Bisimulation Decomposition for PDA Decidability

Yuxi Fu
BASICS, Shanghai Jiao Tong University
fu-yx@cs.sjtu.edu.cn

Qiang Yin
BDBC, Beihang University
yinqiang@buaa.edu.cn

***Abstract*—Sénizergues has proved that language equivalence is decidable for disjoint ϵ-deterministic PDA. Stirling has showed that strong bisimilarity is decidable for PDA. On the negative side Srba demonstrated that the weak bisimilarity is undecidable for normed PDA. Jančar and Srba showed the undecidability of the weak bisimilarity for disjoint ϵ-pushing PDA and disjoint ϵ-popping PDA. In this paper it is shown that the branching bisimilarity of the normed ϵ-pushing PDA is decidable and the branching bisimilarity of the ϵ-pushing PDA is Σ^1_1-complete.**

I. Introduction

"Is it recursively unsolvable to determine if $L_1 = L_2$ for arbitrary deterministic languages L_1 and L_2"? The question was raised in Ginsburg and Greibach's 1966 paper [7] titled Deterministic Context Free Languages. The equality referred to in the quotation is the language equivalence between context free grammars. It is well known that the context free languages are precisely those accepted by pushdown automata (PDA) [10]. A PDA extends a finite state automaton with a memory stack. It accepts an input string whenever the memory stack is empty. The operational semantics of a PDA is defined by a finite set of rules of the following form

$$pX \xrightarrow{a} q\alpha \text{ or } pX \xrightarrow{\epsilon} q\alpha.$$

The transition rule $pX \xrightarrow{a} q\alpha$ reads "If the PDA is in state p with X being the top symbol of the stack, then it can accept an input letter a, pop off X, place the string α of stack symbols onto the top of the stack, and turn into state q". The rule $pX \xrightarrow{\epsilon} q\alpha$ describes a silent transition that has nothing to do with any input letter. It was proved early on that language equivalence between pushdown automata is undecidable [10]. A natural question asks what restrictions one may impose on the PDA's so that language equivalence becomes decidable. Ginsburg and Greibach studied deterministic context free languages. These are the languages accepted by deterministic pushdown automata (DPDA) [7].

A deterministic pushdown automaton enjoys disjointness and determinism properties. These conditions are defined as follows:

Disjointness. For all state p and all stack symbol X, if pX can accept a letter then it cannot perform a silent transition, and conversely if pX can do a silent transition then it cannot accept any letter.

A-Determinism. If $pX \xrightarrow{a} q\alpha$ and $pX \xrightarrow{a} q'\alpha'$ then $q = q'$ and $\alpha = \alpha'$.

ϵ-Determinism. If $pX \xrightarrow{\epsilon} q\alpha$ and $pX \xrightarrow{\epsilon} q'\alpha'$ then $q = q'$ and $\alpha = \alpha'$.

These are strong constraints from an algorithmic point of view. It turns out however that the language problem is still difficult even for this simple class of PDA's. One indication of the difficulty of the problem is that there is no size bound for equivalent DPDA configurations. It is easy to design a DPDA such that two configurations pY and pX^nY accept the same language for all n.

It was Sénizergues who proved after 30 years that the problem is decidable [23], [25]. His original proof is very long. Simplified proofs were soon discovered by Sénizergues [26] himself and by Stirlng [33]. After the positive answer of Sénizergues, one wonders if the strong constraints (disjointness+A-determinism+ϵ-determinism) can be relaxed. The first such relaxation was given by Sénizergues himself [24]. He showed that strong bisimilarity on the collapsed graphs of the disjoint ϵ-deterministic pushdown automata is also decidable. In the collapsed graphs all ϵ-transitions are absorbed. This result suggests that *A-non*determinism is harmless as far as decidability is concerned. The silent transitions considered in [24] are ϵ-popping. A silent transition $pX \xrightarrow{\epsilon} q\alpha$ is ϵ-popping if $\alpha = \epsilon$. In this paper we shall use a slightly more liberal definition of this terminology.

A PDA is *ϵ-popping* if $|\alpha| \le 1$ whenever $pX \xrightarrow{\epsilon} q\alpha$.
A PDA is *ϵ-pushing* if $|\alpha| \ge 1$ whenever $pX \xrightarrow{\epsilon} q\alpha$.

A disjoint ϵ-deterministic PDA can be converted to an equivalent disjoint ϵ-popping PDA in the following manner: Without loss of generality we may assume that the disjoint ϵ-deterministic PDA does not admit any infinite sequence of silent transitions. Suppose $pX \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} q\alpha$ and $q\alpha$ cannot do any silent transition. If $\alpha = \epsilon$ then we can redefine the semantics of pX by $pX \xrightarrow{\epsilon} q\epsilon$; otherwise we can remove pX in favour of qZ with Z being the first symbol of α. So under the disjointness condition ϵ-popping condition is weaker than ϵ-determinism.

A paradigm shift from a language viewpoint to a process algebraic viewpoint helps see the issue in a more productive way. Groote and Hüttel [8], [12] pointed out that as far as BPA and BPP are concerned the bisimulation equivalence à la Milner [21] and Park [22] is more tractable than the language equivalence. The best way to understand Senizergues' result is to recast it in terms of bisimilarity. Disjointness and ϵ-determinism imply that all silent transitions preserve equivalence. It follows that the branching bisimilarity [34] of the disjoint ϵ-deterministic PDA's coincides with the strong bisimilarity on the collapsed graphs of these PDA's. So what

Senizergues has proved is that the branching bisimilarity on the disjoint ϵ-deterministic PDA's is decidable.

The process algebraic approach allows one to use the apparatus from the process theory to study the equivalence checking problem for PDA. Stirling's proof of the decidability of the strong bisimilarity for normed PDA (nPDA) [29], [30] exploits the tableau method [13], [11]. Later he extended the tableau approach to the study of the unnormed PDA [32]. Stirling also provided a simplified account of Senizergues' proof [24] using the process method [33]. The proof in [33], as well as the one in [24], is interesting in that it turns the language equivalence of disjoint ϵ-deterministic PDA to the strong bisimilarity of correlated models. Another advantage of bisimulation equivalence is that it admits a nice game theoretical interpretation. This has been exploited in the proofs of negative results using the technique of Defender's Forcing [18]. Srba proved that weak bisimilarity on nPDA's is undecidable [27]. Jančar and Srba improved this result by showing that the weak bisimilarity on the disjoint nPDA's with only ϵ-popping transitions, respectively ϵ-pushing transitions, is already undecidable [18]. In fact they proved that the problems are Π^0_1-complete. Recently Yin, Fu, He, Huang and Tao have proved that the branching bisimilarity for all the models above either the normed BPA or the normed BPP in the hierarchy of process rewriting system [20] are undecidable [36]. This general result implies that the branching bisimilarity on nPDA is undecidable. Defender's Forcing can be used to study complexity bound. An example is Benedikt, Moller, Kiefer and Murawski's proof that the strong bisimilarity on PDA is non-elementary [2]. A summary of the (un)decidability results mentioned above is given in the following table, where $\sim$ stands for the strong bisimilarity, $\simeq$ the branching bisimilarity, and $\approx$ the weak bisimilarity.

	PDA	**nPDA**
$\sim$	Decidable [24], [32] Non-Elementary [2]	Decidable [29], [30] Non-Elementary [2]
$\simeq$	Undecidable [36]	Undecidable [36]
$\approx$	Σ^1_1-Complete [18] Undecidable [27]	Σ^1_1-Complete [18] Undecidable [27]

The decidability of the strong bisimilarity and the undecidability of the weak bisimilarity still leaves a number of questions unanswered. The prime motivation for this work is to establish a stronger result that would subsume both the decidability results in the language theme and the decidability results in the process algebraic line. This is desirable since these two classes of results are incompatible, neither implies the other. A conservative extension of the language equivalence for DPDA is not the strong bisimilarity because language equivalence ignores silent transitions. It is not the weak bisimilarity since the whole point of introducing the disjointness and ϵ-determinism conditions is to force all silent transitions to preserve equivalence. Our basic idea is to look at the decidability issue of the ϵ-pushing and ϵ-popping PDA's. Their decidability would subsume the two classes of incompatible decidability results.

The contributions of this paper are summarized as follows.

1) We prove that the branching bisimilarity on the normed ϵ-pushing PDA is decidable, and that the branching bisimilarity on the ϵ-pushing PDA is Σ^1_1-complete.
2) We propose a bisimulation decomposition approach, for decidability proof of bisimulation equivalence that evolves silent transitions, that could have applications in other models of process rewriting system.

The rest of the paper is organised as follows. Section II fixes the syntax and the semantics of PDA. Section III reviews the basic properties of the branching bisimilarity. Section IV discusses the finite branching property for the normed ϵ-pushing PDA. Section V introduces bisimulation trees. Section VI and Section VII discuss decomposition of bisimulation trees. Section VIII studies bisimulation base. Section IX establishes the decidability of the normed ϵ-pushing PDA by exploiting the finite tree property. Section X proves the high undecidability of the general ϵ-pushing PDA. Section XI concludes.

II. PDA

A *pushdown automaton* (PDA) $\Gamma = (\mathcal{Q}, \mathcal{V}, \mathcal{L}, \mathcal{R})$ consists of

- a state set $\mathcal{Q} = \{p_1, \ldots, p_q\}$ ranged over by o, p, q, r, s, t,
- a symbol set $\mathcal{V} = \{X_1, \ldots, X_n\}$ ranged over by X, Y, Z,
- a letter set $\mathcal{L} = \{a_1, \ldots, a_s\}$ ranged over by a, b, c, d, and
- a finite set $\mathcal{R}$ of transition rules.

If we think of a PDA as a process we may interpret a letter in $\mathcal{L}$ as an action label. The set $\mathcal{L}^*$ of words is ranged over by u, v, w. Following the process algebraic convention a silent action will be denoted by τ. The set $\mathcal{A} = \mathcal{L} \cup \{\tau\}$ of actions is ranged over by ℓ. The set $\mathcal{A}^*$ of action sequence is ranged over by ℓ^*. The set $\mathcal{V}^*$ of finite strings of symbols is ranged over by small Greek letters. The empty string is denoted by ϵ. We write $\alpha\beta$ for the concatenation of α and β. Since concatenation is associative no parenthesis is necessary when we write $\alpha\beta\gamma$. The length of α is denoted by $|\alpha|$.

The syntax of a *PDA process* is $p\alpha$, where $p \in \mathcal{Q}$ is a state and $\alpha \in \mathcal{V}^*$ is called a stack. We shall write L, M, N, O, P, Q for PDA processes. If $P = p\alpha$ then $P\beta$ stands for the PDA process $p\alpha\beta$. The transition set $\mathcal{R}$ of a PDA contains rules of the form $pX \xrightarrow{\ell} q\alpha$. The semantics of the PDA processes is defined by the following two structural rules.

$$\frac{pX \xrightarrow{\ell} q\alpha \in \mathcal{R}}{pX \xrightarrow{\ell} q\alpha} \qquad \frac{pX \xrightarrow{\ell} q\alpha}{pX\beta \xrightarrow{\ell} q\alpha\beta}$$

We shall use the standard notation $\xrightarrow{\ell^*}$ and $\Longrightarrow$ and $\xRightarrow{\ell^*}$. We say that P' is a descendant of P if $P \xrightarrow{\ell^*} P'$ for some ℓ^*. A process P *accepts* a word w if $P \xRightarrow{w} p\epsilon$ for some p. A process P is *normed*, or P is an nPDA process, if $P \xrightarrow{\ell^*} p\epsilon$ for some ℓ^*, p. A PDA $\Gamma = (\mathcal{Q}, \mathcal{V}, \mathcal{L}, \mathcal{R})$ is normed, or Γ is an nPDA, if pX is normed for all $p \in \mathcal{Q}$ and all $X \in \mathcal{V}$. The notation (n)PDA$^{\epsilon+}$ will refer to the variant of (n)PDA with ϵ-pushing transitions, and (n)PDA$^{\epsilon-}$ to the variant of (n)DPA with ϵ-popping transitions.

III. Branching Bisimilarity

The definition of branching bisimilarity is due to van Glebbeek and Weijland [35]. Care should be taken to processes of the form $p\epsilon$ since we want our bisimilarity to be a congruence relation [29].

Definition 1. *A binary relation $\mathcal{R}$ on PDA processes is a branching simulation if the following statements are valid whenever $P\mathcal{R}Q$:*

1) *If $P \xrightarrow{a} P'$ then there are some Q', Q'' such that $Q \Longrightarrow Q'' \xrightarrow{a} Q'$ and $P\mathcal{R}Q''$ and $P'\mathcal{R}Q'$.*
2) *If $P \xrightarrow{\tau} P'$ then either $Q \Longrightarrow Q'$ and $P\mathcal{R}Q'$ and $P'\mathcal{R}Q'$ for some Q' or $Q \Longrightarrow Q'' \xrightarrow{\tau} Q'$ and $P\mathcal{R}Q''$ and $P'\mathcal{R}Q'$ for some Q', Q''.*
3) *If $P = p\epsilon$ then $Q' \Longrightarrow p\epsilon$ whenever $Q \Longrightarrow Q'$.*

The relation is a branching bisimulation *if both $\mathcal{R}$ and its inverse $\mathcal{R}^{-1}$ are branching simulation. The* branching bisimilarity $\simeq$ *is the largest branching bisimulation.*

We write $\simeq_{\mathrm{nPDA}^{\epsilon+}}$, or simply $\simeq$, for the branching bisimilarity on nPDA$^{\epsilon+}$ processes. The proof of the following is easy.

Lemma 2. *$P \simeq_{\mathrm{nPDA}^{\epsilon+}} p\epsilon$ if and only if $P = p\epsilon$.*

Let $\mathcal{R}_1, \mathcal{R}_2$ be two branching bisimulations. The composition $\mathcal{R}_1;\mathcal{R}_2 = \{(O,Q) \mid \exists P.(O,P) \in \mathcal{R}_1 \wedge (P,Q) \in \mathcal{R}_2\}$ is a branching bisimulation. This is proved in [1]. Branching bisimulations are also closed under set theoretical union. Consequently $\simeq$ is an equivalence. Moreover it is also a congruence. Therefore $P\alpha \xrightarrow{\epsilon} P'\alpha$ whenever $P \xrightarrow{\epsilon} P'$. A technical lemma that plays an important role in the study of branching bisimilarity is the Computation Lemma [35], [5].

Lemma 3. *If $P_0 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_k \simeq P_0$, then $P_0 \simeq \ldots \simeq P_k$.*

A silent transition $P \xrightarrow{\tau} P'$ is *state-preserving*, notation $P \xrightarrow{\epsilon} P'$, if $P \simeq P'$. It is a *change-of-state*, notation $P \xrightarrow{\iota} P'$, if $P \not\simeq P'$. We write $(\rightarrow^*)$ $\rightarrow^+$ for the (reflexive and) transitive closure of $P \xrightarrow{\epsilon} P'$. The notation $P \not\rightarrow$ stands for the fact that $P \not\simeq P'$ for all P' such that $P \xrightarrow{\tau} P'$. Let $\jmath$ range over $\mathcal{L} \cup \{\iota\}$. We will find it necessary to use the notation $\xrightarrow{\jmath}$. The transition $P \xrightarrow{\jmath} P'$ refers to either $P \xrightarrow{a} P'$ for some $a \in \mathcal{L}$ or $P \xrightarrow{\iota} P'$. Lemma 3 implies that if $P_0 \xrightarrow{\jmath} P_1$ is bisimulated by $Q_0 \xrightarrow{\tau} Q_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} Q_k \xrightarrow{\jmath} Q_{k+1}$, then $Q_0 \xrightarrow{\epsilon} Q_1 \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} Q_k$. This is a very useful property.

Given a PDA process P, the *norm* of P *over* σ, denoted by $\|P\|_\sigma$, is a function from $[\mathfrak{q}]$ to $\mathbb{N} \cup \{\perp\}$, where $[\mathfrak{q}] = \{1, \ldots, \mathfrak{q}\}$ and $\perp$ stands for undefinedness, such that the following holds:

- $\|P\|_\sigma(h) = \perp$ if there is no ℓ^* such that $P\sigma \xrightarrow{\ell^*} p_h\sigma$.
- $\|P\|_\sigma(h)$ is the least number i such that $P\sigma \rightarrow^* \xrightarrow{\jmath_1} \ldots \rightarrow^* \xrightarrow{\jmath_i} \rightarrow^* p_h\sigma$ for some $\jmath_1 \ldots \jmath_i$.

By definition $\|P\|_\sigma(h) = \|Q\|_\sigma(h)$ if $P \simeq Q$. Let $\ker\|P\|_\sigma$ be the finite set $\{h \mid \|P\|_\sigma(h) \neq \perp\}$. For nPDA$^{\epsilon+}$ process P we introduce the following notations.

$$\begin{aligned}
\min\|P\|_\sigma &= \min\{\|P\|_\sigma(h) \mid h \in \ker\|P\|\}, \\
\max\|P\|_\sigma &= \max\{\|P\|_\sigma(h) \mid h \in \ker\|P\|\}.
\end{aligned}$$

We will omit the subscript σ if $\sigma = \epsilon$. A process P is said to be *normed* if $\ker\|P\| \neq \emptyset$. It is *unnormed* otherwise. We shall use the following convention in the rest of the paper.

$$\begin{aligned}
\mathfrak{r} &= \max\left\{|\eta| \,\middle|\, pX \xrightarrow{\ell} q\eta \in \mathcal{R} \text{ for some } p,q \in \mathcal{Q}, X \in \mathcal{V}\right\}, \\
\mathfrak{m} &= \max\{\max\|pX\| \mid p \in \mathcal{Q}, X \in \mathcal{V}\}.
\end{aligned}$$

The values $\mathfrak{r}$ and $\mathfrak{m}$ can be effectively calculated using a dynamic programming algorithm. By definition $\|pX\|(i) \leq \mathfrak{m}$ for all p, X and all $i \in \ker\|pX\|$.

IV. Finite Branching Property

Generally bisimilarity is undecidable for models with infinite branching transitions. For the branching bisimilarity the finite branching property is defined by the following statement:

> For each P and ℓ there is a finite set of processes $\{P_i\}_{i\in I}$ such that $P' \simeq P_i$ for some $i \in I$ whenever $P \rightarrow^* \xrightarrow{\ell} P'$.

We prove in this section that nPDA$^{\epsilon+}$ enjoys the finite branching property. Before doing that we need be assured that silent transition cycles of nPDA$^{\epsilon+}$ processes do not render a problem. There is in fact an effective procedure to remove such a silent transition cycle. A clique $\mathcal{S}$ is a subset of $\{pX \mid p \in \mathcal{Q}, \text{ and } X \in \mathcal{V}\}$ such that for every two distinct members pX, qY of $\mathcal{S}$ there is a silent transition sequence from pX to qY. It follows from Lemma 3 that the members of a clique are branching bisimilar. We can remove a maximal clique $\mathcal{S}$ in two steps.

1) Remove all rules of the form $pX \xrightarrow{\tau} qY$ such that $pX, qY \in \mathcal{S}$.
2) For each $pX \in \mathcal{S}$ introduce the rule $pX \xrightarrow{\lambda} P$ whenever there is some $qY \in \mathcal{S}$ that is distinct from pX and the rule $qY \xrightarrow{\lambda} P$ has not been removed in the first step.

In the new nPDA$^{\epsilon+}$ there is no circular silent transition sequence involving any member of $\mathcal{S}$ due to the maximality of $\mathcal{S}$. The legitimacy of transformation is guaranteed by Lemma 3. From now on we assume that such circularity does not occur in our nPDA$^{\epsilon+}$. Consequently for an nPDA$^{\epsilon+}$ with $\mathfrak{n}$ variables and $\mathfrak{q}$ states the length of a silent transition sequence of the form $qX \xrightarrow{\tau} q_1X_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} q_kX_k$ is upper bounded by $\mathfrak{n}\mathfrak{q}$.

Lemma 4. *In nPDA$^{\epsilon+}$, $|\alpha| \leq \min\|p\alpha\|$ holds for all $p\alpha$.*

Proof. In nPDA$^{\epsilon+}$ only external actions remove symbols from a stack. Silent actions never decrease the size of a stack. □

Using the simple property stated in Lemma 4, one can show that there is a constant bound for the length of the state-preserving transitions in nPDA$^{\epsilon+}$.

Lemma 5. *Suppose $qX\sigma \xrightarrow{\epsilon} q_1\beta_1\sigma \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} q_k\beta_k\sigma$ for an nPDA$^{\epsilon+}$ process $qX\sigma$. Then $k < \mathfrak{q}\mathfrak{n}\mathfrak{r}(\mathfrak{m}+1)^{\mathfrak{q}}$.*

Proof. Suppose $qX\sigma \xrightarrow{\epsilon} q_1Z_1\delta_1\sigma$. Let

$$q_1Z_1\delta_1\sigma \rightarrow^* \xrightarrow{\jmath_1^1} \ldots \rightarrow^* \xrightarrow{\jmath_{j_1}^1} \rightarrow^* r_1\epsilon\sigma \rightarrow^* \xrightarrow{\jmath_{j_1+1}^1} \ldots \rightarrow^* \xrightarrow{\jmath_{j_{k_1}}^1} \rightarrow^* p_{h_1}\epsilon$$

be a transition sequence of minimal length that empties the stack, where $k_1 = \min\|q_1Z_1\delta_1\sigma\|$. Clearly $j_1 \le \mathfrak{r}\mathfrak{m}$. Suppose $q_1Z_1\delta_1\sigma \to^* q_2Z_2\delta_2\delta_1\sigma$ such that $\mathfrak{r}\mathfrak{m} < |Z_2\delta_2\delta_1| \le \mathfrak{r}(\mathfrak{m}+1)$. Let $Q_2 = q_2Z_2\delta_2\delta_1\sigma$ and $k_2 = \min\|Q_2\|$, and let

$$Q_2 \to^* \xrightarrow{j^2_1} \ldots \to^* \xrightarrow{j^2_{j_2}} \to^* r_2\epsilon\sigma \to^* \xrightarrow{j^2_{j_2+1}} \ldots \to^* \xrightarrow{j^2_{j_{k_2}}} \to^* p_{h_2}\epsilon$$

be a transition sequence of minimal length that empties the stack. One must have $j_2 > j_1$ according to the size bound on $Z_2\delta_2\delta_1$. By iterating the above argument one gets from

$$\begin{aligned} q_1Z_1\delta_1\sigma \;&\to^*\; q_2Z_2\delta_2\delta_1\sigma \\ &\to^*\; \ldots \\ &\to^*\; q_{i+1}Z_{i+1}\delta_{i+1}\delta_i\ldots\delta_1\sigma \\ &\to^*\; \ldots \\ &\to^*\; q_{\mathfrak{q}+1}Z_{\mathfrak{q}+1}\delta_{\mathfrak{q}+1}\delta_{\mathfrak{q}}\ldots\delta_1\sigma \end{aligned}$$

with $\mathfrak{r}\mathfrak{m}(\mathfrak{m}+1)^{i-1} < |Z_{i+1}\delta_{i+1}\delta_i\ldots\delta_1| \le \mathfrak{r}(\mathfrak{m}+1)^i$ for all $i \in [\mathfrak{q}]$, some states $r_1,\ldots,r_{\mathfrak{q}+1}$, some numbers $k_1 < \ldots < k_{\mathfrak{q}+1}$ and $h_1,\ldots,h_{\mathfrak{q}+1}$. For each $i \in [\mathfrak{q}+1]$ there is some transition sequence

$$Q_i \to^* \xrightarrow{j^i_1} \ldots \to^* \xrightarrow{j^i_{j_i}} \to^* r_i\epsilon\sigma \to^* \xrightarrow{j^i_{j_i+1}} \ldots \to^* \xrightarrow{j^i_{j_{k_i}}} \to^* p_{h_i}\epsilon$$

where $Q_i = q_iZ_i\delta_i\ldots\delta_1\sigma$ and $k_i = \min\|Q_i\|$. Since there are only $\mathfrak{q}$ states, there must be some t_1, t_2 such that $0 < t_1 < t_2 \le \mathfrak{q}+1$ and $r_{t_1} = r_{t_2}$. It follows from the minimality that $j_{k_{t_1}} - j_{t_1} = j_{k_{t_2}} - j_{t_2}$. But $j_{t_2} > j_{t_1}$. Consequently $j_{k_{t_1}} < j_{k_{t_2}}$. This inequality contradicts to the fact that $q_{t_1}Z_{t_1}\delta_{t_1}\ldots\delta_1\sigma \simeq q_{t_2}Z_{t_2}\delta_{t_2}\ldots\delta_1\sigma$. We conclude that if $qX\sigma \to^* q'\gamma\sigma$ then $|\gamma| < \mathfrak{r}(\mathfrak{m}+1)^{\mathfrak{q}}$. Since there is no ϵ-loop the bound $k < \mathfrak{q}\mathfrak{n}\mathfrak{r}(\mathfrak{m}+1)^{\mathfrak{q}}$ follows. □

Corollary 6. *Suppose P is an* nPDA$^{\epsilon+}$ *process. There is a computable bound on the size of any* nPDA$^{\epsilon+}$ *process Q such that $Q \simeq P$.*

Proof. The norm of Q is bounded by $\mathfrak{m}|Q|$. It follows that $|P$ is bounded by $(\mathfrak{q}\mathfrak{n}\mathfrak{r}(\mathfrak{m}+1)^{\mathfrak{q}})\,\mathfrak{m}|Q|$. □

Using the finite branching property guaranteed by Lemma 5 it is standard to prove the following.

Proposition 7. *The relation $\not\simeq_{\mathrm{nPDA}^{\epsilon+}}$ is semidecidable.*

Proof. Let $\simeq_0$ be the total relation. The symmetric relation $\simeq_{k+1}$ is defined as follows: $P \simeq_{k+1} Q$ if the following statements are valid:

1) If $Q \xrightarrow{a} Q'$ then $P \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_j \xrightarrow{a} P' \simeq_k Q'$ for some $P_1,\ldots,P_j,P'$ such that $P_i \simeq_k Q$ for all $i \in [j]$.
2) If $Q \xrightarrow{\epsilon} Q'$ then either $P \simeq_k Q'$ or some $P_1,\ldots,P_j,P'$ exist such that $P \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_j \xrightarrow{\epsilon} P' \simeq_k Q'$ and $P_i \simeq_k Q$ for all $i \in [j]$.
3) If $P = p\epsilon$ then $Q = p\epsilon$.

The approximation $\simeq_0 \supseteq \simeq_1 \supseteq \simeq_2 \supseteq \ldots$ approaches to $\simeq$. By standard argument using Lemma 5 one shows that $\bigcap_{i\ge 0} \simeq_i$ is $\simeq$. So $P \not\simeq Q$ can be checked by checking $P \simeq_i Q$ for $i > 0$. □

V. Bisimulation Tree

Intuitively a bisimulation tree for (P,Q) is a stratified presentation of a branching bisimulation for (P,Q) in which one tries to collapse all state-preserving silent transitions from a pair of bisimilar processes as one mega node. This reminds one of the collapsed graphs introduced by Sénizergues. Formally a *bisimulation tree* $\mathfrak{B}$ *for* (P,Q) is a finite branching rooted tree such that each of its nodes is labeled by a pair (M,N) of PDA$^{\epsilon+}$ processes and a directed edge is labeled by a member of $\mathcal{L}\cup\{\tau\}\cup\{\epsilon\}$. The root is labeled by (P,Q), and the following properties hold for every node labeled by say (M,N).

1) M is descendant of P and N is a descendant of Q.
2) If there is an edge labeled ϵ from the node labeled (M,N) to a node labeled (M',N') then either $M \stackrel{\tau}{\Longrightarrow} M'$ and $N \Longrightarrow N'$ or $M \Longrightarrow M'$ and $N \stackrel{\tau}{\Longrightarrow} N'$, and the following are valid.
   a) There is no edge labeled ϵ from the node labeled (M,N) to a node labeled by (M'',N'') such that both $(M \Longrightarrow M'' \stackrel{\tau}{\Longrightarrow} M') \vee (M' \stackrel{\tau}{\Longrightarrow} M'')$ and $(N \Longrightarrow N'' \stackrel{\tau}{\Longrightarrow} N') \vee (N' \stackrel{\tau}{\Longrightarrow} N'')$.
   b) If there are ϵ labeled edges from the node labeled (M,N) to nodes labeled by (M',N') and (M'',N'') respectively, then there are ϵ labeled edges from the node labeled (M,N) to nodes labeled by (M',N'') and (M'',N') respectively.
   c) There is no ϵ labeled edge from the node labeled (M',N').
3) Suppose there is an ϵ labeled edge from the node labeled (L,O) to a node labeled (M,N). A silent transition $P_0 \xrightarrow{\tau} P_1$, respectively $Q_0 \xrightarrow{\tau} Q_1$, is *implicit* with regards to (L,O) if there is an ϵ labeled edge from the node labeled (L,O) to a node labeled by some (M',N') such that $P_0 \xrightarrow{\tau} P_1$, respectively $Q_0 \xrightarrow{\tau} Q_1$, appears in a silent transition sequence from L to M', respectively from O to N'. The third requirement states that for the ϵ labeled edge from the node labeled (L,O) to the node labeled (M,N) the following are valid.
   a) If $L \Longrightarrow L' \Longrightarrow M$ and $L' \xrightarrow{\lambda} L''$ is not implicit with regards to (L,O), then some N' exists such that $N \xrightarrow{\lambda} N'$ and there is a λ labeled edge from the node labeled (M,N) to a node labeled (L'',N').
   b) If $O \Longrightarrow O' \Longrightarrow N$ and $O' \xrightarrow{\lambda} O''$ is not implicit with regards to (L,O), then some M' exists such that $M \xrightarrow{\lambda} M'$ and there is a λ labeled edge from the node labeled (M,N) to a node labeled (M',O'').

We write $\mathfrak{B}(P,Q)$ for a bisimulation tree for (P,Q).

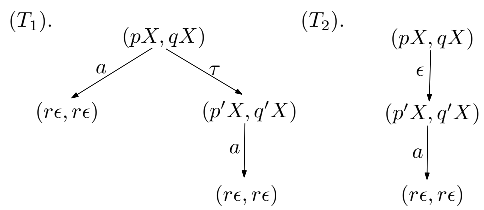
Recall that our PDA$^{\epsilon+}$ does not admit silent loop action sequence and that by Lemma 5 there is a computable bound on the length of all state-preserving silent transition sequences, hence the well-definedness of the ϵ-labeled edges. We shall not introduce a formal treatment of rooted trees. For proof of decidability this level of formality should be sufficient.

It is easy to see that if one reverses the order of the labels of a bisimulation tree $\mathfrak{B}(P,Q)$ for (P,Q) one obtains a bisimulation tree for (Q, P), denoted by $\mathfrak{B}^{-1}(P, Q)$. In what follows we often refer to a node by its label. Accordingly we call an edge $(M, N) \stackrel{\tau}{\longrightarrow} (M', N')$ for example a τ-edge.

Let's see some bisimulation trees. Suppose a PDA has the following semantic rules.

$$pX \stackrel{\tau}{\longrightarrow} p'X,\ p'X \stackrel{a}{\longrightarrow} r\epsilon,\ pX \stackrel{a}{\longrightarrow} r\epsilon;$$
$$qX \stackrel{\tau}{\longrightarrow} q'X,\ q'X \stackrel{a}{\longrightarrow} r\epsilon,\ qX \stackrel{a}{\longrightarrow} r\epsilon.$$

The following are two distinct bisimulation trees, T_1 and T_2, for the process pair (pX, qX).



In this example $pX \simeq p'X$ and $qX \simeq q'X$. So the ϵ edge in T_2 is justified. We can construct a different bisimulation tree without making use of the fact $pX \to p'X$ and $qX \to q'X$.

Given a bisimulation tree $\mathfrak{B}$ for (P, Q). Define $\overline{\mathfrak{B}}$ inductively as follows.

1) If (M, N) is a label in $\mathfrak{B}$ then $(M, N) \in \overline{\mathfrak{B}}$.
2) If $(M, N) \stackrel{\epsilon}{\longrightarrow} (M', N')$, then $(M'', N'') \in \overline{\mathfrak{B}}$ for each pair (M'', N'') such that $M \Longrightarrow M'' \Longrightarrow M'$ and $N \Longrightarrow N'' \Longrightarrow N'$.

In the above example, $\overline{T_1} = \{(pX, qX), (p'X, q'X), (r\epsilon, r\epsilon)\}$ and $\overline{T_2} = \{(pX, qX), (p'X, q'X), (p'X, qX), (pX, q'X), (r\epsilon, r\epsilon)\}$.

The almost trivial proof of the property stated next justifies the slightly complicated definition of bisimulation tree.

Lemma 8. *$\overline{\mathfrak{B}}$ is a branching bisimulation.*

It follows immediately from Lemma 8 that $(M, N) \stackrel{\epsilon}{\longrightarrow} (M', N')$ implies either $M \to^+ M'$ and $N \to^* N'$ or $M \to^* M'$ and $N \to^+ N'$.

From a proof perspective we think of a pair (P, Q) as a *goal*. The bisimulation tree for (P, Q) is a proof that the goal can be established in the sense that $P \simeq Q$. If $P \simeq Q$ there is a canonical bisimulation tree for (P, Q) in which every τ-edge represents a change-of-state silent transition. However there could be bisimulation trees for (P, Q) in which a τ-edge is actually a state-preserving silent transition; see the T_1 in the above example. We will refer to a bisimulation for a pair (P, Q) satisfying $P \simeq Q$ a *bisimulation tree of* $P \simeq Q$. In this case we also say that $P \simeq Q$ is a goal.

We assign a level number to every node of the tree. The level number of the root is 0. Suppose the level number of (M, N) is k. Then the level number of (M', N') is k if $(M, N) \stackrel{\epsilon}{\longrightarrow} (M', N')$, it is $k + 1$ if $(M, N) \stackrel{\tau}{\longrightarrow} (M', N')$ or $(M, N) \stackrel{a}{\longrightarrow} (M', N')$. We say that a bisimulation tree is generated or *grown* level by level if for each $k \geq 0$ none of its nodes at the $(k{+}1)$-th level is generated before all nodes at the k-th level have been generated. The *k-subtree* of a tree consists of all the nodes of the latter whose level number is no more than k.

VI. Bisimulation Trees of Bisimilar Processes

Bisimulation trees of bisimilar nPDA$^{\epsilon+}$ processes are in general infinite. Our task is to decompose every such tree to a finite number of finite trees such that a new bisimulation tree of the root label of the tree can be composed by piecing together the finite trees in an inductive manner. The first step to achieve that is to transform a goal to another goal such that the two processes in the latter share a common suffix. This is done by increasing the size of common suffix and controlling the size of prefixes. Bisimulation trees of a pair of bisimilar processes with shared suffix can be decomposed by introducing a finite number of subgoals. In other words we can start with a goal that has an empty common suffix and carry out decomposition inductively. These will be explored in this section. Most results and proof ideas of this section are due to Stirling [31]. Our presentation is in terms of branching bisimulation rather than tableaux systems for strong bisimilarity. The simple syntax of PDA will be respected as much as possible.

A. Decomposing Bisimulation Tree over Common Suffix

Suppose $P\sigma \simeq Q\sigma$ with $|P| > 0$ and $|Q| > 0$. A *bisimulation tree of $P\sigma \simeq Q\sigma$ over σ* is grown like a bisimulation tree for $(P\sigma, Q\sigma)$. The major difference is that the suffix σ should remain intact throughout the construction. A transition $O\sigma \stackrel{a}{\longrightarrow} O'\sigma$ for example is admitted in the buildup of the bisimulation tree only if $|O| > 0$. In the following construction we maintain three parameters modified dynamically.

- The first is a set $\mathcal{G} = \{G_i\}_{i \in [\mathfrak{q}]}$ of processes such that
$$p_1\sigma \simeq G_1\sigma,\ p_2\sigma \simeq G_2\sigma,\ \ldots,\ p_{\mathfrak{q}}\sigma \simeq G_{\mathfrak{q}}\sigma \qquad (1)$$
Initially $G_i = i$ for all $i \in [\mathfrak{q}]$.
- The second is an equivalence relation $\mathcal{E}$ on $[\mathfrak{q}]$. We write $i \in \mathcal{E}$ if $\langle i, i \rangle \in \mathcal{E}$. Initially $\mathcal{E} = \{\langle i, i \rangle \mid i \in [\mathfrak{q}]\}$.
- The third is a *recursive stack V à la* Stirling [31]. The generalized stack V is defined by $\mathfrak{q}$ grammar equalities
$$p_1 V = L_1 V,\ p_2 V = L_2 V,\ \ldots,\ p_{\mathfrak{q}} V = L_{\mathfrak{q}} V \qquad (2)$$
where $L_1, \ldots, L_{\mathfrak{q}}$ are PDA processes defined by
$$L_i = \begin{cases} G_i, & \text{if } i \notin \mathcal{E}, \\ \min\{j \mid (i, j) \in \mathcal{E}\}, & \text{if } i \in \mathcal{E}. \end{cases}$$
Initially $L_i = i$ for all $i \in [\mathfrak{q}]$.

We shall update $\mathcal{G}$ and $\mathcal{E}$ dynamically while we build up the tree level by level. The definition of the recursive stack V is updated accordingly. The correlation between (1) and (2) renders true the fundamental property stated next.

Lemma 9. *$PV \simeq QV$ implies $P\sigma \simeq Q\sigma$ whenever $|P|, |Q| > 0$.*

Proof. Let $\mathcal{R}$ be $\{(P\sigma, Q\sigma) \mid PV \simeq QV \wedge |P| > 0 \wedge |Q| > 0\}$. We prove that $(\simeq;\mathcal{R};\simeq) \cup \simeq$ is a branching bisimulation. Suppose $M \simeq P\sigma\mathcal{R}Q\sigma \simeq N$ and $M \stackrel{a}{\longrightarrow} M'$. Then $P\sigma \stackrel{\epsilon}{\longrightarrow} P_1\sigma \stackrel{\epsilon}{\longrightarrow} \ldots \stackrel{\epsilon}{\longrightarrow} P_i\sigma \stackrel{a}{\longrightarrow} P'\sigma$ bisimulates $M \stackrel{a}{\longrightarrow} M'$ for some $P_1, \ldots, P_i, P'$. By the ϵ-pushing property $|P_1| > 0, \ldots, |P_i| > 0$. Thus $PV \stackrel{\tau}{\longrightarrow} P_1 V \stackrel{\tau}{\longrightarrow} \ldots \stackrel{\tau}{\longrightarrow} P_i V \stackrel{a}{\longrightarrow} P'V$. Since $PV \simeq QV$ this action sequence must be bisimulated by some $QV \xrightarrow{\tau} Q_1V \xrightarrow{\tau} \ldots \xrightarrow{\tau} Q_jV \xrightarrow{a} Q'V$ for some $Q_1, \ldots, Q_j, Q'$. Also $Q\sigma \xrightarrow{\tau} Q_1\sigma \xrightarrow{\tau} \ldots \xrightarrow{\tau} Q_j\sigma \xrightarrow{a} Q'\sigma$ must be bisimulated by $N \xrightarrow{\tau} N_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} N_k \xrightarrow{a} N'$ for some $N_1, \ldots, N_j, N'$. It is easy to see that $(M, N_1) \in \simeq;\mathcal{R};\simeq$, $\ldots$, $(M, N_k), (M', N') \in \simeq;\mathcal{R};\simeq$. Finally observe that $P' = p\epsilon$ if and only if $Q' = p\epsilon$. □

Let's define how to grow a bisimulation tree of $P\sigma \simeq Q\sigma$ over σ level by level with the initial $\mathcal{G}, \mathcal{E}, V$. Two things are worth spelling out. Firstly we will consider all bisimilar pairs between the descendants of $P\sigma$ and the descendants of $Q\sigma$ with intact σ. Secondly since all transitions are ϵ-pushing, the following construction never gets stuck along ϵ-edges. The growth of a node labeled by $(p_i\sigma, N\sigma)$, for $N \neq p_i$, is defined as follows.

1) $i \notin \mathcal{E}$. Relabel the node by $(G_i\sigma, N\sigma)$ and grow the node.
2) $i \in \mathcal{E}$, and $|N| > 0$ or $N = p_j\epsilon$ for some $j \notin \mathcal{E}$. Update $\mathcal{G}$ by letting $G_h = N$, respectively $G_h = G_j$, for every h in the equivalence class of i, remove the equivalence class of i from $\mathcal{E}$, relabel the node by $(N\sigma, N\sigma)$, respectively $(G_j\sigma, G_j\sigma)$, and stop growing the node.
3) $i \in \mathcal{E}$ and $N = p_j\epsilon$ for some $j \in \mathcal{E}$. Update $\mathcal{E}$ by joining the equivalence class of i with that of j, relabel the node by $(p_{\min\{i,j\}}\sigma, p_{\min\{i,j\}}\sigma)$, and stop growing the node.

The growth of a node labeled $(N\sigma, p_i\sigma)$ is symmetric. Each time $\mathcal{G}$ or $\mathcal{E}$ has been modified, we check if the semantic equivalence $PV \simeq QV$ holds. If $PV \not\simeq QV$ then there must be a least i such that the construction of the bisimulation tree for (PV, QV) gets stuck at the i-th level. When this happens further update of $\mathcal{G}$ and/or $\mathcal{E}$ can be carried out. After a finite number of levels both $\mathcal{G}$ and $\mathcal{E}$ must stabilize and $PV \simeq QV$ must hold.

We call the final $\mathcal{G} = \{o_i\gamma_i\}_{i\in[\mathfrak{q}]}$ a *recursive guard*, and

$$p_i\sigma \simeq o_i\gamma_i\sigma \tag{3}$$

a *recursive subgoal*. We say that the goal $P\sigma \simeq Q\sigma$ over σ has the recursive guard $\mathcal{G}$ and that it is decomposed into the subgoals $p_1\sigma \simeq o_i\gamma_1\sigma$, $\ldots$, $p_{\mathfrak{q}}\sigma \simeq o_i\gamma_{\mathfrak{q}}\sigma$. Let k be the minimal number such that the recursive guard $\mathcal{G}$ of (P, Q) over σ is generated by the k-subtree of the bisimulation tree of $P\sigma \simeq Q\sigma$ over σ. We call this subtree the *generating tree* for $\mathcal{G}$.

A recursive guard for P, Q is a recursive guard of $P\sigma \simeq Q\sigma$ over some σ. The following observation is crucial to the decidability argument.

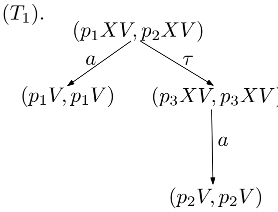
Lemma 10. *The number of recursive guards for P, Q is finite.*

Proof. The initial V does not depend on any suffix. Suppose at a particular stage there is a minimal i such that the construction of the bisimulation tree for (PV, QV) gets stuck at level i. Due to Lemma 5 there are only finitely many ways to update V and the maximal number of ways to do that is dependent of P and Q and is independent of any suffix. We are done by induction. □

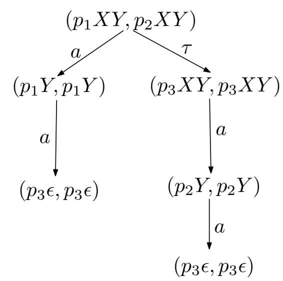
Now construct a bisimulation tree $\mathfrak{B}(PV, QV)$ of $PV \simeq QV$. Using the grammar equalities in (2) one realizes that this is also a bisimulation tree of $PV \simeq QV$ over V. By Lemma 9 one gets from the latter tree a bisimulation tree of $P\sigma \simeq Q\sigma$ over σ if V is replaced by σ. This tree is in general finer than any bisimulation tree of $P\sigma \simeq Q\sigma$ over σ in the sense that two nodes in the former may be collapsed into one node in the former. An example is described in the following diagram. For the PDA defined in the diagram T_1 and T_3 are bisimulation trees of $p_1XY \simeq p_2XY$ (over ϵ), while T_1 is the bisimulation tree of $p_1V \simeq p_2V$.

- $p_1X \xrightarrow{a} p_1,\ p_1X \xrightarrow{\tau} p_3X$;
- $p_2X \xrightarrow{a} p_1,\ p_2X \xrightarrow{\tau} p_3X$;
- $p_1Y \xrightarrow{a} p_3,\ p_2Y \xrightarrow{a} p_3$;
- $p_3X \xrightarrow{a} p_2$.
- $V = (p_1\epsilon, p_2\epsilon, p_3\epsilon)$.

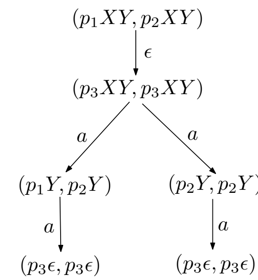
(T_1).



(T_2).



(T_3).



The recursive stack V defined in the above construction is not unique since in the middle of the construction any effort to build up the bisimulation tree of $PV \simeq QV$ over V may get stuck at the i-level for different pairs. Suppose a different set of choices produces a different recursive stack V'. Since we have considered all bisimilar pairs between the descendants of $P\sigma$ and the descendants of $Q\sigma$ with intact σ, it must be the case that $p_iV' \simeq o_i\gamma_iV'$ for all $i \in [\mathfrak{q}]$. We derive from Lemma 9 that $LV' \simeq NV'$ implies $LV \simeq NV$ whenever $|L|, |N| > 0$. Using symmetric argument we derives that $LV \simeq NV$ implies $LV' \simeq NV'$ whenever $|L|, |N| > 0$. That brings us to an important observation: For fixed P, Q, the bisimulation tree of $PV \simeq QV$ is unique for all possible recursive stacks produced in the construction of the tree. The *characteristic tree* of $P\sigma \simeq Q\sigma$ over σ, denoted by $\chi_\sigma(P, Q)$, is the unique bisimulation tree of $P\sigma \simeq Q\sigma$ over σ obtained from the unique $\mathfrak{B}(PV, QV)$ by substituting σ for V. We will focus on the characteristic trees exclusively in algorithmic study.

We now explain how to make use of $\chi_\sigma(P, Q)$. Suppose $\{p_i\sigma \simeq P_i\sigma\}_{i\in[\mathfrak{q}]}$ are the set of subgoals generated in the construction of $\chi_\sigma(P, Q)$. Let $\mathcal{B} = \bigcup_{k\in\omega} \mathcal{B}_k$, where $\mathcal{B}_k$ is defined as follows:

1) $\mathcal{B}_0 = \bigcup_{i\in[\mathfrak{q}]} \overline{\mathfrak{B}(p_i\sigma, P_i\sigma)}$.
2) $\mathcal{B}_{k+1} = \mathcal{B}_k \cup \mathcal{B}_k;\mathcal{B}_0$.

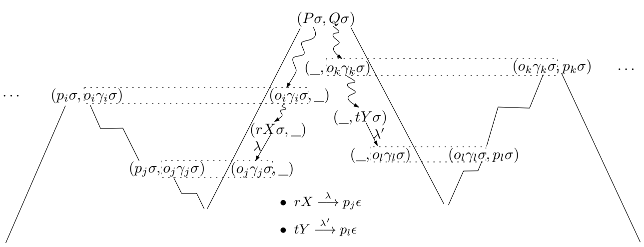
Since bisimulation is closed under union and composition the relation $\mathcal{B}$ as well as the relations $\mathcal{B}_k$ for $k \geq 0$ is a branching bisimulations. Define $\mathcal{B}(P\sigma, Q\sigma)$ by

$$\mathcal{B}(P\sigma, Q\sigma) \quad = \quad (\mathcal{B} \cup \mathcal{I});\overline{\chi_\sigma(P, Q)};(\mathcal{B}^{-1} \cup \mathcal{I}),$$

where $\mathcal{I}$ is the identity relation on PDA$^{\epsilon+}$ processes.

Lemma 11. *$\mathcal{B}(P\sigma, Q\sigma)$ is a branching bisimulation.*

Proof. The relation $\overline{\chi_\sigma(P,Q)}$ is not a branching bisimulation for two reasons. Firstly it may contain pairs of the form $(p_i\sigma, p_i\sigma)$ which is a leaf in $\chi_\sigma(P,Q)$. This is taken care of by $\mathcal{I}$. Secondly $\overline{\chi_\sigma(P,Q)}$ may contain a pair $(o_i\gamma_i\sigma, N\sigma)$ obtained from $(p_i\sigma, N\sigma)$ or a pair $(M\sigma, o_i\gamma_i\sigma)$ obtained from $(M\sigma, p_i\sigma)$. But notice that for each $i \in [\mathfrak{q}]$ the relation $\mathcal{B}$ contains a branching bisimulation for $p_i\sigma \simeq o_i\gamma_i\sigma$ and the relation $\mathcal{B}^{-1}$ contains a branching bisimulation for $o_i\gamma_i\sigma \simeq p_i\sigma$. Therefore the pairs of the form $(p_i\sigma, N\sigma)$ and of the form $(M\sigma, p_i\sigma)$ for $i \in [\mathfrak{q}]$ enjoy the bisimulation property by composition. A diagrammatic illustration of the composition is given below. The middle tree is $\overline{\chi_\sigma(P,Q)}$. The left and right trees are part of $\mathcal{B}$ and $\mathcal{B}^{-1}$ respectively. A dot rectangle indicates the point a composition starts. □



We conclude that to prove the goal $P\sigma \simeq Q\sigma$ we only have to construct the characteristic tree of $P\sigma \simeq Q\sigma$ over σ and the bisimulation trees for the subgoals generated therein.

B. Extending Common Suffix

Suppose $pX\alpha\sigma \simeq M\delta\sigma$, $|M| = \mathfrak{m}$ and $|\delta| > 0$. If $i \in \ker\|pX\|$ we let $pX\alpha\sigma \to^* \xrightarrow{J_1} \to^* \ldots \to^* \xrightarrow{J_k} \to^* p_i\alpha\sigma$ be a sequence reaching $p_i\alpha\sigma$ with minimal k. Since $\simeq$ is closed under composition, k cannot be greater than $\mathfrak{m}$. The above action sequence must be bisimulated by $M\delta\sigma$ in the following manner:

$$M\delta\sigma \to^* \xrightarrow{J_1} M_1\delta\sigma \to^* \xrightarrow{J_2} M_2\delta\sigma \ldots \to^* \xrightarrow{J_k} s_i\kappa_i\delta\sigma.$$

Since M is thick enough as it were, $\delta\sigma$ remains intact in the above transitions. We get for every $i \in \ker\|pX\|$ the subgoal

$$p_i\alpha\sigma \simeq s_i\kappa_i\delta\sigma. \tag{4}$$

We call (4) a *simple subgoal* and the family $\{s_i\kappa_i\}_{i\in\ker\|pX\|}$ a *simple guard*. By Lemma 5 the length of the simulating sequence is bounded by $\mathfrak{q}\mathfrak{n}\mathfrak{r}(\mathfrak{m}+1)^{\mathfrak{q}}\mathfrak{m} < \mathfrak{q}\mathfrak{n}\mathfrak{r}(\mathfrak{m}+1)^{(\mathfrak{q}+1)}$. It follows easily that $|\kappa_i| < \mathfrak{q}\mathfrak{n}\mathfrak{r}^2(\mathfrak{m}+1)^{(\mathfrak{q}+1)}$. Here is a consequence of the size bound on κ_i.

Lemma 12. *There are only finitely many simple guards.*

Stirling introduced a meta symbol U, called simple constant, with the following grammar equalities

$$p_iU = \begin{cases} s_i\kappa_i, & \text{if } i \in \ker\|pX\|, \\ p_i\epsilon, & \text{if } i \notin \ker\|pX\|. \end{cases}$$

Using the simple constant and the equivalence (4) one can turn the subgoal $pX\alpha\sigma \simeq M\delta\sigma$ into the subgoal $pXU\delta\sigma \simeq M\delta\sigma$, extending the size of the common suffix. The introduction of U allows us to think of $PX\alpha\sigma$ and $pXU\delta\sigma$ as if they were grammatically equal and that $\delta\sigma$ were the common suffix of $PX\alpha\sigma$ and $M\delta\sigma$.

Let V be the recursive stack generated in the definition of $\chi_{\delta\sigma}(pXU, M)$. We remark that the subgoals generated in the construction of $\chi_{\delta\sigma}(pXU, M)$ could be of two type. The first is of the type

$$o_i\gamma_i\delta\sigma \simeq p_i\delta\sigma, \tag{5}$$

which is of the reversal shape of the one given in (3). The second is of the type $p_j\delta\sigma \simeq r_j\lambda_jU\delta\sigma$, which is equivalent to

$$r_j\lambda_j\alpha\sigma \simeq p_j\delta\sigma. \tag{6}$$

We will see that the order is reversed for algorithmic reason. Let $\chi_{\alpha,\delta\sigma}(pX, M)$ be obtained from $\chi_{\delta\sigma}(pXU, M)$ by replacing $U\delta\sigma$ in the latter by $\alpha\sigma$. Strictly speaking $\chi_{\alpha,\delta\sigma}(pX, M)$ is not unique because more than one simple guard may rendering (4) true. We could introduce a canonical representation of every simple constant. We shall however not do that in this paper. We say that $\chi_{\alpha,\delta\sigma}(pX, M)$ and $\chi_{\alpha',\delta'\sigma'}(pX, M)$ are of same *type* if they are obtained from the same bisimulation tree $\mathfrak{B}(pXUV, MV)$ of $pXUV \simeq MV$. We also say that the trees $\chi_{\alpha,\delta\sigma}(pX, M)$, $\chi_{\alpha',\delta'\sigma'}(pX, M)$ are *duplicate* of each other.

A *mixed recursive guard* consists of $\mathfrak{q}$ processes, each being a process $o_i\gamma_i$ that appears in a subgoal of the shape (5) or a process $r_j\lambda_j$ that appears in a subgoal of the shape (6).

Lemma 13. *There are only finitely many mixed recursive guards for fixed pX and M.*

Proof. By Lemma 12 there are only finitely many simple constants. For each such constant there are finitely many mixed recursive stacks by recycling the proof of Lemma 10. □

For each $k \in \ker\|pX\|$ let $\mathfrak{B}^k$ be a bisimulation tree of $p_k\alpha\sigma \simeq s_k\kappa_k\delta\sigma$. Suppose $\mathfrak{q}_0, \mathfrak{q}_1$ are such that $\mathfrak{q}_0 \cup \mathfrak{q}_1 = \mathfrak{q}$ and $\{o_i\gamma_i\delta\sigma \simeq p_i\delta\sigma\}_{i\in[\mathfrak{q}_0]} \cup \{r_j\lambda_j\alpha\sigma \simeq p_j\delta\sigma\}_{j\in[\mathfrak{q}_1]}$ are the recursive subgoals. Let $\{\mathfrak{B}_0^i\}_{i\in[\mathfrak{q}_0]} \cup \{\mathfrak{B}_1^j\}_{j\in[\mathfrak{q}_1]}$ be the bisimulation trees of these goals. Let $\mathcal{E} = \bigcup_{k\in\ker\|pX\|} \overline{\mathfrak{B}^k} \cup \bigcup_{k\in\omega} \mathcal{E}_k$, where $\mathcal{E}_k$ is defined inductively by the following two clauses.

1) $\mathcal{E}_0 = \left(\bigcup_{i\in[\mathfrak{q}_0]} \overline{\mathfrak{B}_0^i}\right)^{-1} \cup \left(\bigcup_{j\in[\mathfrak{q}_1]} \overline{\mathfrak{B}_1^j}\right)^{-1}$.
2) $\mathcal{E}_{k+1} = \mathcal{E}_k \cup \mathcal{E}_k; \mathcal{E}_0$.

Define $\mathcal{E}(pX\alpha\sigma, M\delta\sigma)$ as follows:

$$\mathcal{E}(pX\alpha\sigma, M\delta\sigma) = (\mathcal{E} \cup \mathcal{I}); \overline{\chi_{\alpha,\delta\sigma}(pX, M)}; (\mathcal{E}^{-1} \cup \mathcal{I}).$$

Using again the fact that bisimulations are closed under composition and union one sees that $\mathcal{E}$ and $\mathcal{E}_i$ for all $i \geq 0$ are branching bisimulations.

Lemma 14. *$\mathcal{E}(pX\alpha\sigma, M\delta\sigma)$ is a branching bisimulation.*

Proof. The argument is similar to the one for Lemma 11. The treatment of a pair of the form $(s_i\kappa_i\delta\sigma, N)$ obtained from the pair $(p_i\alpha\sigma, N)$ is by simple composition. □

Every nontrivial goal is of the form $pX\alpha\sigma \simeq M\delta\sigma$ with $|M\delta\sigma| > 0$. What we have shown is that a bisimulation tree of a nontrivial goal can be decomposed to a finite number of bisimulation trees of subgoals and a characteristic tree.

Stirling also pointed out in [30] a special case of Lemma 13.

Lemma 15. *For fixed pX and M with $|M| = \mathfrak{m}$, there are a finite number of recursive guards $\{o_i^j\gamma_i^j\}_{i\in[\mathfrak{q}],j\in[k]}$ such that for every pair α,σ satisfying $pX\alpha\sigma \simeq M\sigma$ there is some $j \in [k]$ rendering true the following.*

$$o_i^j\gamma_i^j\sigma \simeq p_i\sigma, \tag{7}$$

$$pX\alpha V_j \simeq MV_j, \tag{8}$$

where V_j is defined by $p_iV_j = o_i^j\gamma_i^jV_j$ for all $i \in [\mathfrak{q}]$.

Proof. The number of pairs α,σ rendering $pX\alpha\sigma \simeq M\sigma$ could be infinite. But there are finitely many simple constant U such that $pXU\sigma \simeq pX\alpha\sigma \simeq M\sigma$. For each such simple constant U there are only a finite number of V such that $pXUV \simeq MV$. If $pX\alpha V \simeq pXUV$ then $pX\alpha V \simeq MV$. Otherwise we can use the method in Section VI-A to extend V to some V' such that $p_i\alpha V' \simeq p_iUV'$ for all $i \in \ker(pX)$ by exploiting the fact that $p_i\alpha\sigma \simeq p_iU\sigma$ for all $i \in \ker(pX)$. Now $p_i\alpha V' \simeq p_iUV'$ for all $i \in \ker(pX)$ implies $pX\alpha V' \simeq pXUV'$. And $pXUV \simeq MV$ implies $pXUV' \simeq MV'$ by Lemma 9. Hence $pX\alpha V' \simeq MV'$. There are only a finite number of ways to extend V to V'. □

The algorithmic significance of Lemma 15 is that the goal $pX\alpha\sigma \simeq M\sigma$ can be reduced to the characteristic tree of $pX\alpha\sigma \simeq M\sigma$ and a number of recursive subgoals. We will see that this will give rise to a self-reduction strategy since no simple subgoal is generated.

VII. Generic Bisimulation Tree

In this section we complete the process of cutting down the size of bisimulaiton trees by introducing conditions for nodes not to grow. Here are two obvious conditions.

1) If the label of a node is the same label as an ancestor, the node stops to grow.
2) If the label of a node is a pair of identical processes, the node stops to grow.

The result of Section VI-B implies that there is a bound $\mathfrak{c}$ such that the following property holds of all nontrivial goals of the form $pX\alpha\sigma \simeq M\delta\sigma$ such that $|M| = \mathfrak{m}$ and $|\delta\sigma| \geq \mathfrak{c}$: There is a goal $pX\alpha'\sigma' \simeq M\delta'\sigma'$ with $0 < |\delta'\sigma'| < \mathfrak{c}$ such that $\chi_{\alpha,\delta\sigma}(pX, M)$ and $\chi_{\alpha',\delta'\sigma'}(pX, M)$ are of same type. It follows that we only have to consider two types of goals.

1) The goals $L \simeq M$ are such that $|M| \leq \mathfrak{m}$. Grow a bisimulation tree of $L \simeq M$ as is defined in Section V. When a node labeled (L', M') is generated such that $|M'| \geq \mathfrak{m} + \mathfrak{c}$, the node stops growing.
2) The goals $pX\alpha\sigma \simeq M\delta\sigma$ are such that $|M| = \mathfrak{m}$ and $0 < |\delta\sigma| < \mathfrak{c}$. Grow the characteristic tree of $pX\alpha\sigma \simeq M\delta\sigma$ over $\delta\sigma$ with the following additional constraints: If a node labeled (L', M') is generated such that $|M'| \geq \mathfrak{m}+\mathfrak{c}$, the node stops growing. We call the leaf a *large leaf*.

For the above construction to make sense, we should choose the bound $\mathfrak{c}$ such that

1) it is larger than the size of all simple and recursive guards, and
2) it is larger than the height of all the generating trees for the recursive guards.

We call the goals of the two types *generic goals* and the characteristic trees of these goals *generic bisimulation trees*. By definition and Corollary 6 the set of generic goals as well as the set of generic bisimulation trees is finite. Moreover we have the following important fact.

Lemma 16. *Every generic bisimulation tree is finite.*

Proof. The generic bisimulation trees of the first type is obviously finite. In the light of Corollary 6 if in a path no large leaf is ever generated, repeat must occur. We are done by applying König Lemma. □

Suppose $q_1X_1\alpha_1\sigma_1 \simeq M_1\delta_1\sigma_1, \ldots, q_gX_g\alpha_g\sigma_g \simeq M_g\delta_g\sigma_g$ are the generic goals and $T_{q_1X_1\alpha_1\sigma_1\simeq M_1\delta_1\sigma_1}, \ldots, T_{q_gX_g\alpha_g\sigma_g\simeq M_g\delta_g\sigma_g}$ are the corresponding generic bisimulation trees. We say that a goal $q_iX_i\alpha\sigma \simeq M_i\delta\sigma$ with $|\delta\sigma| \geq \mathfrak{c}$ is *of type i* if $\chi_{\alpha,\delta\sigma}(q_iX_i, M_i)$ and $\chi_{\alpha_i,\delta_i\sigma_i}(q_iX_i, M_i)$ are of same type. For each $i \in [g]$ let

$$\mathcal{B}_i = \bigcup_{q_iX_i\alpha\sigma\simeq M_i\delta\sigma \text{ is of type } i} \overline{T_{q_iX_i\alpha_i\sigma_i\simeq M_i\delta_i\sigma_i}}\{\alpha/\alpha_i, \delta/\delta_i, \sigma/\sigma_i\},$$

where the relation $\overline{T_{q_iX_i\alpha_i\sigma_i\simeq M_i\delta_i\sigma_i}}\{\alpha/\alpha_i, \delta/\delta_i, \sigma/\sigma_i\}$ is obtained from $\overline{T_{q_iX_i\alpha_i\sigma_i\simeq M_i\delta_i\sigma_i}}$ by substituting α for α_i, δ for δ_i and σ_i for σ. Let

$$\mathcal{BB} = \bigcup_{i\in[g]} \mathcal{B}_i.$$

Finally let $\mathcal{BB}^* = \left(\mathcal{I} \cup \mathcal{BB} \cup \mathcal{BB}^{-1}\right)^*$.

Proposition 17. *$\mathcal{BB}^*$ is a branching bisimulation.*

Proof. $\overline{T_{q_iX_i\alpha_i\sigma_i\simeq M_i\delta_i\sigma_i}}$ and $\overline{T_{q_iX_i\alpha_i\sigma_i\simeq M_i\delta_i\sigma_i}}\{\alpha/\alpha_i, \delta/\delta_i, \sigma/\sigma_i\}$ for $i \in [g]$ are in general not branching bisimulations. We can piece these finite trees in two ways. Vertically we can graft a duplicate of a generic bisimulation tree on a large leaf of a generic bisimulation tree. This procedure can be carried out ad infinitum. What we get eventually are bisimulation trees of generic goals over suffix. Now horizontally we can compose the bisimulation trees of generic goals over suffix to form bisimulation trees. □

VIII. Bisimulation Base

From an algorithmic point of view it is insufficient to know the *existence* of the set of generic bisimulation trees defined in Section VII. More useful is a set of generic bisimulation trees, not necessarily complete, that enjoys certain closure property with regards to decomposition. Suppose $r_1Y_1\alpha_1\sigma_1 \simeq M_1\delta_1\sigma_1, \ldots, r_hY_h\alpha_h\sigma_h \simeq M_h\delta_h\sigma_h$ are the generic goals and $T_{r_1Y_1\alpha_1\sigma_1\simeq M_1\delta_1\sigma_1}, \ldots, T_{r_hY_h\alpha_h\sigma_h\simeq M_h\delta_h\sigma_h}$ are the corresponding generic bisimulation trees. We require that the following closure property hold of these generic bisimulation trees.

1) For every $i \in [h]$ and every large leaf of $T_{r_iY_i\alpha_i\sigma_i\simeq M_i\delta_i\sigma_i}$ is of some type $j \in [h]$.

1) Guess a set $\mathfrak{G}$ of generic goals $L \simeq N$ with $|N| < \mathfrak{m} + \mathfrak{c}$.
2) For every guessed generic goal $L \simeq N$ with $|N| \leq \mathfrak{m}$, guess a generic bisimulation tree. If an internal node of the tree fails the bisimulation property, report a failure.
3) For every guessed generic goal $pX\alpha\sigma \simeq M\delta\sigma$ such that $|M| = \mathfrak{m}$ and $|\delta| > 0$, do the following.
   a) Guess the characteristic tree of the goal with a simple guard and a recursive guard. The size of the simple and recursive guards and the height of the generating tree are bounded by $\mathfrak{c}$.
      i) Check if every internal node of the guessed bisimulation/chararcteristic tree satisfies the bisimulation property. If not, report failure.
      ii) For every guessed simple subgoal, if the size of the right hand side is less than $\mathfrak{m} + \mathfrak{c}$, then check if it is in $\mathfrak{G}$, otherwise goto Step 3a.
      iii) For every guessed recursive subgoal $L\sigma \simeq p\sigma$, if $|\sigma| < \mathfrak{m} + \mathfrak{c}$, then check if it is in $\mathfrak{G}$, otherwise let $\sigma'\sigma'' = \sigma$ and $|\sigma'| = \mathfrak{m}$ and do the following.
         A) Guess the characteristic tree of $L\sigma'\sigma'' \simeq p\sigma'\sigma''$ together with a recursive guard. The size of the recursive guard and the height of the generating tree are bounded by $\mathfrak{c}$.
         B) Check if every internal node of the guessed chararcteristic tree satisfies the bisimulation property. If not, report failure.
         C) Goto Step 3(a)iii to check the recursive subgoals generated in Step 3(a)iiiA.
4) For every leaf (L, O) of a guessed generic bisimulation tree do the following.
   a) If $|O| < \mathfrak{m} + \mathfrak{c}$, pass if $L = O$ or if (L, O) is the label of an ancestor, otherwise report a failure.
   b) If $|O| \geq \mathfrak{m} + \mathfrak{c}$, guess that $L \simeq N$ is of the same type as some guessed generic goal $pX\alpha\sigma \simeq M\delta\sigma$. Generate new subgoals by using the simple guard and recursive guard of the characteristic tree of $pX\alpha\sigma \simeq M\delta\sigma$. Apply the checks defined in Step 3(a)ii and Step 3(a)iii to the new simple subgoals and the new recursive subgoals respectively.
5) If there is no report of failure, output $\mathfrak{G}$.

Fig. 1. ClosedSet ($\mathfrak{c}$)

2) Every subgoal generated in the construction of any generic bisimulation tree $T_{r_k Y_k \alpha_k \sigma_k \simeq M_k \delta_k \sigma_k}$, where $k \in [h]$, is of some type $j \in [h]$.
3) Every subgoal generated in the construction of the characteristic tree of any of the subgoals generated in (2) is of some type $j \in [h]$.
4) Every subgoal generated in the construction of the characteristic tree of any of the subgoals generated in (3) is of some type $j \in [h]$.
5) So on and so forth.

1) If $|Q| < \mathfrak{m} + \mathfrak{c}$, check if $P \simeq Q \in \mathfrak{G}$.
2) If $|Q| \geq \mathfrak{m} + \mathfrak{c}$, then guess the type of the goal $P \simeq Q$ and apply Decomposable$_{\mathfrak{G}}$ to each of the subgoals.

Fig. 2. Decomposable$_{\mathfrak{G}}$ (P, Q)

In the above process the construction of a recursive subgoal of the form $o_i\gamma_i\delta\sigma \simeq p_i\delta\sigma$ no simple guard need be introduced. This is because the subgoal is of the form $o_iX'\alpha'\sigma' \simeq M'\sigma'$ such that $X'\alpha'\sigma' = \gamma_i\delta\sigma$, $M'\sigma' = p_i\delta\sigma$ and $|M'| = \mathfrak{m}$. So Lemma 15 applies. The common suffix σ' satisfies $|\sigma'| < |\delta\sigma|$. This leads to two observations. Firstly there are only a finite number of simple subgoals generated in the above procedure. Every time a new simple subgoal is produced the size of its left hand side is smaller than the size of the left hand side of the simple subgoals already produced. Secondly recursive treatment of the recursive goals does not introduce any simple subgoals. Every time a new recursive subgoal is produced the size of its right hand side is smaller than the size of the right hand side of one recursive subgoal already produced. It follows easily from these observations that the procedure defined in the above terminates in finite steps. If the procedure ends in success, we say that the set of the generic goals and the set of the generic bisimulation trees are *closed*.

We shall describe an algorithm that can check if within a given size bound a closed set of generic subgoals, as well as the corresponding closed set of the generic bisimulation trees, exists; and if they exist, output them. This is the nondeterministic algorithm ClosedSet($\mathfrak{c}$) defined in Fig. 1. By Corollary 6 the guess in Step 1 is a bounded guess. Since the size of the right hand side of every node of every guessed generic bisimulation tree is bounded by $\mathfrak{m} + \mathfrak{c} + \mathfrak{r}$, by Lemma 5 and Corollary 6 there is a bound on the size of the guessed generic bisimulation trees, which is computable from the definition of PDA and the bound $\mathfrak{c}$. Thus ClosedSet($\mathfrak{c}$) terminates for every set of guesses. Since all the guesses are computationally bounded, ClosedSet($\mathfrak{c}$) can be turned to a deterministic algorithm. We remark that even if ClosedSet($\mathfrak{c}$) terminates successfully, it does not mean that ClosedSet($\mathfrak{c}$) has found out every generic goal. If the bound $\mathfrak{c}$ is not large enough the algorithm cannot discover all the generic goals.

Next we design a nondeterministic algorithm, defined in Fig. 2 that checks if a goal can be decomposed in terms of the elements of a given closed set $\mathfrak{C}$ of generic subgoals. The description of Step 2 is greatly simplified. Had we written down all the details it would look very much the same as the main body of ClosedSet ($\mathfrak{c}$). An input pair (P, Q) with $|Q| \geq \mathfrak{m} + \mathfrak{c}$ is processed in the same way a large leaf is processed in ClosedSet ($\mathfrak{c}$). This algorithm certainly terminates since the size of every subgoal is controlled. We say that a goal $P \simeq Q$ is *decomposable with regards to* $\mathfrak{C}$ if Decomposable$_{\mathfrak{C}}$ (P, Q) returns true.

Closed sets of generic goals (bisimulation trees) are still insufficient. We need an even stronger closure property that requires a generic bisimulation tree to be extensible. Let $\mathfrak{G} =$

1) Let $\mathfrak{BB} = \emptyset$.
2) For $i = 1$ to h, do the following.
   a) Let $T = T_{r_i Y_i \alpha_i \sigma_i \simeq M_i \delta_i \sigma_i}$.
   b) If there is a large leaf of T of type say j that is a large leaf of a duplicate of say type k, and there is no node in the path to the root that is also a large leaf of type j of a duplicate of type k, grow a duplicate of $T_{r_j Y_j \alpha_j \sigma_j \simeq M_j \delta_j \sigma_j}$ at this large leaf, check if every subgoal generated therein is decomposable with regards to $\mathfrak{T}$. If any of the subgoals is not decomposable, report failure and exit.
   c) Let $\mathfrak{BB} = \mathfrak{BB} \cup \{T\}$.
3) Output $\mathfrak{BB}$.

Fig. 3. BisimulationBase ($\mathfrak{T}$)

$\{r_1 Y_1 \alpha_1 \sigma_1 \simeq M_1 \delta_1 \sigma_1, \ldots, r_h Y_h \alpha_h \sigma_h \simeq M_h \delta_h \sigma_h\}$ be a closed set of goals, and let $\mathfrak{T} = \{T_{r_1 Y_1 \alpha_1 \sigma_1 \simeq M_1 \delta_1 \sigma_1}, \ldots, T_{r_h Y_h \alpha_h \sigma_h \simeq M_h \delta_h \sigma_h}\}$ be the set of the generic bisimulation trees. We say that $\mathfrak{T}$ is a *bisimulation base* if the algorithm BisimulationBase ($\mathfrak{T}$) terminates without reporting any failure. Notice that since there are finitely many generic bisimulation trees, the termination condition must be met on every path. It is easy to see that there is a computable bound on the height of the final tree. Consequently the size of subgoals generated during the generation of the final tree is controlled. In other words there is a computable bound on the number of subgoals one has to check. We conclude that BisimulationBase ($\mathfrak{T}$) must terminate. The algorithm either reports a failure or or confirms that the input is a bisimulation base. If BisimulationBase ($\mathfrak{T}$) terminates with an output set $\mathfrak{BB}$, we call an element of the set a *productive tree*. An illustration of a productive tree is given by the diagram in Fig. 4. In the diagram each triangle is a duplicate of a generic bisimulation tree. The two shaded triangles are duplicates of a same generic bisimulation tree. The two bullets are nodes in the same position of the respective duplicates and are of same type.

Now suppose $\mathfrak{T} = \{T_{r_1 Y_1 \alpha_1 \sigma_1 \simeq M_1 \delta_1 \sigma_1}, \ldots, T_{r_h Y_h \alpha_h \sigma_h \simeq M_h \delta_h \sigma_h}\}$ is a bisimulation base and $\mathfrak{P}_1, \ldots, \mathfrak{P}_h$ are the productive trees. For each $i \in [h]$ we can unfold the productive tree $\mathfrak{P}_i$ to a bisimulation tree of $r_i Y_i \alpha_i \sigma_i \simeq M_i \delta_i \sigma_i$. This is done as follows. The initial part of the bisimulation tree is obtained by composing $\mathfrak{P}_i$ with bisimulation trees of the subgoals generated in the definition of the characteristic tree of $r_i Y_i \alpha_i \sigma_i \simeq M_i \delta_i \sigma_i$. For each large leaf (L, N) of $\mathfrak{P}_i$, there is by definition an ancestor (L', N') of (L, N) such that the latter node is the root of a duplicate of some $\mathfrak{P}_j$ and the goals $L \simeq N$ and $L' \simeq N'$ are of same type. We can grow a bisimulation tree of $L \simeq N$ as a duplicate of a bisimulation tree of $L' \simeq N'$. The initial part of the bisimulation tree of $L' \simeq N'$ is grown by composing the shaded tree with bisimulation trees of the relevant subgoals. To continue we need to grow the large leaf (L, N) again, which can be done in completely the same manner as in the above. In conclusion we can turn all the productive trees to bisimulation trees of the respective generic goals $r_1 Y_1 \alpha_1 \sigma_1 \simeq M_1 \delta_1 \sigma_1, \ldots, r_h Y_h \alpha_h \sigma_h \simeq M_h \delta_h \sigma_h$.

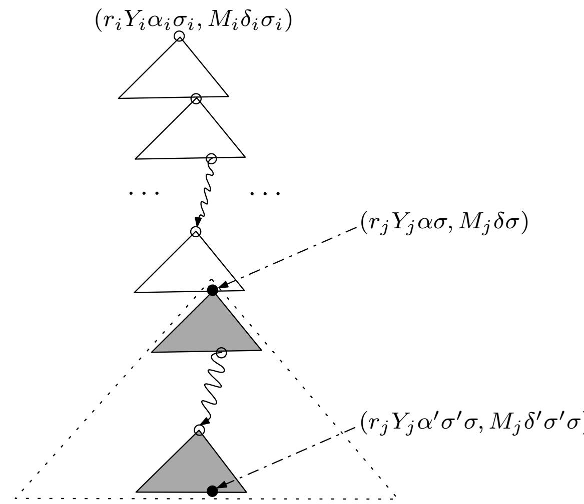


Fig. 4. Productive Tree

1) Check if $P \not\simeq Q$, and at the same time run in parallel ClosedSet($\mathfrak{m}$), ClosedSet($\mathfrak{m}+1$), ...
2) If $P \not\simeq Q$ then answer 'no'.
3) If for some $\mathfrak{c} \geq \mathfrak{m}$, ClosedSet($\mathfrak{c}$) terminates successfully with a closed set $\mathfrak{C}$ of generic bisimulation trees and if moreover BisimulationBase ($\mathfrak{C}$) is successful, then let $\mathfrak{BB}$ = BisimulationBase ($\mathfrak{C}$). If Decomposable$_{\mathfrak{C}}$ (P, Q) is successful with regards to $\mathfrak{BB}$, answer 'yes' and halt.

Fig. 5. Equivalence (P, Q)

In the above account the constructions of bisimulation trees of the subgoals is missing. We now remedy this. By definition every such subgoal is decomposable with regards to $\mathfrak{BB}$. We can grow a bisimulation tree of the subgoal in the way described in Section VI-A. During the construction we need to grow bisimulation trees of further subgoals in the same manner. Eventually the growth of the tree rely on the unfolding of the productive trees.

Proposition 18. *Suppose P, Q are* nPDA$^{\epsilon+}$ *processes. If a goal (P, Q) is decomposable with regards to a bisimulation base, then $P \simeq Q$.*

Proof. The above argument is sufficient. □

IX. Decision Algorithm

We are ready to give a decision algorithm. The algorithm Equivalence is defined in Fig. 5. The termination of the algorithm is clear in the light of Proposition 7. We have effectively proved the main result of the paper.

Theorem 19. *The relation $\simeq_{\mathrm{nPDA}^{\epsilon+}}$ is decidable.*

Proof. Since the goal $P \simeq Q$ is decomposable, we can construct a bisimulation in the same manner we have done for Proposition 18. □

X. High Undecidability of ϵ-Nondeterminism

In the proofs of this section we need to use the game theoretical interpretation of bisimulation. A *bisimulation game* [31], [18] for a pair of processes (P_0, P_1), called a *configuration*, is played between Attacker and Defender in an alternating fashion. It is played according to the following rules: Suppose (P_0, P_1) is the current configuration.

- $|P_i| > 0$ or $P_i = \mathbf{0}$ for each $i \in \{0, 1\}$.
  1) Attacker picks up some P_i, where $i \in \{0, 1\}$, to start with and chooses some $P_i \xrightarrow{\ell} P'_i$.
  2) Defender must respond in the following manner:
     a) Do nothing. This option is available if $\ell = \epsilon$.
     b) Choose a transition sequence $P_{1-i} \xrightarrow{\epsilon} P^1_{1-i} \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} P^{k-1}_{1-i} \xrightarrow{\ell} P^k_{1-i}$.
  3) If case 2(a) happens the new configuration is (P'_i, P_{1-i}). If case 2(b) happens Attacker chooses one of
  $$\{(P_i, P^1_{1-i}), \ldots, (P_i, P^{k-1}_{1-i}), (P'_i, P^k_{1-i})\}$$
  as the new configuration.
  4) The game continues with the new configuration.
- $P_i = p\epsilon$.
  1) Attacker chooses some $P_{i-1} \Longrightarrow P'_{i-1}$ for some P'_{i-1}.
  2) Defender must respond with $P'_{i-1} \Longrightarrow P_i$.

Attacker wins a bisimulation game if Defender gets stuck in the game. Defender wins a bisimulation game if Attacker cannot win the game. Attacker/Defender has a winning strategy if it can win no matter how its opponent plays. The effectiveness of the bisimulation game is enforced by the following lemma.

Lemma 20. *$P \simeq Q$ if and only if Defender has a winning strategy for the bisimulation game starting with the configuration (P, Q).*

The above lemma is the basis for game theoretical proofs of process equality. It is also the basis for game constructions using Defender's Forcing.

We will show that the branching bisimilarity is highly undecidable on PDA$^{\epsilon+}$. This is done by a reduction from a Σ^1_1-complete problem. A *nondeterministic Minsky counter machine* $\mathcal{M}$ with two counters c_1, c_2 is a program of the form $1 : I_1;\ 2 : I_2;\ \ldots;\ n{-}1 : I_{n-1};\ n$: halt, where for each $i \in \{1, \ldots, n-1\}$ the instruction I_i is in one of the following forms, assuming $1 \le j, k \le n$ and $e \in \{1, 2\}$.

- $c_e := c_e + 1$ and then goto j.
- if $c_e = 0$ then goto j; otherwise $c_e := c_e - 1$ and then goto k.
- goto j or goto k.

The problem rec-NMCM asks if $\mathcal{M}$ has an infinite computation on $(c_1, c_2) = (0, 0)$ such that I_1 is executed infinitely often. We shall use the following fact from [9].

Proposition 21. rec-NMCM *is Σ^1_1-complete.*

Following [18] we transform a nondeterministic Minsky counter machine $\mathcal{M}$ with two counters c_1, c_2 into a machine $\mathcal{M}'$ with three counters c_1, c_2, c_3. The machine $\mathcal{M}'$ makes use of a new nondeterministic instruction of the following form.

- $i : c_e := *$ and then goto j.

The effect of this instruction is to set c_e by a nondeterministically chosen number and then go to I_j. Every instruction "$i : I_i$" of $\mathcal{M}$ is then replaced by two instructions in $\mathcal{M}'$, with respective labels $2i{-}1$ and $2i$.

- $1 : I_1$ is replaced by
  $1 : c_3 := *$ and goto 2;
  $2 : I_1$.
- $i : I_i$, where $i \in \{2, \ldots, n\}$, is replaced by
  $2i - 1$: if $c_3 = 0$ then goto $2n$; otherwise $c_3 := c_3 - 1$ and goto $2i$;
  $2i : I_i$
- Inside each I_i, where $i \in \{1, \ldots, n\}$, every occurrence of "goto j" is replaced by "goto $2j - 1$".

It is easy to see that $\mathcal{M}'$ has an infinite computation if and only if $\mathcal{M}$ has an infinite computation that executes the instruction I_1 infinitely often. Our goal is to construct a PDA$^{\epsilon+}$ system $\mathcal{G} = \{\mathcal{Q}, \mathcal{L}, \mathcal{V}, \mathcal{R}\}$ in which we can define two processes $p_1X\bot$ and $q_1X\bot$ that render true the following equivalence.

$p_1X\bot \simeq q_1X\bot$ if and only if $\mathcal{M}'$ has an infinite computation.

The system $\mathcal{G} = \{\mathcal{Q}, \mathcal{L}, \mathcal{V}, \mathcal{R}\}$ contains the following key elements:

- Two states $p_i, q_i \in \mathcal{Q}$ are introduced for each instruction I_i.
- $\mathcal{L} = \{a, b, c, c_1, c_2, c_3, f, f'\}$.
- Three stack symbols $C_1, C_2, C_3 \in \mathcal{V}$ are introduced for the three counters respectively. A bottom symbol $\bot \in \mathcal{V}$ is also introduced.

Our construction borrows ideas from [19], [18], [36], making use of the game characterization of branching bisimulation and Defender's Forcing technique. A configuration of $\mathcal{M}'$ that consists of instruction label i and counter values $(c_1, c_2, c_3) = (n_1, n_2, n_3)$ is represented by the game configuration $(p_iXC_1^{n_1}C_2^{n_2}C_3^{n_3}\bot, q_iXC_1^{n_1}C_2^{n_2}C_3^{n_3}\bot)$. In the rest of the section we shall complete the definition of $\mathcal{G}$ and explain its working mechanism.

A. Test on Counter

We need some rules to carry out testing on the counters. In the rules given in Fig. 6, j and e range over the set $\{1, 2, 3\}$. These rules are straightforward. The following proposition summarizes the correctness requirement on the equality test, the successor and predecessor tests, and the zero test. Its routine proof is omitted.

Proposition 22. *Let $\alpha = C_1^{n_1}C_2^{n_2}C_3^{n_3}$ and $\beta = C_1^{m_1}C_2^{m_2}C_3^{m_3}$. The following statements are valid.*

1) $t\alpha\bot \simeq t\beta\bot$ *if and only if* $n_e = m_e$ *for* $e = 1, 2, 3$.
2) $t(3, *)\alpha\bot \simeq t'(3, *)\beta\bot$ *if and only if* $n_e = m_e$ *for* $e = 1, 2$.
3) $t(e, +)\alpha\bot \simeq t'(e, +)\beta\bot$ *if and only if* $n_e + 1 = m_e$ *and* $n_j = m_j$ *for* $j \neq e$.

- $tC_1 \xrightarrow{c_1} t$, $tC_2 \xrightarrow{c_2} t$, $tC_3 \xrightarrow{c_3} t$;
- $t(e,+)C_j \xrightarrow{c_j} t(e,+)$ if $j < e$, $t(e,+)C_j \xrightarrow{c_e} tC_j$ if $j \geq e$, $t(e,+)\bot \xrightarrow{c_e} t\bot$;
$t'(e,+)C_j \xrightarrow{c_j} t$;
- $t(e,*)C_1 \xrightarrow{c_1} t(e,*)$, $t(e,*)C_2 \xrightarrow{c_2} t(e,*)$, $t(e,*)C_3 \xrightarrow{b} t\bot$;
$t'(e,*)C_1 \xrightarrow{c_1} t(e,*)$, $t'(e,*)C_2 \xrightarrow{c_2} t(e,*)$, $t'(e,*)C_3 \xrightarrow{b} t\bot$;
- $t(e,-)C_j \xrightarrow{c_j} t$;
$t'(e,-)C_j \xrightarrow{c_j} t'(e,-)$ if $j < e$, $t'(e,-)C_j \xrightarrow{c_e} tC_j$ if $j \geq e$, $t'(e,-)\bot \xrightarrow{c_e} t\bot$;
- $t(e,0)C_j \xrightarrow{c_j} t(e,0)$ if $j \neq e$, $t(e,0)C_e \xrightarrow{f} t(e,0)$;
$t'(e,0)C_j \xrightarrow{c_j} t'(e,0)$ if $j \neq e$, $t'(e,0)C_e \xrightarrow{f'} t(e,0)$;
- $t(e,1)C_j \xrightarrow{c_j} t(e,1)$ if $j < e$, $t(e,1)C_e \xrightarrow{c_e} t$, $t(e,1)C_j \xrightarrow{f} t$ if $j > e$; $t(e,1)\bot \xrightarrow{f} t\bot$;
$t'(e,1)C_j \xrightarrow{c_j} t'(e,1)$ if $j < e$, $t'(e,1)C_e \xrightarrow{c_e} t$, $t'(e,1)C_j \xrightarrow{f'} t$ if $j > e$; $t'(e,1)\bot \xrightarrow{f'} t\bot$;
- $p\bot \xrightarrow{b} t\bot$ for every $p \in \{t, t', t(e,+), t'(e,+), t(e,-), t'(e,-), t(e,0), t'(e,0), t(e,1), t'(e,1)\}$.

Fig. 6. Test on Counter

4) *$t(e,-)\alpha\bot \simeq t'(e,-)\beta\bot$ if and only if $n_e = m_e + 1$ and $n_j = m_j$ for $j \neq e$.*
5) *$t(e,0)\alpha\bot \simeq t'(e,0)\beta\bot$ if and only if $n_j = m_j$ for $j = 1,2,3$ and $n_e = 0$.*
6) *$t(e,1)\alpha\bot \simeq t'(e,1)\beta\bot$ if and only if $n_j = m_j$ for $j = 1,2,3$ and $n_e > 0$.*
7) *$p\alpha\bot \simeq p\alpha\bot\beta$ for all $p \in Q$ and all $\alpha,\beta \in \mathcal{V}^*$.*

B. Operation on Counter

There are three basic operations on counters, the increment operation, the decrement operation and the nondeterministic assignment operation. Our task is to encode these operations in the branching bisimulation game $\mathcal{G}$. To do that we use a technique from [36], which is a refinement of Defender's Forcing technique [18], taking into account of the subtlety of the branching bisimulation. The idea can be explained using the following system.

1) $P \xrightarrow{a} P'$, $P \xrightarrow{\epsilon} Q_0$. The latter is the only silent transition of P.
2) $Q \xrightarrow{\epsilon} Q_0$. This is the only transition Q may perform. Hence $Q \simeq Q_0$.
3) $Q_0 \simeq \overline{Q}$ whenever $Q_0 \Longrightarrow \overline{Q}$.

Condition 1 and condition 2 guarantee that $P \simeq Q$ if and only if $P \simeq Q_0$. So the effectiveness of the Defender's Forcing the copycat rules $P \xrightarrow{\epsilon} Q_0$, $Q \xrightarrow{\epsilon} Q_0$ intend to achieve depends on how we define Q_0. Condition 3 is forced upon us by the previous two conditions. A standard approach to meet the requirement 3 is to make sure that everything that has been

- $u(e,o,j)X \xrightarrow{a} u_1(e,o,j)X$, $u(e,o,j)X \xrightarrow{\epsilon} r'(e,o,j)X$; $u'(e,o,j)X \xrightarrow{\epsilon} r'(e,o,j)X$;
- $r'(e,o,j)X \xrightarrow{\epsilon} g'(e,o,j)X\bot$;
$g'(e,o,j)X \xrightarrow{\epsilon} g'(e,o,j)X_3$;
$g'(e,o,j)X_3 \xrightarrow{\epsilon} g'(e,o,j)X_3C_3$, $g'(e,o,j)X_3 \xrightarrow{\epsilon} g'(e,o,j)X_2$;
$g'(e,o,j)X_2 \xrightarrow{\epsilon} g'(e,o,j)X_2C_2$, $g'(e,o,j)X_2 \xrightarrow{\epsilon} g'(e,o,j)X_1$;
$g'(e,o,j)X_1 \xrightarrow{\epsilon} g'(e,o,j)X_1C_1$, $g'(e,o,j)X_1 \xrightarrow{\epsilon} r'(e,o,j)X$;
- $g'(e,o,j)X_1 \xrightarrow{a} u'_1(e,o,j)X$;
- $u_1(e,o,j) \xrightarrow{a} u_2(e,o,j)X$, $u_1(e,o,j)X \xrightarrow{c} t(e,o)$;
$u'_1(e,o,j) \xrightarrow{a} u'_2(e,o,j)X$, $u'_1(e,o,j)X \xrightarrow{c} t'(e,o)$;
- $u_2(e,o,j)X \xrightarrow{\epsilon} r(e,o,j)X$;
$u'_2(e,o,j)X \xrightarrow{\epsilon} r(e,o,j)X$, $u'_2(e,o,j)X \xrightarrow{a} u'_3(e,o,j)X$;
- $r(e,o,j)X \xrightarrow{\epsilon} g(e,o,j)X\bot$; $g(e,o,j)X \xrightarrow{\epsilon} g(e,o,j)X_3$;
$g(e,o,j)X_3 \xrightarrow{\epsilon} g(e,o,j)X_3C_3$, $g(e,o,j)X_3 \xrightarrow{\epsilon} g(e,o,j)X_2$;
$g(e,o,j)X_2 \xrightarrow{\epsilon} g(e,o,j)X_2C_2$, $g(e,o,j)X_2 \xrightarrow{\epsilon} g(e,o,j)X_1$;
$g(e,o,j)X_1 \xrightarrow{\epsilon} g(e,o,j)X_1C_1$, $g(e,o,j)X_1 \xrightarrow{\epsilon} r(e,o,j)X$;
- $g(e,o,j)X_1 \xrightarrow{a} u_3(e,o,j)X$;
- $u_3(e,o,j)X \xrightarrow{a} p_jX$, $u_3(e,o,j)X \xrightarrow{c} t$;
$u'_3(e,o,j)X \xrightarrow{a} q_jX$, $u'_3(e,o,j)X \xrightarrow{c} t$.

Fig. 7. Operation on Counter

done to derive $Q_0 \Longrightarrow \overline{Q}$ can be undone. In our setting this is accomplished by starting all over again with the help of the bottom symbol $\bot$. Once we know that condition 3 is indeed satisfied, the argument for the correctness of the bisimulation game can be simplified in the following sense: In the game of (P,Q) Attacker would play $P \xrightarrow{a} P'$. Defender's optimal response must be of the following form

$$Q \xrightarrow{\epsilon} Q_0 \xrightarrow{\epsilon} Q_1 \xrightarrow{\epsilon} Q_2 \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} Q_k \xrightarrow{a} Q'.$$

For both players only the configuration (P', Q') need be checked.

With the above remark in mind we turn to the part of the game that implements the basic operations. Let e range over $\{1,2,3\}$, o over $\{+,-,*\}$, and j over $\{1,\ldots,2n\}$. For each triple (e,o,j) we introduce the rules given in Fig. 7. The following lemma identifies some useful state preserving silent transitions.

Lemma 23. *$P \simeq g(e,o,j)X\bot$ for all P such that $g(e,o,j)X\bot \Longrightarrow P$. Similarly $Q \simeq g'(e,o,j)X\bot$ for all Q such that $g'(e,o,j)X\bot \Longrightarrow Q$.*

Proof. Suppose $g(e,o,j)X\bot \Longrightarrow P$. Then $P \Longrightarrow g(e,o,j)X\bot\alpha$ for some α. By (7) of Proposition 22 one has $g(e,o,j)X\bot \simeq g(e,o,j)X\bot\alpha$. Consequently $g(e,o,j)X\bot \simeq P$. $\square$

The next lemma states the soundness property of the rules defined in Fig. 7, in which we write $\mathbf{1}^1$, $\mathbf{1}^2$ and $\mathbf{1}^3$ respectively for $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$.

Lemma 24. *Suppose $\alpha = C_1^{m_1}C_2^{m_2}C_3^{m_3}$. The following statements are valid.*

1) *In the bisimulation of $(u(e,+,j)X\alpha\bot, u'(e,+,j)X\alpha\bot)$, Defender, respectively Attacker, has a strategy to win or at least push the game to (P,Q) such that $P \simeq p_jXC_1^{n_1}C_2^{n_2}C_3^{n_3}\bot$ and $Q \simeq q_jXC_1^{n_1}C_2^{n_2}C_3^{n_3}\bot$ and $(n_1,n_2,n_3) = (m_1,m_2,m_3)+\mathbf{1}^e$.*
2) *If $m_e > 0$ then in the bisimulation game of $(u(e,-,j)X\alpha\bot, u'(e,-,j)X\alpha\bot)$, Defender, respectively Attacker, has a strategy to win or at least push the game to (P,Q) such that $P \simeq p_jXC_1^{n_1}C_2^{n_2}C_3^{n_3}\bot$ and $Q \simeq q_jXC_1^{n_1}C_2^{n_2}C_3^{n_3}\bot$ and $(n_1,n_2,n_3) = (m_1,m_2,m_3)-\mathbf{1}^e$.*
3) *Suppose $n \geq m_3$. In the bisimulation game of $(u(3,*,j)X\alpha\bot, u'(3,*,j)X\alpha\bot)$, Defender has a strategy to win or at least push the game to (P,Q) such that $P \simeq p_jXC_1^{n_1}C_2^{n_2}C_3^{n_3}\bot$ and $Q \simeq q_jXC_1^{n_1}C_2^{n_2}C_3^{n_3}\bot$ and $(n_1,n_2,n_3) = (m_1,m_2,m_3) + (n-m_3)\cdot\mathbf{1}^3$.*

Proof. We prove the first statement. The proof for the other two is similar. Let $\beta = C_1^{n_1}C_2^{n_2}C_3^{n_3}$ such that $(n_1,n_2,n_3) = (m_1,m_2,m_3)+\mathbf{1}^e$. In what follows we describe Defender and Attacker's step-by-step optimal strategy in the bisimulation game of $(u(e,+,j)X\alpha\bot, u'(e,+,j)X\alpha\bot)$.

1) By Defender's Forcing, Attacker plays $u(e,+,j)X\alpha\bot \xrightarrow{a} u_1(e,+,j)X\alpha\bot$. Defender responds with
$$u'(e,+,j)X\alpha\bot \overset{\epsilon}{\Longrightarrow} g'(e,+,j)X_1\beta\bot\alpha\bot \xrightarrow{a} u'_1(e,+,j)X\beta\bot\alpha\bot.$$
According to Lemma 23, Attacker's optimal move is to continue the game from
$$(u_1(e,+,j)X\alpha\bot, u'_1(e,+,j)X\beta\bot\alpha\bot).$$
2) It follows from Proposition 22 that $t(e,+)X\alpha\bot \simeq t'(e,+)X\beta\bot\alpha\bot$. If Attacker plays an action labeled c, Defender wins. Attacker's optimal move is to play an action labeled a. Defender then follows suit, and the game reaches the configuration $(u_2(e,+,j)X\alpha\bot, u'_2(e,+,j)X\beta\bot\alpha\bot)$.
3) Attacker's next move is $u'_2(e,+,j)X\beta\bot\alpha\bot \xrightarrow{a} u'_3(e,+,j)X\beta\bot\alpha\bot$. This is optimal by Proposition 22. Defender responds with
$$u_2(e,+,j)X\alpha\bot \overset{\epsilon}{\Longrightarrow} g(e,+,j)X_1\beta\bot\alpha\bot \xrightarrow{a} u_3(e,+,j)X\beta\bot\alpha\bot.$$
By an argument similar to the one given in (i) Attacker would choose
$$(u_3(e,+,j)X\beta\bot\alpha\bot, u'_3(e,+,j)X\beta\bot\alpha\bot)$$
as the next configuration.
4) If Attacker plays an action labeled c, Defender wins by Proposition 22. So Attacker's best bet is to play an action labeled by a. The game reaches the configuration $(p_jX\beta\bot\alpha\bot, q_jX\beta\bot\alpha\bot)$.

The above argument shows that the configuration $(p_jX\beta\bot\alpha\bot, q_jX\beta\bot\alpha\bot)$ is optimal for both Attacker and Defender. We are done. □

C. Control Flow

We now encode the control flow of $\mathcal{M}'$ by the rules of the bisimulation game. We will introduce a number of rules for each instruction in $\mathcal{M}'$.

1) The following rules are introduced in the game $\mathcal{G}$ for an instruction of the form "$i : c_e := c_e + 1$ and then goto j".
$$p_iX \xrightarrow{a} u(e,+,j)X, \quad q_iX \xrightarrow{a} u'(e,+,j)X.$$
2) For each instruction of the form "$i : c_e := *$ and then goto j" the following two rules are added to $\mathcal{R}$.
$$p_iX \xrightarrow{a} u(e,*,j)X, \quad q_iX \xrightarrow{a} u'(e,*,j)X.$$
3) For each instruction of the form "i : goto j or goto k", we have the following.
   - $p_iX \xrightarrow{a} p_i^1X$, $p_iX \xrightarrow{a} q_i^1X$, $p_iX \xrightarrow{a} q_i^2X$; $q_iX \xrightarrow{a} q_i^1X$, $q_iX \xrightarrow{a} q_i^2X$;
   - $p_i^1X \xrightarrow{a} p_jX$, $p_i^1X \xrightarrow{a} p_kX$; $q_i^1X \xrightarrow{a} q_jX$, $q_i^1X \xrightarrow{a} p_kX$; $q_i^2X \xrightarrow{a} p_jX$, $q_i^2X \xrightarrow{a} q_kX$.

   These rules embody precisely the idea of Defender's Forcing [18]. It is Defender who makes the choice.
4) For each instruction of the form

   "i : if $c_e = 0$ then goto j; otherwise $c_e = c_e - 1$ and then goto k"

   we construct a system defined by the following rules.
   - $p_iX \xrightarrow{a} p_i(e,0,j)X$, $p_iX \xrightarrow{c} p_i(e,1,k)X$; $q_iX \xrightarrow{a} q_i(e,0,j)X$, $q_iX \xrightarrow{c} q_i(e,1,k)X$;
   - $p_i(e,0,j)X \xrightarrow{a} v_1(e,0,j)X$, $p_i(e,1,k)X \xrightarrow{a} v_1(e,1,k)X$; $p_i(e,0,j)X \xrightarrow{a} v_2(e,0,j)X$, $p_i(e,1,k)X \xrightarrow{a} v_2(e,1,k)X$; $p_i(e,0,j)X \xrightarrow{a} v_3(e,0,j)X$, $p_i(e,1,k)X \xrightarrow{a} v_3(e,1,k)X$;
   - $q_i(e,0,j)X \xrightarrow{a} v_2(e,0,j)X$, $q_i(e,1,k)X \xrightarrow{a} v_2(e,1,k)X$; $q_i(e,0,j)X \xrightarrow{a} v_3(e,0,j)X$, $q_i(e,1,k)X \xrightarrow{a} v_3(e,1,k)X$;
   - $v_1(e,0,j)X \xrightarrow{a} t(e,1)X$, $v_1(e,0,j)X \xrightarrow{a} p_jX$; $v_2(e,0,j)X \xrightarrow{a} t'(e,1)X$, $v_2(e,0,j)X \xrightarrow{a} p_jX$; $v_3(e,0,j)X \xrightarrow{a} t(e,1)X$, $v_3(e,0,j)X \xrightarrow{a} q_jX$;
   - $v_1(e,1,k)X \xrightarrow{a} t(e,0)X$, $v_1(e,1,k)X \xrightarrow{a} u(e,-,k)X$; $v_2(e,1,k)X \xrightarrow{a} t'(e,0)X$, $v_2(e,1,k)X \xrightarrow{a} u(e,-,k)X$;

	ϵ-Pushing **nPDA**	ϵ-Pushing **PDA**
$\simeq$	Decidable	Σ^1_1-Complete
$\approx$	Π^0_1-Complete	Σ^1_1-Complete

Fig. 8. Decidability and Degree of Undecidability of ϵ-Pushing PDA

$v_3(e,1,k)X \xrightarrow{a} t(e,0)X, \quad v_3(e,1,k)X \xrightarrow{a} u'(e,-,k)X.$

The idea of the above encoding is that Attacker must claim either "$c_e = 0$" or "$c_e > 0$". Defender can check the claim and wins if Attacker lies. If Attacker has not lied, Defender can force Attacker to do what Defender wants.

5) For "$2n$: halt", we add the rules

$$p_{2n}X \xrightarrow{f} p_{2n}\bot, \quad q_{2n}X \xrightarrow{f'} q_{2n}\bot.$$

So Attacker wins if the game ever terminates.

This completes the definition of $\mathcal{G}$.

With the help of Proposition 22 and Lemma 24, it is a routine to prove the next lemma.

Lemma 25. *$\mathcal{M}'$ has an infinite computation if and only if $p_1X\bot \simeq q_1X\bot$.*

Branching bisimilarity on PDA$^{\epsilon+}$ is in Σ^1_1 for the following reason: For any PDA$^{\epsilon+}$ processes P and Q, $P \simeq Q$ if and only if there exists a set of pairs that contains (P,Q) and satisfies the first order arithmetic definable conditions prescribed in Definition 1. Together with the reduction justified by Lemma 25 we derive the main result of the section.

Theorem 26. *The relation $\simeq_{\mathrm{PDA}^{\epsilon+}}$ is Σ^1_1-complete.*

It has been proved in [36] that the branching bisimilarity is undecidable on normed PDA. The reduction defined in the above can be constructed for nPDA too. This is because in nPDA the stack can be reset by popping off all the symbols in the stack using ϵ-popping transitions and creating new stack content using ϵ-pushing transitions, achieving the same effect as the bottom symbol $\bot$ has achieved in PDA$^{\epsilon+}$. The details are omitted.

Theorem 27. *The branching bisimilarity of normed* PDA *is Σ^1_1-complete.*

XI. Conclusion

The results of this paper and the results of Jančar and Srba [18] are summarized in Fig. 8. Stirling's work on the decidability of the strong bisimilarity of nPDA has strong influence on the present work. We have attempted to prove the result of this paper by using tableau system as is done in Stirling's work, see [6] for a report. It turned out that due to the presence of the silent transitions, proof based on a tableau system is not easy to handle. There are a number of difficulties. Firstly in the presence of silent actions the k-bisimilarity, as introduced in the proof of Proposition 7, is very subtle. It is a powerful tool to establish negative results. It is however a little tricky to use it to prove process equivalence. The reason is that transitivity can easily fail if one is not careful about the definition of $\simeq_k$. If transitivity fails, the proof of the backward soundness of tableau rules suffers. Secondly an alternative would be to construct branching bisimulations from a tableau, bypassing the use of k-bisimilarity. This cannot be done by generalizing the similar idea for the strong bisimilarity. Every goal appearing in a tableau is the root of a branching bisimulation. Branching bisimulation of a goal in the conclusion of a tableau rule and that of a goal in the premises have different structure. That makes composition of these bisimulations difficult to define. The way out of the problem is Lemma 9. Using this idea one soon realizes that it would be simpler to work directly with the bisimulation trees. In this paper we have developed decomposition approach to branching bisimulations that in our opinion is better suited to deal with the branching structure in the presence of silent transitions. We hope to say something about the ϵ-popping PDA in another occasion. That would complete the picture initiated here.

In addition to the relationship to the tableau approach, the technique used in this paper can also be seen as a generalization of the bisimulation base method [4]. In Caucal's approach every process has a prime decomposition such that two processes are equivalent if their prime decompositions are equivalent according to a set of axioms. For PDA processes rewriting of processes is insufficient. We have to take into account of the tree structures of these processes. The characteristic trees of the generic goals of nPDA$^{\epsilon+}$ capture the prime structure of equivalent nPDA$^{\epsilon+}$'s. The branching bisimilarity of every pair of nPDA$^{\epsilon+}$ processes can be accounted for in terms of the characteristic trees in a structural way. It would be interesting to see if our method can be applied to other equivalence checking problems to derive new results.

Jančar introduced the notion of first order grammar [14] and provided a quite different proof for the decidability of the strong bisimilarity of nPDA [16]. In the full paper he also outlined an idea of how to extend his proof to take care of silent transitions. The extended PDA model introduced in [6] is similar to the first order grammar of Jančar.

Stirling proved that the language equivalence of DPDA is primitive recursive [28]. Benedikt, Goller, Kiefer and Murawski showed that the strong bisimilarity on nPDA is non-elementary [2]. More recently Jančar observed that the strong bisimilarity of first-order grammar is Ackermann-hard [15], a consequence of which is that the strong bisimilarity proved decidable by Sénizergues in [24] is Ackermann-hard. It is an interesting research direction to look for tighter upper and lower bounds on the branching bisimilarity of nPDA$^{\epsilon+}$.

Acknowledgment

We thank the members of BASICS for their interest. We are grateful to Prof. Jančar for his insightful discussion. The support from NSFC (61472239, ANR 61261130589, 91318301) is gratefully acknowledged.

References

[1] J. Baeten. Branching bisimilarity is an equivalence indeed. *Information Processing Letters* 58:141–147, 1996.

[2] M. Benedikt, S. Moller, S. Kiefer, and A. Murawski. Bisimilarity of Pushdown Automata is Nonelementary. In *LICS'13*, pages 488–498, 2013.

[3] O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on Infinite Structures. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 545–623. North-Holland, 2001.

[4] D. Caucal. Graphes canoniques de graphes algébriques. *Informatique théorique et Applications*, 24:339–352, 1990.

[5] Y. Fu. Checking Equality and Regularity for Normed BPA with Silent Moves. *ICALP'13*, Lecture Notes in Computer Science 7966, 238–249, 2013.

[6] Y. Fu and Q. Yin. Dividing Line between Decidable PDA's and Undecidable Ones. arXive, `https://arxiv.org/abs/1404.7015`, 2014.

[7] S. Ginsburg and S. Greibach. Deterministic Context Free Languages. *Information and Control*, 9:620–648, 1966.

[8] J. Groote and H. Hüttel. Undecidable Equivalences for Basic Process Algebra. *Information and Computation*, 115:354–371, 1994.

[9] D. Harel. Effective Transformations on Infinite Trees, with Applications to High Undecidability, Dominoes, and Fairness. *J. ACM*, 33:224–248, 1986.

[10] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Publishing Company, 1979.

[11] H. Hüttel. Silence is Golden: Branching Bisimilarity is Decidable for Context Free Processes. In *CAV'91*, pages 2–12. Lecture Notes in Computer Science 575, Springer, 1992.

[12] H. Hüttel. Undecidable Equivalences for Basic Parallel Processes. In *Theoretical Aspects of Computer Software*, Lecture Notes in Computer Science 789, pages 454–464, 1994.

[13] H. Hüttel and C. Stirling. Actions Speak Louder than Words: Proving Bisimilarity for Context-Free Processes. In *LICS'91*, pages 376–386, 1991.

[14] P. Jančar. Decidability of DPDA Language Equivalence via First-Order Grammars. In *LICS'12*, page 415–424. IEEE Computer Society, 2012.

[15] P. Jančar. Equivalences of Pushdown Systems are Hard. *Foundations of Software Science and Computation*, pages 1–28, 2014.

[16] P. Jančar. Bisimulation Equivalence of First Order Grammars. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, editors, *ICALP'14*, Lecture Notes in Computer Science 8573, pages 232–243, 2014.

[17] P. Jančar. Bisimulation Equivalence of First Order Grammars. arXiv:1405.7923, 2014.

[18] P. Jančar and J. Srba. Undecidability of Bisimilarity by Defender's Forcing. *Journal of ACM*, 55(1), 2008.

[19] E. Mayr. Undecidability of Weak Bisimulation Equivalence for 1-Counter Processes. In *ICALP'03*, Lecture Notes in Computer Science 2719, page 570–583. Springer, 2003.

[20] R. Mayr. Process Rewrite Systems. *Information and Computation*, 156:264–286, 2000.

[21] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[22] D. Park. Concurrency and Automata on Infinite Sequences. In *TCS'81*, Lecture Notes in Computer Science 104, pages 167–183. Springer, 1981.

[23] G. Sénizergues. The Equivalence Problem for Deterministic Pushdown Automata is Decidable. In *ICALP'97*, Lecture Notes in Computer Science 1256, pages 671–681. Springer-Verlag, 1997.

[24] G. Sénizergues. Decidability of Bisimulation Equivalence for Equational Graphs of Finite Out-Degree. In *FOCS'98*, pages 120–129. IEEE, 1998.

[25] G. Sénizergues. L(a)=L(b)? Decidability Results from Complete Formal Systems. *Theoretical Computer Science*, 251(1-2):1–166, 2001.

[26] G. Sénizergues. L(a)=L(b)? A Simplified Decidability Proof. *Theoretical Computer Science*, 281(1):555–608, 2002.

[27] J. Srba. Undecidability of Weak Bisimilarity for Pushdown Processes. In *CONCUR'02*, Lecture Notes in Computer Science 2421, pages 579–593. Springer-Verlag, 2002.

[28] Stirling. Deciding DPDA Equivalence is Primitive Recursive. In *ICALP'02*, Lecture Notes in Computer Science 2380, pages 821–832. Springer, 2002.

[29] C. Stirling. Decidability of Bisimulation Equivalence for Normed Pushdown Processes. In *CONCUR'96*, Lecture Notes in Computer Science, pages 217–232. Springer-Verlag, 1996.

[30] C. Stirling. Decidability of Bisimulation Equivalence for Normed Pushdown Processes. *Theoretical Computer Science*, 195(2):113–131, 1998.

[31] C. Stirling. The Joy of Bisimulation. In *MFCS'98*, Lecture Notes in Computer Science 1450, pages 142–151. Springer, 1998.

[32] C. Stirling. Decidability of Bisimulation Equivalence for Pushdown Processes. 2000.

[33] C. Stirling. Decidability of DPDA Equivalence. *Theoretical Computer Science*, 255(1-2):1–31, 2001.

[34] R. van Glabbeek and W. Weijland. Branching Time and Abstraction in Bisimulation Semantics. In *Information Processing'89*, pages 613–618. North-Holland, 1989.

[35] R. van Glabbeek and W. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of ACM*, 3:555–600, 1996.

[36] Q. Yin, Y. Fu, C. He, M. Huang, and X. Tao. Branching Bisimilarity Checking for PRS. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, editors, *ICALP'14*, Lecture Notes in Computer Science 8573, pages 363–374, 2014.