Termination Condition for Equivalence Checking Algorithm of PDA

Yuxi Fu[a]

[a]*BASICS, Department of Computer Science, Shanghai Jiao Tong University, China*

Abstract

Equivalence checking of PDA processes is a hard problem. After Senizergues' first proof, several simplifications of the proof methodology have been given. A central issue in the proofs is the termination of the decidability algorithms. In the paper two termination conditions are proposed. They are sufficient for converting a strong bisimulation to a finite tree called generating tree in such a way that the former can be recovered from the latter. A simple proof of the decidability of the strong bisimilarity of PDA is then available by guessing a tree for a pair of input PDA processes and checking if the tree satisfies the desirable properties of generating tree.

Keywords: Pushdown automata, bisimulation, equivalence checking

1. Introduction

After the pioneering work of Baeten, Bergstra and Klop [1], algorithmic study of equivalence checking has been an active line of research [18]. Checking equivalence for pushdown automata, PDA for short, is a very hard problem. Early attention was paid to language equivalence of deterministic pushdown automata (DPDA) [11]. The restriction to DPDA is not just technical; the language equivalence of general PDA is in fact undecidable [12]. The problem was open for a long time before Sénizergues announced that it is decidable [22]. The original proof is very long [24]. Simplified proof appeared soon afterwards [26]. Sénizergues also proved using his approach that the strong bisimilarity of PDA is decidable [23]. The full proof is also quite long [27].

The shift from language equivalence to bisimulation equivalence opened door to process algebraic approach [13, 31]. Stirling has given a decidability proof of normed PDA using tableaux method [29, 30]. Later he applied the same method to the general PDA in an unpublished paper [32]. The tableau approach is also used to provide a simplified proof of the DPDA problem [33, 34]. Jančar addressed the issue using Prover-Refuter game in the setting of first order grammar (FOG), which is a proper generalization of PDA. This has been done for the strong bisimilarity of FOG [16, 17] and the language equivalence of deterministic FOG [14].

A key technical issue in any equivalence checking algorithm for PDA is termination. In Stirling's tableaux approach rules must be backward sound, and a consecutive applications of rules must terminate. Starting from a pair of equivalent processes soundness in Jančar's game theoretical framework means that Refuter should never win the game, and termination says that Prover is able to demonstrate to Refuter in a finite number of steps that the latter will never win the game. We will give a proof of the decidability of the strong bisimilarity of PDA in which the argument for termination is simplified. Our contributions are twofold. Firstly we propose two new termination conditions. One condition deals with sequences of pairs of bisimilar PDA processes of the form $(qZ, H_0), (qZ, H_1), (qZ, H_2), \ldots$ where qZ is unnormed. These bisimilar pairs need special treatment for the reason that the size of H_i may keep increasing while i increases. The second termination condition seeks finite structure in sequences of the form $(A_0\sigma, B_0\sigma), (A_1\sigma, B_1\sigma), (A_2\sigma, B_2\sigma), \ldots$, where the processes share the same suffix σ and the size of the prefixes A_i and B_i may keep increasing while i increases. Our decidability algorithm follows Stirling's approach. But instead of using tableaux, we introduce bisimulation trees. Bisimulation trees have the advantage that they are easy to compose and can be generalized to deal with silent transitions in a straightforward manner. By applying the two termination

Email address: fu-yx@cs.sjtu.edu.cn (Yuxi Fu)

conditions to bisimulation trees we are able to give a simpler proof of the decidability of the strong bisimilarity of PDA. In view of the importance of PDA in equivalence checking and programming theory, especially in automatic verification, any effort to simply the decidability proof is worthwhile. A simpler proof is what is necessary to help expose the problem and its proof technique to a wider audience.

The rest of the paper is structured as follows: Section 2 fixes the syntax and the semantics of PDA. Section 3 introduces the important notion of bisimulation tree. Section 4 looks at algebraic structures of bisimulation trees. Section 5 defines a nondeterministic algorithm that outputs a finite representation of a bisimulation tree when it terminates successfully. Section 6 provides a semidecidable procedure to check the strong bisimilarity of PDA processes by enumerating possible finite representations for a bisimulation tree. Section 7 comments on related works.

2. Pushdown Automata

A *pushdown automaton*, *PDA* for short, is a tuple $\mathcal{P} = (Q, \mathcal{V}, \mathcal{L}, \mathcal{R})$ where

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

The notation $\mathcal{S}^*$ stands for the set of finite strings of elements of $\mathcal{S}$, and ϵ for the empty string. We write u, v, w for *words*, the elements of $\mathcal{L}^*$, and small Greek letters for *stacks*, the elements of $\mathcal{V}^*$. We write $\alpha\delta$ for the concatenation of α and δ. A *$\mathcal{P}$-process*, or *process*, is denoted by $p\alpha$, where $p \in Q$ and $\alpha \in \mathcal{V}^*$. We write the capital letters $A, \ldots, T$ for $\mathcal{P}$-processes. We use $=$ for grammar equality. The size $|p\alpha|$ of $p\alpha$ is the same as the length $|\alpha|$ of α. If $P = p\alpha$ then $P\delta = p\alpha\delta$. A transition rule in $\mathcal{R}$ is of the form $pX \xrightarrow{a} q\alpha$. The semantics of $\mathcal{P}$ is defined by the following rule

$$\frac{pX \xrightarrow{a} q\alpha \in \mathcal{R}}{pX\sigma \xrightarrow{a} q\alpha\sigma}. \tag{1}$$

We write $\xrightarrow{w}$ for $\xrightarrow{a_1} \ldots \xrightarrow{a_k}$ if $w = a_1 \ldots a_k$, and we identify $\xrightarrow{\epsilon}$ to $=$. We say that P' is a *descendant* of P, and that P is a *ancestor* of P', if $P \xrightarrow{w} P'$ for some w. By definition P is a descendant of itself.

An example PDA is defined by $\mathcal{P} = (\{p\}, \{X, Z\}, \{a, b\}, \{pX \xrightarrow{a} p\epsilon, pX \xrightarrow{b} p\epsilon, pZ \xrightarrow{a} pZ, pZ \xrightarrow{b} pZ\})$. It should be clear that pX^nZ and pZ have the same behaviour, where X^n is the n copies of X concatenated one after another.

Given a number, say n, the notation $[n]$ stands for the finite set $\{1, \ldots, n\}$. Let ∂P denote the set $\{i \mid \exists w.P \xrightarrow{w} p_i\epsilon\}$. For $i \in [\mathfrak{q}]$ the *norm of P at i*, denoted by $\|P\|_i$, is the size of a shortest word w such that $P \xrightarrow{w} p_i\epsilon$. If there is no such a word, $\|P\|_i = \omega$. A process P is *normed* if $\partial P \neq \emptyset$; it is *unnormed* otherwise. An important parameter of a PDA is the following.

$$\mathfrak{m} = \max_{p\in Q, X\in\mathcal{V}} \max_{i\in[\mathfrak{q}]} \{\|pX\|_i \mid i \in \partial pX\} + 1. \tag{2}$$

In fewer than $\mathfrak{m}$ steps a sequence of transitions from $pX\alpha$ can expose $p_i\alpha$ for every $i \in \partial pX$. Another parameter used in the decidability algorithm is

$$\mathfrak{r} = \max_{p\in Q, X\in\mathcal{V}} \left\{ |\alpha| \mid pX \xrightarrow{a} q\alpha \in \mathcal{R} \right\}. \tag{3}$$

The parameter $\mathfrak{r}$ is easy to calculate, and the parameter $\mathfrak{m}$ can be calculated from the definition of the PDA by dynamic programming [14].

A transition sequence $pX\alpha \xrightarrow{w} p_i\alpha$ is a *decreasing path* if $|w| = \|pX\|_i$. If $\alpha = Y\beta$ and $p_i\alpha \xrightarrow{w'} p_k\beta$ is a decreasing path, we get a longer path $pX\alpha \xrightarrow{ww'} p_k\beta$ that is decreasing. A consecutive sequence of decreasing paths will be called a *long decreasing path*. Now suppose $\alpha = Z$ and p_iZ is unnormed. We think of every transition $p_iZ \xrightarrow{a} q\delta$ as a decreasing step. If $q\delta \xrightarrow{v} r\gamma$ is a long decreasing path, then both $p_i\alpha \xrightarrow{av} r\gamma$ and $pX\alpha \xrightarrow{wav} r\gamma$ are also long decreasing paths. A long decreasing path from p_iZ typically takes the form $p_iZ \xrightarrow{au} p_{i'}Z' \xrightarrow{a'u'} p_{i''}Z'' \xrightarrow{a''u''} \ldots \xrightarrow{a^ku^k} p_{i^{k+1}}Z^{(k+1)}$.

3. Bisimulation Tree

Two equivalence relations for PDA have been intensively studied. They are language equivalence and bisimulation equivalence [20, 21]. The former is undecidable in general [12]. The latter is the equivalence we study in this paper. We fix some notations first. Suppose $\mathcal{R}, \mathcal{S}$ are binary relations on processes, and $\{\mathcal{R}_i\}_{i\in I}$ is a set of relations. We will use the infix notation $P\mathcal{R}Q$ for $(P,Q) \in \mathcal{R}$. The notation $\mathcal{R}^{-1}$ stands for $\{(Q,P) \mid (P,Q) \in \mathcal{R}\}$, $\mathcal{R};\mathcal{S}$ for the composition $\{(P,Q) \mid \exists M.P\mathcal{R}M \wedge M\mathcal{S}Q\}$, and $\bigcup_{i\in I}\mathcal{R}_i$ for the union. The following definition is standard [29, 30, 32, 33].

Definition 1. *A binary relation $\mathcal{R}$ on processes is a strong simulation if the following statements are valid.*

1. *If $P\mathcal{R}Q \xrightarrow{a} Q'$ then P' exists such that $P \xrightarrow{a} P'\mathcal{R}Q'$.*
2. *If $Q = r\epsilon$ then $P = r\epsilon$.*

The relation $\mathcal{R}$ is a strong bisimulation *if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are strong simulations. The* strong bisimilarity $\sim$ *is the largest strong bisimulation.*

It is easy to see that the strong bisimilarity is both an equivalence and a congruence. Thus $P\sigma \sim Q\sigma$ whenever $P \sim Q$. The equivalence property makes use of the fact that the composition of two strong bisimulations is a strong bisimulation and the union of a family of strong bisimulations is a strong bisimulation [20]. Clause 2 of Definition 1 is unusual. Without it one would have $p\epsilon \sim q\epsilon$ for distinct p, q, and the congruence property would be lost.

If P is unnormed then $P\sigma \sim P$. We shall use the convention that A is normed whenever we write AZ. The next lemma follows immediate from the definition of strong bisimulation.

Lemma 2. *If $P \sim Q$ then $\partial P = \partial Q$ and $\|P\|_i = \|Q\|_i$ for all $i \in \partial P$.*

To help study our equivalence checking problem we will use a constructive version of Definition 1. Let k be a natural number and (P,Q) be a pair of processes. A *k-bisimulation tree $\mathfrak{T}$ for* (P,Q) is a finite tree whose edges are labeled by elements of $\mathcal{L}$ and whose nodes are labeled by pairs of process. For simplicity we shall identify a node with its label. Accordingly we write $(M,N) \xrightarrow{a} (M',N')$ for an edge labeled a from a node labeled (M,N) to a node labeled $(M'N')$. Formally a k-bisimulation tree $\mathfrak{T}$ for (P,Q) is defined as follows:

1. The root of $\mathfrak{T}$ is (P,Q).
2. If $P = Q$, the node (P,Q) is a leaf. It is called an *i-leaf*. And the one-node tree is a 0-bisimulation tree for (P,Q).
3. If $k > 0$ and P, Q are distinct, then every child (P',Q') is the root of a k'-bisimulation tree for (P',Q') for some $k' < k$, and there is at least one child that is the root of a $(k-1)$-bisimulation tree. Moreover the following statements are valid.
   (a) If $(P,Q) \xrightarrow{a} (P',Q')$, then $P \xrightarrow{a} P'$ and $Q \xrightarrow{a} Q'$.
   (b) If $P \xrightarrow{a} P'$ then $Q \xrightarrow{a} Q'$ for some Q' such that $(P,Q) \xrightarrow{a} (P',Q')$. If $Q \xrightarrow{a} Q'$ then $P \xrightarrow{a} P'$ for some P' such that $(P,Q) \xrightarrow{a} (P',Q')$.

A k-bisimulation tree has $k+1$ levels, with the root being at the 0-th level. Its height is k. A $(k+1)$-bisimulation tree $\mathfrak{T}'$ *extends* a k-bisimulation tree $\mathfrak{T}$, notation $\mathfrak{T} \sqsubset \mathfrak{T}'$, if $\mathfrak{T}$ can be obtained by cutting off the leaves of $\mathfrak{T}'$ at the depth $k+1$. If $\mathfrak{T}_i \sqsubset \mathfrak{T}_{i+1}$ for all $i \geq 0$ and $\mathfrak{T}_0$ is the one node tree for (P,Q), then the limit of the chain $\mathfrak{T}_0 \sqsubset \mathfrak{T}_1 \sqsubset \ldots \sqsubset \mathfrak{T}_i \sqsubset \ldots$ is called an ω-bisimulation tree for (P,Q). Intuitively an ω-bisimulation tree is built up level by level. Starting from the root (P,Q), the trivial tree $\mathfrak{T}_0$, one constructs the nodes at the first level. After completing $\mathfrak{T}_1$ one constructs the nodes at the second level. We will also say that the tree is *grown* level by level. The tree-like structure is meant to help one reason about strong bisimulations. The branching version of bisimulation tree was introduced in [9].

Not every path in an ω-bisimulation tree is infinitely long. A finite path ends up in a leaf. By the definition of k-bisimulation, an i-leaf is a pair of identical processes. It is unnecessary to grow such a node since the equality between two identical processes is self evident. We now introduce another kind of leaf. A node is an *r-leaf* if its label coincides with the label of one of its proper ancestors. There is no need to grow an r-leaf because it can be grown in precisely the same way the ancestor has been grown.

In a k-bisimulation tree a node at k'-th level, where $k' < k$, could be either an i-leaf or an r-leaf. If all the nodes at the k-th level are either i-leaves or r-leaves, the tree is called a *complete* k-bisimulation tree, or simply a *complete* bisimulation tree. A *bisimulation tree* is either a complete bisimulation tree or an ω-bisimulation tree. By the next lemma it is not surprising that bisimulation trees play a central role in our decidability proof.

Lemma 3. *$P \sim Q$ if and only if there is a bisimulation tree for (P, Q).*

Proof. For a bisimulation tree $\mathfrak{T}$ we write $\widetilde{\mathfrak{T}}$ for the set of the pairs appearing as labels in $\mathfrak{T}$. Let $\mathcal{I}$ denote the identity relation on processes. By the definition of bisimulation tree $\widetilde{\mathfrak{T}} \cup \mathcal{I}$ is a strong bisimulation. If $P \sim Q$, by the definition of strong bisimulation we can construct $\mathfrak{T}_0 \sqsubset \mathfrak{T}_1 \sqsubset \ldots \sqsubset \mathfrak{T}_i \sqsubset \ldots$ that ends up either with a complete bisimulation tree for (P, Q) or an ω-bisimulation tree for (P, Q). □

Lemma 3 implies that $P \sim Q$ whenever we can grow a bisimulation tree for (P, Q) level by level so that the construction either ends up in a complete bisimulation tree or is non-stopping.

Due to the finite branching property there are only finitely many k-bisimulation trees for (P, Q) for every k. If for some k no k-bisimulation tree for (P, Q) exists, then $P \nsim Q$. This simple observation suggests immediately a semi-decision procedure for $\nsim$.

Lemma 4. *The relation $\nsim$ is semi-decidable.*

The composition of two k-bisimulation trees $\mathfrak{T}$ and $\mathfrak{T}'$, denoted by $\mathfrak{T}; \mathfrak{T}'$, is defined as follows: If $(L, M) \xrightarrow{a} (L', M')$ is an edge in $\mathfrak{T}$ from the i-th level to the $(i{+}1)$-th level and $(M, N) \xrightarrow{a} (M', N')$ is an edge in $\mathfrak{T}'$ from the i-th level to the $(i{+}1)$-th level, then $(L, N) \xrightarrow{a} (L', N')$ is an edge in $\mathfrak{T}; \mathfrak{T}'$ from the i-th level to the $(i{+}1)$-th level. If (L, M) is a node in $\mathfrak{T}$ at the i-th level and (M, M) is an i-leaf in $\mathfrak{T}'$ at the i-th level, then the subtree rooted at (L, M) in $\mathfrak{T}; \mathfrak{T}'$ is the same as the subtree rooted at (L, M) in $\mathfrak{T}$. If (M, M) is both an i-leaf in $\mathfrak{T}$ at the i-th level and an i-leaf in $\mathfrak{T}'$ at the i-th level, it is also an i-leaf in $\mathfrak{T}; \mathfrak{T}'$. It is easy to see that the composition $\mathfrak{T}; \mathfrak{T}'$ is a k-bisimulation tree. Similarly the composition of two ω-bisimulation trees is an ω-bisimulation tree.

4. Periodic Structure of Bisimulation Tree

In Section 3 we have reduced the bisimilarity of a pair of processes to the *existence* of a bisimulation tree for the pair. There is however no way to assert the existence of an ω-bisimulation tree for a pair if ω-bisimulation trees do not exhibit any periodic structure. In this section we give an account of the periodic structure discussed in literature in terms of k-bisimulation trees. We will make use of the periodic structure to design two strategies that constain the growth of bisimulation trees. We will also explain the first termination condition.

Before we proceed, let's explain the idea of saturated bisimulation tree. Suppose $P \sim Q$. We can grow a bisimulation tree for (P, Q) level by level. We say that a k-bisimulation tree for (P, Q) is *saturated* if for every internal node (P', Q') of the k-bisimulation tree whenever $P' \xrightarrow{a} P''$ and $Q' \xrightarrow{a} Q''$ such that $Q'' \sim P''$ then $(P', Q') \xrightarrow{a} (P'', Q'')$ is an out-going edge from $(P'Q')$. A saturated k-bisimulation tree for (P, Q) is the largest k-bisimulation tree for (P, Q). For that reason we may refer to a saturated k-bisimulation tree for (P, Q) as *the* k-bisimulation tree for (P, Q). In the rest of the paper we assume that all k-bisimulation trees for pairs of *bisimilar* processes are saturated.

4.1. Balance Strategy

Suppose $|M| = \mathfrak{m}$ and $|M\sigma| > \mathfrak{m} + 1$. Consider the equality

$$pX\alpha \sim M\sigma. \tag{4}$$

For $i \in \partial pX$ let $pX \xrightarrow{a_0} A_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{k-2}} A_{k-1} \xrightarrow{a_{k-1}} p_i\epsilon$ be a decreasing path. By definition there are $N_1, \ldots, N_{k-1}, M_i$ such that $(pX\alpha, M\sigma) \xrightarrow{a_0} (A_1\alpha, N_1\sigma) \xrightarrow{a_1} \ldots \xrightarrow{a_{k-2}} (A_{k-1}\alpha, N_{k-1}\sigma) \xrightarrow{a_{k-1}} (p_i\alpha, M_i\sigma)$ is a path in the bisimulation tree for $(pX\alpha, M\sigma)$. Clearly $|N_1| > 0, \ldots, |N_{k-1}| > 0$ and $\mathfrak{m}\mathfrak{r} > |M_i| > 0$. We will call the path a *left decreasing path* in the tree. The *left balance strategy* makes use of the minimality of left decreasing paths to reduce imbalance between a pair of processes that descends from (4). The degree of imbalance between A and B is measured by the number $\min_{A',B',\sigma}\{|A'|, |B'| \mid A = A'\sigma, B = B'\sigma\}$. Formally for every $i \in \partial pX$, there is a left decreasing path from $(pX\alpha, M\sigma)$ to $(p_i\alpha, M_i\sigma)$ for some M_i such that

$$p_i\alpha \sim M_i\sigma. \tag{5}$$

1. If (P, Q) is an i-node or an r-node, stop growing.
2. If $|P| \leq \mathfrak{m} + 1$ and $|Q| \leq \mathfrak{m} + 1$, produce all children of (P, Q), and then apply BT to every child.
   - A pair (M, N) is *small* if both $|M| \leq \mathfrak{m} + 1$ and $|N| \leq \mathfrak{m} + 1$.
3. If $P = pX\alpha$ and $Q = M\sigma$ such that $|M| = \mathfrak{m}$, $|M\sigma| > \mathfrak{m} + 1$ and that M does not contain any constant, grow an $(\mathfrak{m}-1)$-bisimulation tree for (P, Q), but suspend the growth of every sink node temporarily.
   - We call a node of the form $(p_i\alpha, M_i\sigma)$ in the $(\mathfrak{m}-1)$-bisimulation tree a *sink node*. Notice that a sink node may appear in the k-bisimulation tree for some $k < \mathfrak{m} - 1$.
   - For each $i \in \partial pX$ fix a sink node $(p_i\alpha, M_i)$ that can be reached from $(pX\alpha, M\sigma)$ by a left decreasing path. These sink nodes define a normed constant U.
   - We call a node at the $(\mathfrak{m}-1)$-th level that is of the form $(L\alpha, N\sigma)$, where $|L| > 0$, a *nonsink node*. To every nonsink node $(L\alpha, N\sigma)$ we create the *u-alias* $(LU\sigma, N\sigma)$. We will explain how to grow the u-alias $(LU\sigma, N\sigma)$ in Section 4.2.

   After growing the $(\mathfrak{m}-1)$-bisimulation tree, apply BT to every sink node, and apply the construction in Fig. 2 to every nonsink node.
4. If $Q = pX\alpha$ and $P = M\sigma$ such that $|M| = \mathfrak{m}$, $|M\sigma| > \mathfrak{m} + 1$ and that M does not contain any constant, carry out the construction symmetric to the one in Case 3.

Figure 1: Construction $BT(P, Q)$ of a Bisimulation Tree for (P, Q) Satisfying $P \sim Q$.

We introduce a mega stack symbol U, which we call a *normed constant* and which is called a simple constant in [29], defined by the following grammar equality:

$$p_iU = \begin{cases} M_i, & \text{if } i \in \partial pX, \\ p_i\epsilon, & \text{if } i \notin \partial pX. \end{cases} \tag{6}$$

For every $i \in [\mathfrak{q}]$ the process p_iU is normed, hence the terminology. The role of U is to transform (4) to $pXU\sigma \sim M\sigma$ by making use of the simple fact that $pX\alpha \sim pXU\sigma$. Algorithmically the pair $(pXU\sigma, M\sigma)$ is easier to deal with than the pair $(pX\alpha, M\sigma)$ because in the latter the imbalance is controlled. The number of the normed constants is finite due to the inequalities

$$0 < |M_i| \leq \mathfrak{m}\mathfrak{r}. \tag{7}$$

Consequently there are only finitely many pairs (pXU, M).

Symmetrically suppose $M\sigma \sim pX\alpha$ such that $|M| = \mathfrak{m}$ and $|M\sigma| > \mathfrak{m} + 1$. There are *right decreasing paths* that allow us to introduce a normed constant U' such that $M\sigma \sim pXU'\sigma$. The *right balance strategy* is then available in this symmetric situation.

We now explain how to grow a bisimulation tree for a bisimilar pair (P, Q) with the help of the normed constants. For that purpose we extend PDA with the normed constants. Every normed constant $U = (M_1, \dots, M_\mathfrak{q})$ satisfies the inequalities (7) such that $M_1, \dots, M_\mathfrak{q}$ are normed and none of $M_1, \dots, M_\mathfrak{q}$ contains any constant. In an extended PDA a stack is a finite string of stack symbols and normed constants. The definition of the strong bisimilarity remains unchanged for the extended model. Suppose $P \sim Q$. The construction of the bisimulation tree for (P, Q) is defined in Fig. 1, where additional definitions and comments are inserted in small fonts. The whole idea of the semi-decidable procedure BT is to grow a bisimulation tree while controlling the imbalance. If a pair (M, N) is small, grow it for one level. The size of its children is bounded. If N is long, grow (M, N) using left balance strategy, and if M is long but N is not, the right balance strategy is applied. The u-aliases created by the left/right balance strategy have bounded size prefixes. In fact if we take the size of a normed constant as one, then

$$|LU| < \mathfrak{m}\mathfrak{r} \text{ and } |N| < \mathfrak{m}\mathfrak{r}. \tag{8}$$

The way to deal with long common suffixes is explained in Section 4.2. So the construction BT gives rise to a partial bisimulation tree in which the growth of the nonsink nodes is suspended. A sink node $(p_i\alpha, M_i\sigma)$ is smaller in the sense that the left process $p_i\alpha$ of the pair is smaller in size than $pX\alpha$. The inductive nature of BT implies that if an infinite path contains an infinite number of sink nodes, then one of the following situations must occur:

1. It contains infinitely many sink nodes $(qY, N_0), (qY, N_1), \ldots, (qY, N_i), \ldots$ for some normed qY. In this case $|N_i| < \mathfrak{m}$ for all $i \geq 0$. Repetition must occur, and the construction ends in either an i-leaf or an r-leaf.
2. It contains infinitely many sink nodes $(rZ, O_0), (rZ, O_1), \ldots, (rZ, O_i), \ldots$ for some unnormed rZ and $\{O_i\}_{i\in\omega}$ is bounded by some number. Repetition must occur, and the construction ends in either an i-leaf or an r-leaf.
3. It contains infinitely many sink nodes $(rZ, O_0), (rZ, O_1), \ldots, (rZ, O_i), \ldots$ for some unnormed rZ such that $|O_i| > \mathfrak{m} + 1$ for all $i \geq 0$ and that no repetition happens. In this case we call (rZ, O_i) an *expansion node*, Z the *principal variable* and $|O_i|$ the size of (rZ, O_i).

We will see in Section 4.3 how to harness the nontermination in the last case.

4.2. Duplication Strategy

We explain now how to grow a u-alias $(LU\sigma, N\sigma)$ generated by BT defined in Fig. 1. The idea is that irrespective of the length of σ a bisimulation tree for $(LU\sigma, N\sigma)$ is a replica of a bisimulation tree for some pair $(LU\nu, N\nu)$ with bounded size suffix ν.

Fix a pair of processes (A, B). Although there could be an infinite number of σ such that

$$A\sigma \sim B\sigma, \tag{9}$$

the bisimulation trees for the pairs $(A\sigma, B\sigma)$ that render (9) true are essentially finite in number. The way to demonstrate this finiteness property is to isolate the growth of some nodes of the form $(p\sigma, N\sigma)$ or $(N\sigma, p\sigma)$ from the rest of the tree whose behaviour then can be completely described without dissolving the suffix σ. We introduce two gadgets to simplify following account. One is an equivalence relation $\mathcal{E}$ on a subset of $[\mathfrak{q}]$. We write $i \in \mathcal{E}$ if i appears in an equivalence class of $\mathcal{E}$, and $i \notin \mathcal{E}$ if otherwise. We write $\langle i \rangle$ for the equivalence class containing i. Initially $\mathcal{E}$ is the reflexive relation on $[\mathfrak{q}]$. The other is a family of equalities

$$p_i V = G_i V, \tag{10}$$

one for each $i \in [\mathfrak{q}]$. Initially $G_i = p_i\epsilon$ for all $i \in [\mathfrak{q}]$. We call V the *recursive constant* defined by the equalities (10). Intuitively V is a mega stack symbol recursively defined such that the process $p_i V$ is the same as the process $G_i V$. If $G_i = p_i\epsilon$ we identify $G_i V$ to $p_i V$. If (10) holds we write $V(i)$ for G_i.

Now suppose the pair $(A\sigma, B\sigma)$ satisfies (9). We can grow level by level a modified bisimulation tree, called the *characteristic tree for* $(A\sigma, B\sigma)$ *over* σ. Suppose we have grown the *k-characteristic tree* $\mathfrak{C}_k$. For each leaf $(p_i\sigma, N\sigma)$ of $\mathfrak{C}_k$ at the k-th level, carry out one of the following.

1. If $i \notin \mathcal{E}$ then replace the label $(p_i\sigma, N\sigma)$ by $(G_i\sigma, N\sigma)$. We may think of the latter as an alias of the former. This is why the bisimulation tree is modified. In this case neither $\mathcal{E}$ nor $\{p_i V = G_i V\}_{i\in[\mathfrak{q}]}$ is changed.
2. If $i \in \mathcal{E}$ and $|N| > 0$, then update the equality family by letting $p_k V = NV$ for all $k \in \langle i \rangle$. Remove $\langle i \rangle$ from $\mathcal{E}$.
3. If $i \in \mathcal{E}$ and $N = p_j\epsilon$ for some $j \notin \mathcal{E}$, then update the equality family by letting $p_k V = G_j V$ for all $k \in \langle i \rangle$. Remove $\langle i \rangle$ from $\mathcal{E}$.
4. If $i \in \mathcal{E}$ and $N = p_j\epsilon$ for some $j \in \mathcal{E}$, then update $\mathcal{E}$ by merging $\langle i \rangle$ with $\langle j \rangle$.

A leaf of $\mathfrak{C}_k$ at the k-th level that is of the form $(N\sigma, p_i\sigma)$ is treated likewise. After all the k-th level nodes of $\mathfrak{C}_k$ have been dealt with, we grow $\mathfrak{C}_k$ into $\mathfrak{C}_{k+1}$. The updates of $\mathcal{E}$ and/or the equality family can only be carried out for a finite number of times. There must exist a number $\mathfrak{h}$ such that all changes of $\mathcal{E}$ have happened in the construction of $\mathfrak{C}_\mathfrak{h}$. After that the characteristic tree can be grown without ever changing $\mathcal{E}$. We say that the characteristic tree $\mathfrak{C}$ is *essentially bounded by* $\mathfrak{h}$. We need to modify the equality family for the final time. For each $i \in \mathcal{E}$ and each $k \in \langle i \rangle$ replace the equality $p_k V = p_k V$ by $p_k V = p_{\min\langle i \rangle} V$. By construction for each i such that $p_i V = G_i V$ and $|G_i| \neq 0$ there is a node $(p_i\sigma, G_i\sigma)$ in $\mathfrak{C}_\mathfrak{h}$. We call such a node a *recursive node*. If $(A\sigma, B\sigma)$ is some u-alias $(LU\sigma, N\sigma)$, then both $|A|$ and $|B|$ are bounded by $\mathfrak{m}\mathfrak{r}$, confer (8). It follows that there exists some constant $\mathfrak{h}_0$ such that

$$|G_i| < \mathfrak{m}\mathfrak{r} + \mathfrak{h}_0\mathfrak{r} \tag{11}$$

for all $i \in [\mathfrak{q}]$.

For the u-alias $(LU\sigma, N\sigma)$ of a nonsink node generated by BT, grow the characteristic tree for $(LU\sigma, N\sigma)$ for $\mathfrak{h}_0$-levels, which produces a recursive constant V.

Call every node $(C\sigma, D\sigma)$ at the $\mathfrak{h}_0$-th level that is not a recursive node a *balance node*. In a balance node $(C\sigma, D\sigma)$ both $|C| < \mathfrak{m}\mathfrak{r} + \mathfrak{h}_0\mathfrak{r}$ and $|D| < \mathfrak{m}\mathfrak{r} + \mathfrak{h}_0\mathfrak{r}$. Thus there are only finitely many balance nodes and of course finitely many v-aliases.

1. Apply BT to every recursive node.
2. For every balance node $(C\sigma, D\sigma)$ create the *v-alias* (CV, DV), and then apply BT to grow the v-alias.

Figure 2: Growth of U-Alias.

We will take every equality in $\{p_iV = G_iV\}_{i\in[\mathfrak{q}]}$ as grammar equality. We say that V is undefined at j if $p_jV = p_iV$ for some $i \leq j$ such that $G_i = p_i\epsilon$.

To demonstrate the power of recursive constants we extend PDA model further with recursive constants. In the new extended model every recursive constant V' is defined by equalities in the form of (10). We require that for every $i \in [\mathfrak{q}]$ either G_i contains no constant or it contains a normed constant that appears as the last symbol in G_i. A stack may contain at most one recursive constant. A recursive constant may only occur as the last symbol in a stack. For PDA's extended with normed constants and recursive constants the definition of strong bisimulations need be strengthened by including the following clause:

3) If $Q = p_iV'$ and V' is undefined at i, then $P = p_iV'$.

By definition if $p_iV' = p_jV'$ and V' is undefined at j, then $p_iV' \sim p_jV'$.

We are now in a position to state some properties for the recursive constant V constructed from the bisimilar processes in (9) and defined by the equalities in (10).

Lemma 5. *$AV \sim BV$.*

Proof. If we substitute V for σ in the characteristic tree for $(A\sigma, B\sigma)$ over σ, we get a bisimulation tree for (AV, BV), bearing in mind the grammar equality in (10). We are done by applying Lemma 3. □

We can also carry out the reverse transformation. Let $\mathfrak{T}$ be a bisimulation tree for (AV, BV). Let $\mathfrak{T}\{\sigma/V\}$ denote the tree obtained from $\mathfrak{T}$ by substituting σ for V. The next lemma implies that $\mathfrak{T}\{\sigma/V\}$ is the characteristic tree for $(A\sigma, B\sigma)$ over σ.

Lemma 6. *If $p_i\delta \sim G_i\delta$ for all $i \in [\mathfrak{q}]$ then for all C, D such that $|C| > 0$ and $|D| > 0$, $CV \sim DV$ implies $C\delta \sim D\delta$.*

Proof. The composition $\sim; \{(C\delta, D\delta) \mid CV \sim DV\}; \sim$ is easily seen to be a strong bisimulation. □

In effect Lemma 6 allows us to decompose a bisimulation tree for $(A\sigma, B\sigma)$ into the bisimulation tree for (AV, BV) and bisimulation trees for the recursive nodes. The bisimulation tree for (AV, BV) acts as a blueprint. This would not be very interesting if there are infinitely many blueprints.

Lemma 7. *For fixed A, B there are only finitely many recursive constants generated by the bisimulation trees for the family $\{(A\sigma, B\sigma) \mid A\sigma \sim B\sigma\}_{\sigma\in\mathcal{V}^*}$.*

Proof. The initial V is defined by the family $\{p_iV = p_i\epsilon V\}_{i\in[\mathfrak{q}]}$, which does not depend on any σ. In any step during the construction of V, if $AV \nsim BV$ then there is a least $k < \mathfrak{h}$ such that there is no k-bisimulation tree for (AV, BV). The crucial point is that $AV \nsim BV$ does not depend on any σ. By the finite branching property there are only finitely many k-bisimulation trees for $A\sigma \sim B\sigma$ with fixed A, B and varying σ. Consequently there are only finitely many ways to update V. □

Lemma 7 implies that there is a single constant, still denoted by $\mathfrak{h}_0$, such that the characteristic trees for all the u-aliases are essentially bounded by $\mathfrak{h}_0$. So we impose the condition (11) for all the recursive constants. We are now in a position to complete the definition of BT. The growth of u-aliases is defined in Fig. 2.

The growth of a v-alias (CV, DV) may expose a normed constant in V. Suppose $(p'X'\alpha'UV, D'\beta'UV)$ is a descendant of (CV, DV) such that $|D'| = \mathfrak{m}$ and a normed constant U appears in both the left and the right processes. Notice that neither α' nor $D'\beta'$ may contain any normed constants by our restriction on recursive constants. If the left balance strategy is applicable to $(p'X'\alpha'UV, D''\beta'UV)$, it introduces sink nodes of the form $(p\alpha'UV, M'\beta'UV)$ and nonsink nodes of the form $(L'\alpha'UV, N'\beta'UV)$. A new normed constant U' is introduced, which does not contain any occurrence of U. The u-alias of $(L'\alpha'UV, N'\beta'UV)$ is $(L'U'\beta'UV, N'\beta'UV)$. The common suffix of $L'U'\beta'UV$ and $N'\beta'UV$ is $\beta'UV$. When growing the characteristic tree for $(L'U'\beta'UV, N'\beta'UV)$ over $\beta'UV$, a new recursive constant V', defined by say $\{p_iV' = G'_iV'\}_{i\in[\mathfrak{q}]}$, is introduced. It is obvious that V' may contain U' as the last symbol in some G'_j, but it definitely does not contain any occurrence of U. We conclude that the construction defined in Fig. 2 only introduces normed constants and recursive constants that meet our constraints. This fine property depends crucially on the requirement $|M\sigma| > \mathfrak{m} + 1$ in the clause 3 and the clause 4 of the definition of BT. The requirement guarantees that M contains no constant.

The construction of the characteristic tree over σ for $(A\sigma, B\sigma)$ satisfying (9) can be generalized. Instead of starting from the trivial recursive constant V_0 defined by $V_0(i) = p_i\epsilon$ for all $i \in [\mathfrak{q}]$, we may begin with a nontrivial recursive constant V'_0 such that $p_i\sigma \sim V'_0(i)\sigma$ for all $i \in [\mathfrak{q}]$. The way to construct the characteristic tree remains the same. The result of the construction is the characteristic tree for $(A\sigma, B\sigma)$ over σ *extending* V'_0. Let V' be the recursive constant thus defined. We write $V'_0 \preceq V'$ to indicate the fact that V' is defined from V'_0.

4.3. Termination Condition for Balance Strategy

We now address the issue pointed out at the end of Section 4.1. The construction BT may not terminate because there exists an infinite path in which there are an infinite number of expansion nodes. Observe that the length of a decreasing path $(pX, q\beta) \xrightarrow{w} (p'X', q'\beta')$ is bounded by $\mathfrak{m}\mathfrak{r}$. In an (infinite) long decreasing path from $(pX, q\beta)$ there must be some expansion node (q_0Z_0, H_0) whose size is minimum. Now consider the infinite long decreasing path

$$(q_0Z_0, H_0) \xrightarrow{w_0} (q_1Z_1, H_1) \xrightarrow{w_1} \ldots \xrightarrow{w_{j-1}} (q_jZ_j, H_j) \xrightarrow{w_j} (q_{j+1}Z_{j+1}, H_{j+1}) \xrightarrow{w_{j+1}} \ldots. \tag{12}$$

The decreasing path $(q_jZ_j, H_j) \xrightarrow{w_j} (q_{j+1}Z_{j+1}, H_{j+1})$ is of length no more than $\mathfrak{m}\mathfrak{r}$. So (12) can be written as

$$(q_0Z_0, G_0\sigma_0) \xrightarrow{w_0} (q_1Z_1, G_1\sigma_0) \xrightarrow{w_1} \ldots \xrightarrow{w_{j-1}} (q_jZ_j, G_j\sigma_0) \xrightarrow{w_j} (q_{j+1}Z_{j+1}, G_{j+1}\sigma_0) \xrightarrow{w_{j+1}} \ldots \tag{13}$$

such that $|G_0| \leq \mathfrak{m}\mathfrak{r}$ and $|G_0| \leq |G_j|$ for all j. For some $k_1 \leq \mathfrak{n}^{\mathfrak{m}\mathfrak{r}}$ we must have $|G_0| < |G_{k_1}|$ for otherwise there would be a repetition. Clearly the length of the long decreasing path from $(q_0Z_0, G_0\sigma_0)$ to $(q_{k_1}Z_{k_1}, G_{k_1}\sigma_0)$ is bounded by $\mathfrak{m}\mathfrak{r}\mathfrak{n}^{\mathfrak{m}\mathfrak{r}}$, hence $|G_{k_1}| < \mathfrak{m}\mathfrak{r}\mathfrak{n}^{\mathfrak{m}\mathfrak{r}}\mathfrak{r}$. Inductively define size function $s(x)$ and time function $t(x)$ as follows:

$$\begin{aligned}
t(0) &= 0, \\
s(0) &= \mathfrak{m}\mathfrak{r}, \\
t(i{+}1) &= t(i) + \mathfrak{m}\mathfrak{r}(\mathfrak{n}{+}1)^{s(i)}, \\
s(i{+}1) &= s(i) + \mathfrak{m}\mathfrak{r}^2(\mathfrak{n}{+}1)^{s(i)}.
\end{aligned}$$

Assume that there are some $k_1 < t(1), \ldots, k_n < t(n)$ such that $0 = k_0 < k_1 < k_2 < \ldots < k_n$, $|G_{k_i}| < s(i)$ for all $i \in [n]$, and $|G_0| = |G_{k_0}| < |G_{k_1}| < \ldots < |G_{k_n}|$. Then by the definition of $t(n+1)$ there must exist some k_{n+1} such that $t(n) < k_{n+1} < t(n+1)$ and $|G_{k_n}| < |G_{k_{n+1}}|$. Let $\bar{t}$ be the computable function defined as follows:

$$\begin{aligned}
\bar{t}(0) &= 0, \\
\bar{t}(j{+}1) &= \bar{t}(j) + \mathfrak{m}s(\bar{t}(j)).
\end{aligned}$$

Define the sequence

$$(r_0Y_0, K_0\sigma_0), (r_1Y_1, K_0\sigma_1), \ldots \tag{14}$$

by letting

$$(r_jY_j, K_j\sigma_0) = (q_{\bar{t}(j)}Z_{\bar{t}(j)}, G_{\bar{t}(j)}\sigma_0)$$

for all $j \geq 0$. The function $\bar{t}$ is so defined to ensure that for all j the length of the long decreasing path from the node $(r_jY_j, K_j\sigma_0)$ to the node $(r_{j+1}Y_{j+1}, K_{j+1}\sigma_0)$ is longer than the length of any long decreasing path from $K_j\sigma_0$ that exposes σ_0. Consider the subsequence

$$(r_0Y_0, K_0\sigma_0), (r_1Y_1, K_1\sigma_0), \ldots, (r_{\mathfrak{q}\mathfrak{n}2^{\mathfrak{q}}}Y_{\mathfrak{q}\mathfrak{n}2^{\mathfrak{q}}}, K_{\mathfrak{q}\mathfrak{n}2^{\mathfrak{q}}}\sigma_0) \tag{15}$$

that contains the first $\mathfrak{q}\mathfrak{n}2^{\mathfrak{q}} + 1$ nodes in (14). The multiplier $\mathfrak{q}\mathfrak{n}$ guarantees that in (15) there must be two expansion nodes $(r_jY_j, K_j\sigma_0)$ and $(r_{j'}Y_{j'}, K_{j'}\sigma_0)$, where $j < j'$, such that $r_jY_j = r_{j'}Y_{j'} = qZ$ for some qZ, and the multiplier $2^{\mathfrak{q}}$ ensures that $K_j = G_a$ and $K_{j'} = G_b$ for some a and b such that

$$\partial G_a = \partial G_b \tag{16}$$

and

$$|G_a| < s(\bar{t}(\mathfrak{q}\mathfrak{n}2^{\mathfrak{q}})) \text{ and } |G_b| < s(\bar{t}(\mathfrak{q}\mathfrak{n}2^{\mathfrak{q}})). \tag{17}$$

The length of the long decreasing path from (qZ, G_a) to (qZ, G_b) is bounded by

$$\bar{t}(\mathfrak{q}\mathfrak{n}2^{\mathfrak{q}}). \tag{18}$$

Now grow the expansion node $(qZ, G_a\sigma_0)$ until the long decreasing path reaches $(qZ, G_b\sigma_0)$. By the property of (15) for every $h \in \partial G_a$ a node of the form $(C_h, p_h\sigma_0)$ satisfying

$$C_h \sim p_h\sigma_0 \tag{19}$$

is generated at a level above the level that $(qZ, G_b\sigma_0)$ appears. The nice thing about (19) is that

$$|C_h| \leq s(\bar{t}(\mathfrak{q}\mathfrak{n}2^{\mathfrak{q}}))\mathfrak{m}\mathfrak{r}. \tag{20}$$

Now introduce a new constant W, called *unnormed constant*, defined by the following grammar equality:

$$p_iW \quad = \quad \begin{cases} C_i, & \text{if } i \in \partial G_a, \\ p_i\epsilon, & \text{if } i \notin \partial G_a. \end{cases} \tag{21}$$

It follows from (21) and (20) that $|p_iW| \leq s(\bar{t}(\mathfrak{q}\mathfrak{n}2^{\mathfrak{q}}))\mathfrak{m}\mathfrak{r}$ for all $i \in [\mathfrak{q}]$, and that both $G_a\sigma_0 \sim G_aW$ and $G_b\sigma_0 \sim G_bW$. So we may let (G_aW, G_bW) be an alias of the expansion node $(qZ, G_b\sigma_0)$, called the *W-alias* of $(qZ, G_b\sigma_0)$, and continue to grow the W-alias. There cannot be an infinite path that contains an infinite number of W-aliases because there are only finitely many W-aliases. We have therefore removed one cause of nontermination from BT. The constant W is unnormed because C_i is unnormed for every $i \in \partial G_a$. So we may ignore the trailing α in $W\alpha$. A long decreasing path from pW is essentially a long decreasing path from some $p'Z'$. So it does not give rise to any new nondeterminism.

The introduction of the recursive constants opens up new possibility for nontermination. Infinite long decreasing sequence of the following form is possible, where $\{|H_j|\}_{j\in\omega}$ is unbounded.

$$(q_0V, H_0) \xrightarrow{w_0} (q_1V, H_1) \xrightarrow{w_1} \ldots \xrightarrow{w_{j-1}} (q_jV, H_j) \xrightarrow{w_j} (q_{j+1}V, H_{j+1}) \xrightarrow{w_{j+1}} \ldots. \tag{22}$$

It ought to be clear that the sequence (22) can be treated in completely the same way as the sequence (12) by exploring the fact that $|V(i)| < \mathfrak{m}\mathfrak{r} + \mathfrak{h}_0\mathfrak{r}$ for all $i \in [\mathfrak{q}]$. A W-alias introduced in this case may contain V as the last symbol in p_iW, where $i \in [\mathfrak{q}]$. Since the number of recursive constants is finite, the number of W-aliases must remain finite.

From now on we assume that an extended PDA admits normed, recursive, and unnormed constants.

4.4. Termination Condition for Duplication Strategy

Let's take another look at the semi-decidable procedure BT defined in Fig. 1 and Fig. 2. Section 4.2 provides a method to grow a nonsink node. Section 4.3 introduces a technique to terminate a long decreasing sequence. Nontermination is however not completely eliminated. Suppose $(L_0U_0\sigma_0, M_0\sigma_0)$ is a u-alias. After growing the characteristic tree for the u-alias over σ_0, we get a recursive node of the form $(p'\sigma_0, G'\sigma_0)$ say. We apply the left balance strategy to grow $(p'\sigma_0, G'\sigma_0)$, assuming that $|G'\sigma_0| > \mathfrak{m} + 1$. Some new u-alias $(L_1U_1\sigma_1, M_1\sigma_1)$ is then generated. According to the construction either σ_0 is a suffix of σ_1 or σ_1 is a suffix of σ_0, depending on whether $|G'| \geq \mathfrak{m}$ or not. In this fashion a sequence of u-aliases

$$(L_0U_0\sigma_0, M_0\sigma_0), (L_1U_1\sigma_1, M_1\sigma_1), \ldots, (L_iU_i\sigma_i, M_i\sigma_i), \ldots . \tag{23}$$

might be produced. Generally we need to consider the situations where the i-th pair in (23) could be $(M_i\sigma_i, L_iU_i\sigma_i)$. The following argument however would be the same in the general case. So we shall only consider the sequence (23). Two exclusive situations may arise. One is that for every $i \geq 0$ there is some $j > i$ such that $|\sigma_j| \leq |\sigma_i|$. In this case a repetition must occur. The other is that there is some i_0 such that $|\sigma_{i_0}| < |\sigma_j|$ for all $j > i_0$. In the latter case we must have a consecutive sequence of u-aliases that looks like the following.

$$(L_{i_0}U_{i_0}\sigma_{i_0}, M_{i_0}\sigma_{i_0}), (L_{i_0+1}U_{i_0+1}\delta_{i_0+1}\sigma_{i_0}, M_{i_0+1}\delta_{i_0+1}\sigma_{i_0}), \ldots, (L_{i_0+j}U_{i_0+j}\delta_{i_0+j}\sigma_{i_0}, M_{i_0+j}\delta_{i_0+j}\sigma_{i_0}), \ldots, \tag{24}$$

where $|\delta_{i_0+j}| > 0$ for all $j > 0$. We will call (24) a *positive u-alias sequence*. An alternative way to see the sequence is that the suffix σ_{i_0} is fixed whereas the prefixes are increasing in size. If we grow the characteristic tree for every pair in (24) *over* σ_{i_0}, we get different recursive constants. Let V_{i_0} be the recursive constant defined by the $\mathfrak{h}_0$-characteristic tree for $(L_{i_0}U_{i_0}\sigma_{i_0}, M_{i_0}\sigma_{i_0})$ over σ_{i_0}. We now explain the second termination condition. There are two cases.

1. $L_{i_0+\mathfrak{h}_0}U_{i_0+\mathfrak{h}_0}\delta_{i_0+\mathfrak{h}_0}V_{i_0} \sim M_{i_0+\mathfrak{h}_0}\delta_{i_0+\mathfrak{h}_0}V_{i_0}$. In this case $(L_{i_0+\mathfrak{h}_0}U_{i_0+\mathfrak{h}_0}\delta_{i_0+\mathfrak{h}_0}\sigma_{i_0}, M_{i_0+\mathfrak{h}_0}\delta_{i_0+\mathfrak{h}_0}\sigma_{i_0})$ is a *deep balance node*. We can rename the node to $(L_{i_0+\mathfrak{h}_0}U_{i_0+\mathfrak{h}_0}\delta_{i_0+\mathfrak{h}_0}V_{i_0}, M_{i_0+\mathfrak{h}_0}\delta_{i_0+\mathfrak{h}_0}V_{i_0})$, called the *d-alias* of the deep balance node.
2. $L_{i_0+\mathfrak{h}_0}U_{i_0+\mathfrak{h}_0}\delta_{i_0+\mathfrak{h}_0}V_{i_0} \nsim M_{i_0+\mathfrak{h}_0}\delta_{i_0+\mathfrak{h}_0}V_{i_0}$. Construct the characteristic tree for $(L_{i_0+\mathfrak{h}_0}U_{i_0+\mathfrak{h}_0}\delta_{i_0+\mathfrak{h}_0}\sigma_{i_0}, M_{i_0+\mathfrak{h}_0}\delta_{i_0+\mathfrak{h}_0}\sigma_{i_0})$ over σ_{i_0} extending V_{i_0}, generating a recursive constant V_0. Now $|V_0(i')| > 0$ but $|V_{i_0}(i')| = 0$ for some $i' \in [\mathfrak{q}]$. Recall that $|V_{i_0}(i')| < \mathfrak{m}\mathfrak{r} + \mathfrak{h}_0\mathfrak{r}$ for all $i' \in [\mathfrak{q}]$, hence $|\delta_{i_0+1}| < \mathfrak{m}\mathfrak{r} + \mathfrak{h}_0\mathfrak{r}$. It follows that $|\delta_{i_0+\mathfrak{h}_0}| < \mathfrak{h}_0(\mathfrak{m}\mathfrak{r} + \mathfrak{h}_0\mathfrak{r})$. In fact $|L_{i_0+\mathfrak{h}_0}U_{i_0+\mathfrak{h}_0}\delta_{i_0+\mathfrak{h}_0}| \leq \mathfrak{h}_0(\mathfrak{m}\mathfrak{r} + \mathfrak{h}_0\mathfrak{r})$ and $|M_{i_0+\mathfrak{h}_0}\delta_{i_0+\mathfrak{h}_0}| \leq \mathfrak{h}_0(\mathfrak{m}\mathfrak{r} + \mathfrak{h}_0\mathfrak{r})$. Let $\mathfrak{h}_1$ be a constant such that the characteristic trees for $(P'\sigma', Q'\sigma')$ over σ' satisfying $|P'| \leq \mathfrak{h}_0(\mathfrak{m}\mathfrak{r} + \mathfrak{h}_0\mathfrak{r})$ and $|Q'| \leq \mathfrak{h}_0(\mathfrak{m}\mathfrak{r} + \mathfrak{h}_0\mathfrak{r})$ are essentially bounded by $\mathfrak{h}_1$. If $L_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}U_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}V_0 \sim M_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}V_0$, then the deep balance node is $(L_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}U_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\sigma_{i_0}, M_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\sigma_{i_0})$ and $(L_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}U_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}V_0, M_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}V_0)$ is the d-alias of the deep balance node. If $L_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}U_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}V_0 \nsim M_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}V_0$, then construct the characteristic tree for $(L_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}U_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\sigma_{i_0}, M_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1}\sigma_{i_0})$ over σ_{i_0} extending V_0, generating a recursive constant V_1. We then continue inductively. The construction must end since there are only $\mathfrak{q}$ entries to a recursive constant.

We conclude that there are constants $\mathfrak{h}_0 < \ldots < \mathfrak{h}_\mathfrak{q}$, depending only on the definition of the PDA, such that for each positive u-alias sequence of the form (24) there is some $h < \mathfrak{q}$ that renders valid the following statement: There are recursive constants $V_0, V_1, \ldots, V_h$ satisfying $V_{i_0} \preceq V_0 \preceq V_1 \preceq \ldots \preceq V_h$ such that for each $j \leq h$ the recursive constant V_j is defined by the $\mathfrak{h}_j$-characteristic tree over σ_{i_0} for

$$(L_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_j}U_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_j}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_j}\sigma_{i_0}, M_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_j}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_j}\sigma_{i_0});$$

moreover $L_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}U_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}V_h \sim M_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}V_h$. The node

$$(L_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}U_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}\sigma_{i_0}, M_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}\sigma_{i_0})$$

is the deep balance node and its d-alias is the node

$$(L_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}U_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}V_h, M_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}\delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_{h+1}}V_h).$$

Without loss of generality we may assume that $|V_h(i)| \leq \mathfrak{h}_h$ for all $i \in [\mathfrak{q}]$.

5. Generating Tree

We are now in a position to turn the *semantic* construction BT into a nondeterministic *algorithm* GT by incorporating all the facts revealed so far in Section 4. We remark that an input to the construction BT is a pair such that $P \sim Q$, whereas an input to the algorithm GT is any pair of extended processes. The nondeterministic algorithm GT is defined in Fig. 3. Again some remarks are inserted in the algorithm in small font. To check the property stated in Step 3(b)iiA, the algorithm has to call a subroutine that upon receiving a number $\mathfrak{h}$ generates an $\mathfrak{h}$-characteristic tree for some node in the output tree of the algorithm. The subroutine will be defined in Section 6. The algorithm GT makes many guesses. However termination is guaranteed.

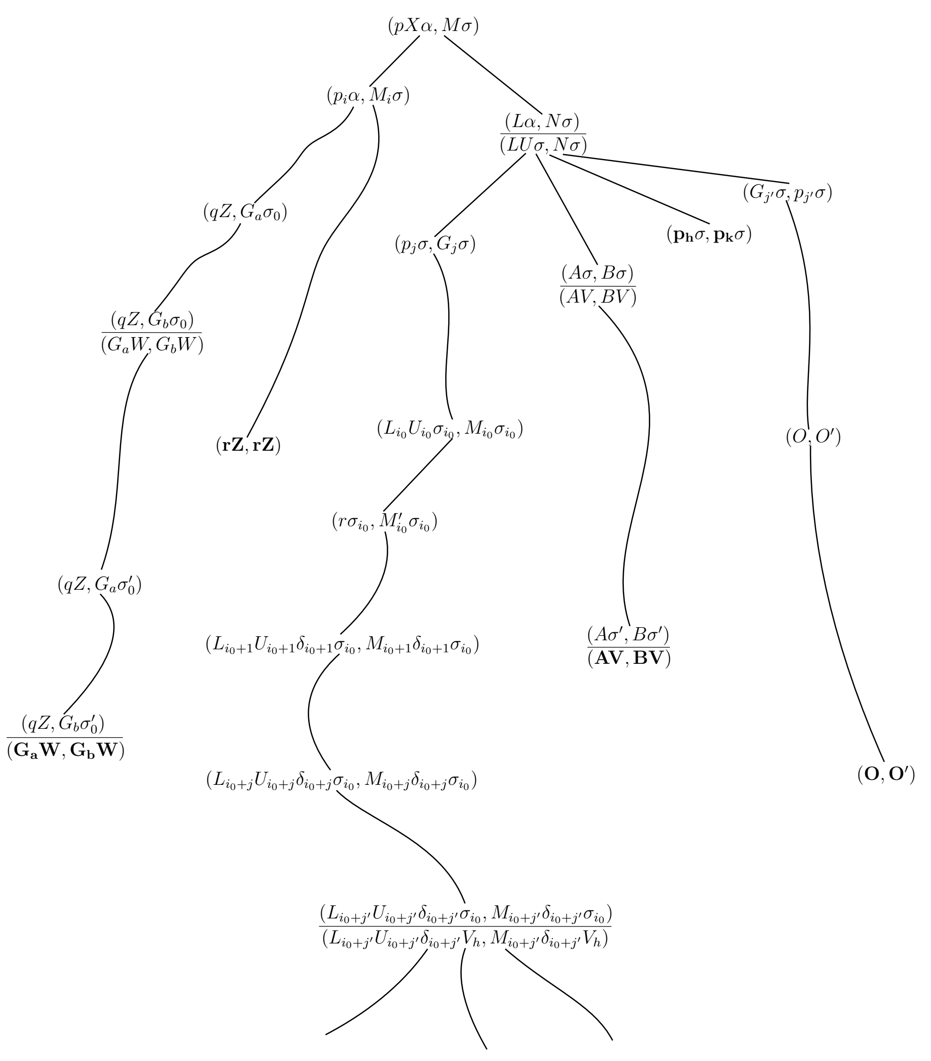
```
INPUT: A pair $(P,Q)$ of processes, and numbers $\mathfrak{h}_0, \mathfrak{h}_1, \ldots, \mathfrak{h}_\mathfrak{q}$ such that $\mathfrak{h}_0 < \mathfrak{h}_1 < \ldots < \mathfrak{h}_\mathfrak{q}$.
ALGORITHM $GT$:
  1. If $(P,Q)$ is an i-node or an r-node, stop.
  2. If $|P| \leq \mathfrak{m}+1$ and $|Q| \leq \mathfrak{m}+1$, guess a finite set of transitions $\left\{(P,Q) \xrightarrow{b_i} (P_i,Q_i)\right\}_{i \in I}$, and verify that the set of edges $\left\{P \xrightarrow{b_i} P_i, Q \xrightarrow{b_i} Q_i\right\}_{i \in I}$ form a 1-bisimulation tree for $(P,Q)$. If it does form a 1-bisimulation tree, apply $GT$ to every $(P_i,Q_i)$ that is neither an i-leaf nor an r-leaf; otherwise abort.
     In the recursive invocations of $GT$ the parameters $\mathfrak{h}_0, \mathfrak{h}_1, \ldots, \mathfrak{h}_\mathfrak{q}$ remain the same.
  3. If $P = pX\alpha$ and $Q = M\sigma$ such that $|M| = \mathfrak{m}$, $|M\sigma| > \mathfrak{m}+1$, guess an $(\mathfrak{m}-1)$-bisimulation tree for $(P,Q)$ but suspend the growth of every sink node so that a normed constant $U$ is defined. Verify that the internal nodes of the guessed tree satisfy the strong bisimulation property. If the verification is unsuccessful, abort; otherwise do the following.
     (a) For every sink node do the following:
         i. If the sink node is not an expansion node, apply $GT$ to the sink node.
         ii. If the sink node is an expansion node $(qZ,N)$, then abort if there is a long left decreasing path from some $(qZ,N')$ to $(qZ,N)$ that is longer than $\bar{t}(\mathfrak{q}\mathfrak{n}2^\mathfrak{q})$, otherwise do (3(a)iiA) or (3(a)iiB) nondeterministically.
             A. Guess a finite set of transitions $\left\{(qZ,N) \xrightarrow{b_i} (P_i,Q_i)\right\}_{i \in J}$, and verify that the set of edges $\left\{qZ \xrightarrow{b_i} P_i, N \xrightarrow{b_i} Q_i\right\}_{i \in J}$ form a 1-bisimulation tree for $(qZ,N)$. If it does form a 1-bisimulation tree, apply $GT$ to every $(P_i,Q_i)$ that is neither an i-leaf nor an r-leaf; otherwise aborts.
             B. Choose a long left decreasing path from some $(qZ,N')$ to $(qZ,N)$, abort if such a path does not exist or if the statement (†) cannot be validated:
                  (†) $N' = G_a\sigma_0$ and $N = G_b\sigma_0$ for some $G_a, G_b$ satisfying $\partial G_a = \partial G_b$.
                For each $h \in \partial G_a$ choose a long right decreasing path from $(qZ, G_a\sigma_0)$ to some $(C_h, p_h\sigma_0)$. Define an unnormed constant $W$ as in (21); introduce the W-alias $(G_aW, G_bW)$ of $(qZ, W_b\sigma_0)$; and apply $GT$ to the W-alias.
     (b) For the u-alias $(LU\sigma, N\sigma)$ of every nonsink node, abort if there is a positive u-alias sequence that ends in $(LU\sigma, N\sigma)$ and is longer than $\mathfrak{h}_0 + \mathfrak{h}_1 + \ldots + \mathfrak{h}_\mathfrak{q}$, otherwise nondeterministically do (3(b)i) or (3(b)ii).
         i. Guess a characteristic tree of height $\mathfrak{h}_0$ for $(LU\sigma, N\sigma)$. If any of the internal nodes of the guessed tree fails the strong bisimulation property, abort; otherwise do the following.
             A. Apply $GT$ to every recursive node.
             B. Apply $GT$ to the v-alias of every balance node.
         ii. If $(LU\sigma, N\sigma)$ is $(L_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_h} U_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_h} \delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_h} \sigma_{i_0}, M_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_h} \delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_h} \sigma_{i_0})$ for some $h \in [\mathfrak{q}]$, guess a recursive constant $V_h$ such that $|V_h(i)| < \mathfrak{h}_h$ for all $i \in [\mathfrak{q}]$, and then apply $GT$ to the deep balance node $(L_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_h} U_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_h} \delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_h} V_h, M_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_h} \delta_{i_0+\mathfrak{h}_0+\mathfrak{h}_1+\ldots+\mathfrak{h}_h} V_h)$.
             In the decidability proof the guessed recursive constant $V_h$ must be validated. Specifically it must be shown that
             A. for each $i$ satisfying $|V_h(i)| > 0$ there is some $j < h$ such that $(p_i\sigma_{i_0}, V_h(i)\sigma_{i_0})$ or $(V_h(i)\sigma_{i_0}, p_i\sigma_{i_0})$ is admitted in the $\mathfrak{h}_j$-characteristic tree constructed from the output of $GT$.
             We will explain how the $\mathfrak{h}_j$-characteristic tree is constructed in the proof of Theorem 9.
  4. If $Q = pX\alpha$, $|Q| \leq \mathfrak{m}+1$ and $P = M\sigma$ such that $|M| = \mathfrak{m}$, $|M\sigma| > \mathfrak{m}+1$ and that $M$ does not contain any constant, carry out the construction symmetric to the one in Case 3.
For simplicity we have ignored the nontermination caused by sequences like (22) since it introduces no new difficulty.
```

Figure 3: Nondeterministic Algorithm GT.

Lemma 8. *The algorithm GT always terminates.*

Proof. Suppose an execution of GT on an input pair (P, Q) does not abort. A path cannot contain an infinite number of small nodes, bearing in mind that there are only a finite number of constants. A long decreasing path may switch to a long right decreasing path, but not vice versa. By definition the execution path cannot contain, starting from an expansion node, a long left/right decreasing path longer than $\bar{t}(\mathfrak{q}\mathfrak{n}2^{\mathfrak{q}})$, nor can it contain any positive u-alias sequence longer than $\mathfrak{h}_0 + \mathfrak{h}_1 + \ldots + \mathfrak{h}_{\mathfrak{q}}$. Between any two such long decreasing paths/positive u-alias sequences there must be other kind of nodes. If a path contains a nonsink (balance) node, it contains its u-alias (v-alias). There cannot be any path that contains infinitely many u-aliases/v-aliases/d-aliases/w-aliases since they are all bounded in size. We conclude by König lemma that the output tree must be finite. □

We call the output of a successful execution of GT a *potentially generating tree* for the input pair (P, Q). It is a *generating tree* for (P, Q) if statement 3(b)iiA of Fig. 3 is valid throughout the execution of the algorithm. A graphical outline of a generating tree is given in Fig. 4, where $j' = \mathfrak{h}_0 + \mathfrak{h}_1 + \ldots + \mathfrak{h}_h$, and the nodes in boldface are leaves, and a node is separated from its alias by a horizontal line.



$(pX\alpha, M\sigma)$
$(p_i\alpha, M_i\sigma)$
$\frac{(L\alpha, N\sigma)}{(LU\sigma, N\sigma)}$
$(qZ, G_a\sigma_0)$
$(p_j\sigma, G_j\sigma)$
$(G_{j'}\sigma, p_{j'}\sigma)$
$(\mathbf{p_h}\sigma, \mathbf{p_k}\sigma)$
$\frac{(A\sigma, B\sigma)}{(AV, BV)}$
$\frac{(qZ, G_b\sigma_0)}{(G_aW, G_bW)}$
$(\mathbf{rZ}, \mathbf{rZ})$
$(L_{i_0}U_{i_0}\sigma_{i_0}, M_{i_0}\sigma_{i_0})$
(O, O')
$(r\sigma_{i_0}, M'_{i_0}\sigma_{i_0})$
$(qZ, G_a\sigma'_0)$
$(L_{i_0+1}U_{i_0+1}\delta_{i_0+1}\sigma_{i_0}, M_{i_0+1}\delta_{i_0+1}\sigma_{i_0})$
$\frac{(A\sigma', B\sigma')}{(\mathbf{AV}, \mathbf{BV})}$
$\frac{(qZ, G_b\sigma'_0)}{(\mathbf{G_aW}, \mathbf{G_bW})}$
$(L_{i_0+j}U_{i_0+j}\delta_{i_0+j}\sigma_{i_0}, M_{i_0+j}\delta_{i_0+j}\sigma_{i_0})$
$(\mathbf{O}, \mathbf{O}')$
$\frac{(L_{i_0+j'}U_{i_0+j'}\delta_{i_0+j'}\sigma_{i_0}, M_{i_0+j'}\delta_{i_0+j'}\sigma_{i_0})}{(L_{i_0+j'}U_{i_0+j'}\delta_{i_0+j'}V_h, M_{i_0+j'}\delta_{i_0+j'}V_h)}$

Figure 4: A Generating Tree.

6. Decidability

Suppose that some execution of GT on input process pair (P, Q) and input constants $\mathfrak{h}_0, \ldots, \mathfrak{h}_h$ returns a generating tree for (P, Q). Let's explain how to grow the generating tree into a bisimulation tree for (P, Q) *level by level*. Equivalently we define a procedure that upon receiving a node in the generating tree construct a bisimulation tree for the node in a coinductive manner. This is done inductively as follows.

1. Suppose $(A\sigma, B\sigma)$ is a balance node and (AV, BV) is its v-alias. The growth of $(A\sigma, B\sigma)$ simply duplicates the growth of (AV, BV).
2. Suppose $(pXU\sigma, M\sigma)$ is the u-alias of a nonsink node. Firstly notice that we can grow every balance node of the characteristic tree for $(pXU\sigma, M\sigma)$ as described in the previous case. We then simply compose the resulting tree level by level with the bisimulation trees for the recursive nodes $(p_i\sigma, L_i\sigma)$ or $(L_i\sigma, p_i\sigma)$ such that $|L_i| > 0$.
3. If U is defined as in (6), we can grow a bisimulation tree for $(pX\alpha, pXU\sigma)$ by appending the bisimulation tree for $(p_i\alpha, M_i\sigma)$. We can then grow a bisimulation tree for $(pX\alpha, M\sigma)$ by composing the bisimulation tree for $(pX\alpha, pXU\sigma)$ with the bisimulation tree for $(pXU\sigma, M\sigma)$, the construction of the latter is described in the previous case.
4. Let's now take a look at the expansion node $(qZ, G_b\sigma_0)$ as discussed before and after (16). We can grow a bisimulation tree for the alias $(G_a\sigma_0, G_b\sigma_0)$ by copying the growth of (G_aW, G_bW), and whenever the suffix σ_0 is exposed continuing the imitation by composing with the bisimulation trees for $(C_h, p_h\sigma_0)$ or $(p_h\sigma_0, C_h)$ for $h \in [\mathfrak{q}]$, confer (19) and (21). We then grow a bisimulation tree for $(qZ, G_b\sigma_0)$ by composing the bisimulation tree for $(qZ, G_a\sigma_0)$ with the bisimulation tree for $(G_a\sigma_0, G_b\sigma_0)$. See the left and middle diagrams in Fig. 5.
5. Let $(L_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_h} U_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_h} \delta_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_h} \sigma_{i_0}, M_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_h} \delta_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_h} \sigma_{i_0})$ be a deep balance node as in (24). A bisimulation tree for the deep balance node is obtained by duplicating the bisimulation tree for the alias
$$(L_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_h} U_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_h} \delta_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_h} V_h, M_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_h} \delta_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_h} V_h), \tag{25}$$
where V_h is the recursive constant guessed in Step 3(b)ii of GT and checked in Step 3(b)iiA. By definition the bisimulation tree for (25) is grown after the $\mathfrak{h}_j$-characteristic tree for
$$(L_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_j} U_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_j} \delta_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_j} \sigma_{i_0}, M_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_j} \delta_{i_0+\mathfrak{h}_0+\ldots+\mathfrak{h}_j} \sigma_{i_0}) \tag{26}$$
has been produced by GT for all $j < h$. So there is no circularity. See the right diagram in Fig. 5, where the small triangles are $(\mathfrak{m}-1)$-bisimulation trees that define normed constants. The point is that when the characteristic tree generating $V_{i_0+\mathfrak{h}_0}$ starts to grow, the growth of the $\mathfrak{h}_0$-characteristic tree generating V_{i_0} is already completed.

We conclude that if $GT(P, Q, \mathfrak{h}_0, \ldots, \mathfrak{h}_h)$ returns a generating tree then $P \sim Q$. Decidability proof is now a formality.

Theorem 9. *The strong bisimilarity for PDA is decidable.*

Proof. We only have to give a semidecidable procedure for $\sim$. Given a pair (P, Q), guess numbers $\mathfrak{h}_0, \ldots, \mathfrak{h}_{\mathfrak{q}}$ such that $0 < \mathfrak{h}_0 < \ldots < \mathfrak{h}_{\mathfrak{q}}$. Apply GT to construct a generating tree for (P, Q) nondeterministically. If the construction successfully outputs a potential generating tree and the output is in fact a generating tree, return 'yes'.

Checking if a successful output is a generating tree is equivalent to checking if statement 3(b)iiA can be validated. Referring to (24) we can firstly construct for each $j \leq h$ an $\mathfrak{h}_{j+1}$-bisimulation tree $\mathfrak{B}_j$ for (26) as described in the beginning of the section, and this can be done before GT processes the deep balance node. Secondly we can construct from $\mathfrak{B}_0$ and V_{i_0} an $\mathfrak{h}_1$-characteristic tree $\mathfrak{C}_0$ for $(L_{i_0+1} U_{i_0+1} \delta_{i_0+1} \sigma_{i_0}, M_{i_0+1} \delta_{i_0+1} \sigma_{i_0})$ over σ_{i_0} extending V_{i_0}. This can be done by the construction defined on page 6. The difference between the construction on page 6 and the present construction is that the former is semantic whereas the latter is algorithmic. Imagine we are growing $\mathfrak{B}_0$ level by level, and at the same time we define a recursive constant V^0. Initially V^0 is V_{i_0}. Whenever we come across a node of the form $(p_k\sigma_{i_0}, M'_k\sigma_{i_0})$, we check if V^0 is defined at k. If it is not defined at k we extend the definition of V^0 by $V^0(k) = M'_k$. Otherwise we compose the bisimulation tree for $(V^0(k)\sigma_{i_0}, p_k\sigma_{i_0})$ with the subtree of $\mathfrak{B}_0$ rooted at $(p_k\sigma_{i_0}, M'_k\sigma_{i_0})$. Now either $(V^0(k)\sigma_{i_0}, p_k\sigma_{i_0})$ and/or $(p_k\sigma_{i_0}, V^0(k)\sigma_{i_0})$ is a node of $\mathfrak{B}_{i_0}$ or $\mathfrak{B}_0$. So the bisimulation tree for $(V^0(k)\sigma_{i_0}, p_k\sigma_{i_0})$ is essentially a subtree of $\mathfrak{B}_{i_0}$ or $\mathfrak{B}_0$. In this way we get the $\mathfrak{h}_0$-characteristic tree $\mathfrak{C}_0$ by composition. Let V^0 be the recursive constant induced by $\mathfrak{C}_0$. Thirdly we can construct from $\mathfrak{B}_h$ and V^{h-1} an $\mathfrak{h}_{h+1}$-characteristic tree $\mathfrak{C}_h$ for the deep balance node over σ_{i_0} extending V^{h-1}, inducing a recursive constant V^h. The validation of statement 3(b)iiA is done by comparing V^h against V_h gussed by the algorithm. □

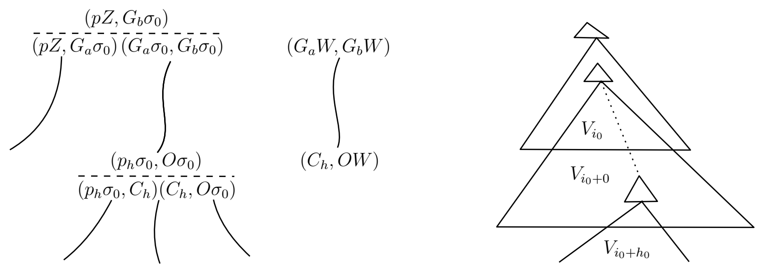

$(pZ, G_b\sigma_0)$
$(pZ, G_a\sigma_0)$ $(G_a\sigma_0, G_b\sigma_0)$
(G_aW, G_bW)
$(p_h\sigma_0, O\sigma_0)$
$(p_h\sigma_0, C_h)$ $(C_h, O\sigma_0)$
(C_h, OW)
V_{i_0}
V_{i_0+0}
$V_{i_0+h_0}$

Figure 5: Bisimulation Trees from Generating Trees.

7. Conclusion

All proofs of the decidability of equivalence checking problem for (D)PDA are based on Senizergues' fundamental observation on the recurrent behaviours of processes [22, 23, 24, 26, 27]. A pair of equivalent processes can be turned into an equivalent pair of prefix bounded processes using the balance strategy. The latter pair has finite behaviours by the duplication technique. By applying the two strategies in an orderly manner, using weights, conditional rules, etc., one hopes to construct a finite tree-like structure that witnesses the equivalence of two (D)PDA processes. The finite structure can be a proof system, a tableau, or a game. There are two difficulties when reasoning about the finite structures. One is to show the compositionality of them. For DPDA equivalence and strong bisimilarity of PDA compositionality is relative easy to establish. For branching bisimilarity compositionality depends crucially on a correct definition of finite structure. The main difficulty in decidability proof is to do with termination. In the light of König lemma termination is equivalent to the finiteness of tree-like structures. In Stirling's approach [29, 30, 32] the key is to argue that all tableaux are finite. In Jančar's proof [16, 17] the main step is to demonstrate the existence of an (n, g)-sequence convincing Refuter of the fact that he will never have any chance of winning.

Our motivation for this work is to give a simpler and more intuitive proof of the decidability of the strong bisimilarity of PDA. The endeavor is worthwhile for at least two reasons. Theoretically the strong bisimilarity of PDA poses one of the most challenging problems in the field of equivalence checking. Decidability proofs of simpler models and their variants in the general framework of process rewriting system [19] are a lot easier [5, 6, 7, 8, 38]. Any improvement of a decidability proof about PDA may have implications to the decidability proofs of related problems. For instance a more streamlined proof may help generalize the current results to models that admit far more liberal silent transitions than DPDA. At pragmatic level algorithms about PDA are important in programme analysis. A simpler and more intuitive proof may suggest how to constrain the PDA model to obtain efficient equivalence checking algorithms. This would be welcome in view of the fact that the general problem is highly complex [3]. It turns out for us that the idea of the proof is most easily explained directly in terms of strong bisimulations. It is well-known that bisimulations are closed under set union and composition. So the compositionality would not be an issue. Our termination conditions draw inspiration from Jančar's treatment to long prefix increasing sequence in the setting of Defender-Refuter game. The introduction of the unnormed constants follows the commmon practice in the study of PDA, that is to extend the basic PDA model so that the extended model enjoys better algebraic property. The structure of our decidability proof is basically Stirling's proof using tableau system plus our termination conditions. When one works with tableau systems one actually has in mind the bisimulation trees. Simplification is achieved by working directly with the latter objects.

The termination conditions of this paper can be applied to richer models. They are applied in the proof that the branching bisimilarity [36, 37, 2] is decidable for ϵ-pushing normed PDA [9] and for ϵ-popping PDA [10]. It remains to see if the proof methodology is useful in saying anything about the branching bisimilarity of finite turn PDA [25].

Acknowledgment

We thank BASICS members for frequent discussions on issues related to PDA. We thank particularly Qiang Yin for collaboration on the decidability of branching bisimilarity of PDA variants [9, 10], which has greatly enhanced our understanding of the subject. We also thank Petr Jančar for discussions on related issues. The support from NSFC (61472239, 61772336) is gratefully acknowledged.

References

[1] J. Baeten, J. Bergstra, and J. Klop. Decidability of Bisimulation Equivalence for Processes Generating Context-free Languages. *PARLE 1987*, Lecture Notes in Computer Science 259, 94-111, 1987.

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

[3] M. Benedikt, S. Göller, S. Kiefer, and A. Murawski. Bisimilarity of Pushdown Automata is Nonelementary. In *LICS'13*, 488-498, 2013.

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

[5] W. Czerwiński, P. Hofman, and S. Lasota. Decidability of branching bisimulation on normed commutative context-free processes. In *CONCUR'11*, pages 528-542. Lecture Notes in Computer Science 6901, Springer, 2011.

[6] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. In *CONCUR'92*, pages 138-147. Lecture Notes in Computer Science 630, Springer, 1992.

[7] S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation equivalence is decidable for basic parallel processes. In *Prodeedings of the 4th International Conference on Concurrency Theory (CONCUR'93)*, pages 143-157. Springer, 1993.

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

[9] Y. Fu and Q. Yin. Decidability of Epsilon Pushing PDA, 2018.

[10] Y. Fu and Q. Yin. Decidability of Epsilon Popping PDA, 2018.

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

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

[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*, 415-424. IEEE Computer Society, 2012.

[15] P. Jančar. Equivalences of Pushdown Systems are Hard. *Foundations of Software Science and Computation*, 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, 232-243, 2014.

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

[18] A. Kučera and P. Jančar. Equivalence-Checking on Infinite-State Systems: Techniques and Results. *Theory and Practice of Logic Programming*, 6:227-264, 2006.

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

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

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

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

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

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

[25] G. Sénizergues. The Equivalence Problem for t-Turn DPDA is co-NP. *ICALP 2003*, Lecture Notes in Computer Science 2719, 478-489, 2003.

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

[27] G. Sénizergues. Decidability of Bisimulation Equivalence for Equational Graphs of Finite Out-Degree. *SIAM Journal of Computing*, 34(5):1025-1106, 2005.

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

[29] C. Stirling. Decidability of Bisimulation Equivalence for Normed Pushdown Processes. In *CONCUR'96*, Lecture Notes in Computer Science, 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, 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] Stirling. An Introduction to Decidability of DPDA Equivalence. In *FSTTCS'01*, Lecture Notes in Computer Science 2245, 42-56. Springer, 2001.

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

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

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

[38] 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, 363-374, 2014.