTH}(\sigma''(x_i)) \leq M_1{-}1$ where each $\gamma \in \text{DOM}(\sigma''(x_i))$ satisfies $|\gamma| = M{-}1$ iff $(\sigma''(x_i))(\gamma) \in \text{SUPP}(\sigma)$. (We use subterms of V occurring at depth M_1 to create σ.) We can use variables so that $\text{SUPP}(\sigma) \subseteq \{x_1, x_2, \ldots, x_n\}$ where

$$n = c^{M_1} \text{ for } c = \max\{\, arity(Y) \mid Y \in \mathcal{N}\,\}.$$

Recall that the bal-result corresponding to the pivot $H_j\sigma' = H_j\sigma''\sigma$ is composed from some terms reachable from $H_j\sigma''\sigma$ within M_1 moves (recall Figures 10 and 11), possibly also completed with a finite rest-head G where $\text{DEPTH}(G) \leq 1 + (2M_0{-}1) \cdot \text{STEPDINC}$. Since a path $H_j\sigma''\sigma \xrightarrow{u}$ of length at most M_1 can sink into depth at most M_1 in $\text{DOM}(H_j\sigma''\sigma)$,

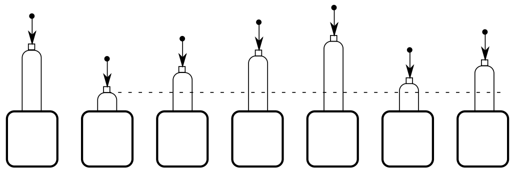

Fig. 15. First steps in path (8), the second term happens to be a "stair-base"

the bal-result related to $H_j\sigma''\sigma$ can be written as $(E_j\sigma, F_j\sigma)$ for finite terms E_j, F_j where $\text{DEPTH}(E_j)$, $\text{DEPTH}(F_j)$ are bounded by $\text{DEPTH}(H_j) + (M_1{-}1) + M_1 \cdot \text{STEPDINC} + (1 + (2M_0{-}1) \cdot \text{STEPDINC})$. This obviously yields some $g : \mathbb{N} \to \mathbb{N}$ (determined by $\mathcal{G}$) such that $\text{PRESSIZE}(E_j, F_j) \leq g(j)$ for $j = 1, 2, \ldots$.

Acknowledgment. The author thanks Colin Stirling for initial discussions, and Luca Aceto, Arnaud Carayol, Didier Caucal, and anonymous reviewers for later comments. The thanks go also to Philippe Schnoebelen and his colleagues at LSV ENS Cachan for organizing the first public presentation of this proof (in Cachan, January 2011).

REFERENCES

[1] J. Baeten, J. Bergstra, and J. Klop, "Decidability of bisimulation equivalence for processes generating context-free languages," *J.ACM*, vol. 40, no. 3, pp. 653–682, 1993.
[2] S. Böhm and S. Göller, "Language equivalence of deterministic real-time one-counter automata is NL-complete," in *MFCS 2011*, ser. LNCS, vol. 6907. Springer, 2011, pp. 194–205.
[3] C. H. Broadbent, A. Carayol, C.-H. L. Ong, and O. Serre, "Recursion schemes and logical reflection," in *LICS 2010*. IEEE Computer Society, 2010, pp. 120–129.
[4] O. Burkart, D. Caucal, F. Moller, and B. Steffen, "Verification on infinite structures," in *Handbook of Process Algebra*, J. Bergstra, A. Ponse, and S. Smolka, Eds. North-Holland, 2001, pp. 545–623.
[5] B. Courcelle, "Recursive applicative program schemes," in *Handbook of Theoretical Computer Science, vol. B*, J. van Leeuwen, Ed. Elsevier, MIT Press, 1990, pp. 459–492.
[6] W. Czerwiński and S. Lasota, "Fast equivalence-checking for normed context-free processes," in *Proc. FSTTCS'10*, ser. LIPIcs, vol. 8. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.
[7] S. Ginsburg and S. A. Greibach, "Deterministic context free languages," *Information and Control*, vol. 9, no. 6, pp. 620–648, 1966.
[8] S. Kiefer, A. S. Murawski, J. Ouaknine, B. Wachter, and J. Worrell, "On the complexity of the equivalence problem for probabilistic automata," in *FoSSaCS'12*, ser. LNCS, vol. 7213. Springer, 2012, pp. 467–481.
[9] A. Kučera and R. Mayr, "On the complexity of checking semantic equivalences between pushdown processes and finite-state processes," *Inf. Comput.*, vol. 208, no. 7, pp. 772–796, 2010.
[10] I. Lanese, J. A. Pérez, D. Sangiorgi, and A. Schmitt, "On the expressiveness and decidability of higher-order process calculi," *Inf. Comput.*, vol. 209, no. 2, pp. 198–226, 2011.
[11] S. Salvati and I. Walukiewicz, "Krivine machines and higher-order schemes," in *ICALP(2)'11*, ser. LNCS, vol. 6756. Springer, 2011, pp. 162–173.
[12] G. Sénizergues, "L(A)=L(B)? Decidability results from complete formal systems," *Theoretical Computer Science*, vol. 251, no. 1–2, pp. 1–166, 2001, (a preliminary version appeared at ICALP'97).
[13] ——, "L(A)=L(B)? a simplified decidability proof," *Theoretical Computer Science*, vol. 281, no. 1–2, pp. 555–608, 2002.
[14] ——, "The bisimulation problem for equational graphs of finite out-degree," *SIAM J.Comput.*, vol. 34, no. 5, pp. 1025–1106, 2005, (a preliminary version appeared at FOCS'98).
[15] G. Sénizergues, "The equivalence problem for t-turn dpda is co-NP," in *ICALP'03*, ser. LNCS, vol. 2719. Springer, 2003, pp. 478–489.
[16] C. Stirling, "Decidability of DPDA equivalence," *Theoretical Computer Science*, vol. 255, no. 1–2, pp. 1–31, 2001.
[17] ——, "Deciding DPDA equivalence is primitive recursive," in *Proc. ICALP'02*, ser. LNCS, vol. 2380. Springer, 2002, pp. 821–832.