# Branching Bisimilarity Checking for PRS

Qiang Yin, Yuxi Fu, Chaodong He, Mingzhang Huang, and Xiuting Tao

BASICS, Department of Computer Science, Shanghai Jiao Tong University

**Abstract.** Recent studies reveal that branching bisimilarity is decidable for both nBPP (normed Basic Parallel Processes) and nBPA (normed Basic Process Algebras). These results lead to the question if there are any other models in the hierarchy of PRS (Process Rewrite Systems) whose branching bisimilarity is decidable. It is shown in this paper that the branching bisimilarity for both nOCN (normed One Counter Nets) and nPA (normed Process Algebras) is undecidable. These results essentially imply that the question has a negative answer.

## 1 Introduction

Verification on infinite-state systems has been intensively studied for the past two decades [2,12]. One major concern in these studies is equivalence checking. Given a specification $\mathcal{S}$ of an intended behaviour and a claimed implementation $\mathcal{I}$ of $\mathcal{S}$, one is supposed to demonstrate that $\mathcal{I}$ is correct with respect to $\mathcal{S}$. A standard interpretation of correctness is that an implementation should be behaviourally equivalent to its specification. Among all the behavioural equalities studied so far, bisimilarity stands out as the most abstract and the most tractable one. Two well known bisimilarities are the strong bisimilarity and the weak bisimilarity due to Park [16] and Milner [15]. Considerable amount of effort has been made to investigate the decidability and the algorithmic aspect of the two bisimilarities on various models of infinite state system [18]. These models include pushdown automata, process algebras, Petri nets and their restricted and extended variations. An instructive classification of the models in terms of PRS (Process Rewrite Systems) is given by Mayr [13].

The strong bisimilarity checking problem has been well studied for PRS hierarchy. Influential decidability results include for example [1,4,3,21,8]. On the negative side, Jančar attained in [9] the undecidable result of strong bisimilarity on nPN (normed Petri Nets). The proof makes use of a powerful technique now known as Defender's Forcing [11], which remains a predominant tool to establish negative results about equivalence checking.

In the weak case the picture is less clear. It is widely believed that weak bisimilarity is decidable for both nBPA (normed Basic Process Algebras) and nBPP (normed Basic Parallel Processes). The problem has been open for a long time. Srba [17] showed that weak bisimilarity on nPDA (normed Pushdown Automata) is undecidable by a reduction from the halting problem of Minsky Machine. The undecidability was soon extended to nOCN (normed One Counter Nets), a submodel of both nPDA and nPN, by Mayr [14]. Srba also showed that

| | nBPA | nBPP | nPDA | nPA | nPN |
|---|---|---|---|---|---|
| Strong Bisimilarity | ✓[1] | ✓[3] | ✓[21] | ✓[8] | ×[9] |
| Branching Bisimilarity | ✓[7] | ✓[5] | ×[this paper] | ×[this paper] | ×[9] |
| Weak Bisimilarity | ? | ? | ×[14] | ×[this paper] | ×[9] |

**Fig. 1.** Decidability of Branching Bisimilarity for Normed PRS

the weak bisimilarity on PA (Process Algebras) is undecidable [19]. Later several highly undecidable results were established by Jančar and Srba [20,10,11] for the weak bisimilarity checking problem on PN, PDA and PA.

The decidability of the weak bisimilarity on nBPA and nBPP has been open for well over twenty years. Encouraging progress has been made recently. Czerwiński, Hofman and Lasota proved that branching bisimilarity, a standard refinement of the weak bisimilarity, is decidable on nBPP [5]. The novelty of their approach is the discovery of some kind of normal form for nBPP. Using a quite different technique Fu showed that the branching bisimilarity is also decidable on nBPA [7]. In retrospect one cannot help thinking that more attention should have been paid to the branching bisimilarity. Going back to the original motivation to equivalence checking, one would agree that a specification $\mathcal{S}$ normally contains no silent actions because silent actions are about how-to-do. Consequently all the silent actions introduced in an implementation must be bisimulated vacuously by the specification. It follows that $\mathcal{S}$ is weakly bisimilar to an implementation $\mathcal{I}$ if and only if $\mathcal{S}$ is branching bisimilar to $\mathcal{I}$. What this observation tells us is that as far as verification is concerned the branching bisimilarity ought to play a role no less than the weak bisimilarity.

The above discussion suggests to address the following question: Is there any other model in the PRS hierarchy whose branching bisimilarity is decidable? The purpose of this paper is to resolve this issue. Our contributions are as follows:

– We establish the fact that on both nOCN and nPA every relation between the branching bisimilarity and the weak bisimilarity is undecidable. These are improvement of Mayr's result about the undecidability of the weak bisimilarity on nOCN [14] and Srba's result [19] about the undecidability of the weak bisimilarity on PA. These new results together with the previous (un)decidability results about the *normed* models in PRS are summarized in Fig. 1, where a tick is for 'decidable' and a cross for 'undecidable'.
– We showcase the subtlety of Defender's Forcing technique usable in branching bisimulation game. It is pointed out that the technique must be of a semantic nature for it to be applicable to the branching bisimilarity.

The two negative results imply that in the PRS hierarchy the branching bisimilarity on every normed model above either nBPA or nBPP is undecidable.

The rest of the paper is organized as follows. Section 2 introduces the necessary preliminaries. Section 3 establishes the undecidability result for nOCN and demonstrates Defender's Forcing technique for branching bisimulation game. Section 4 proves the undecidability result about nPA. Section 5 concludes.

## 2 Preliminaries

A *process algebra* $\mathcal{P}$ is a triple $(\mathcal{C}, \mathcal{A}, \Delta)$, where $\mathcal{C}$ is a finite set of process constants, $\mathcal{A}$ is a finite set of actions ranged over by $\ell$, and $\Delta$ is a finite set of transition rules. The *processes* defined by $\mathcal{P}$ are generated by the following grammar:

$$P ::= \epsilon \mid X \mid PP' \mid P \,\|\, P'.$$

The grammar equality is denoted by $=$. We assume that the sequential composition $PP'$ is associative up to $=$ and the parallel composition $P \,\|\, P'$ is associative and commutative up to $=$. We also assume that $\epsilon P = P\epsilon = \epsilon \,\|\, P = P \,\|\, \epsilon = P$. There is a special symbol $\tau$ in $\mathcal{A}$ for silent transition. The set $\mathcal{A} \setminus \{\tau\}$ is ranged over by $a, b, c, d$. The transition rules in $\Delta$ are of the form $X \xrightarrow{\ell} P$. The following labeled transition rules define the operational semantics of the processes.

$$\frac{X \xrightarrow{\ell} P \in \Delta}{X \xrightarrow{\ell} P} \qquad \frac{P \xrightarrow{\ell} P'}{PQ \xrightarrow{\ell} P'Q} \qquad \frac{P \xrightarrow{\ell} P'}{P \,\|\, Q \xrightarrow{\ell} P' \,\|\, Q} \qquad \frac{Q \xrightarrow{\ell} Q'}{P \,\|\, Q \xrightarrow{\ell} P \,\|\, Q'}$$

The operational semantics is structural, meaning that $PQ \xrightarrow{\ell} P'Q$, $P \,\|\, Q \xrightarrow{\ell} P' \,\|\, Q$ and $Q \,\|\, P \xrightarrow{\ell} Q \,\|\, P'$ whenever $P \xrightarrow{\ell} P'$. We write $\Longrightarrow$ for the reflexive transitive closure of $\xrightarrow{\tau}$, and $\overset{\widehat{\ell}}{\Longrightarrow}$ for $\Longrightarrow\xrightarrow{\ell}\Longrightarrow$ if $\ell \neq \tau$ and for $\Longrightarrow$ otherwise.

A *one counter net* $\mathcal{M}$ is a 4-tuple $(\mathcal{Q}, X, \mathcal{A}, \Delta)$, where $\mathcal{Q}$ is a finite set of states ranged over by $p, q, r, s$, $X$ represents a place, $\mathcal{A}$ is a finite set of actions as in a process algebra, and $\Delta$ is a finite set of transition rules. A *process* defined by $\mathcal{M}$ is of the form $pX^n$, where $n$ indicates the number of tokens in $X$. A transition rule in $\Delta$ is of the form $pX^i \xrightarrow{\ell} qX^j$ with $i < 2$. The semantics is structural in the sense that $pX^{i+k} \xrightarrow{\ell} qX^{j+k}$ whenever $pX^i \xrightarrow{\ell} qX^j$. A process $P$ defined in $\mathcal{P}$, respectively $\mathcal{M}$, is *normed* if $\exists \ell_1, \ldots, \ell_n . P \xrightarrow{\ell_1} \ldots \xrightarrow{\ell_n} \epsilon$, respectively $\exists \ell_1, \ldots, \ell_n, p.(P \xrightarrow{\ell_1} \ldots \xrightarrow{\ell_n} p) \wedge \forall \ell, Q. \neg(p \xrightarrow{\ell} Q)$. We say that $\mathcal{P}/\mathcal{M}$ is normed if only normed processes are definable in it. We write (n)PA for the (normed) Process algebras and (n)OCN for the (normed) One Counter Nets.

In the presence of silent actions two well known process equalities are the weak bisimilarity [15] and the branching bisimilarity [24].

**Definition 1.** *A relation $\mathcal{R}$ is a* weak bisimulation *if the following are valid:*

*1. Whenever $P\mathcal{R}Q$ and $P \xrightarrow{\ell} P'$, then $Q \overset{\widehat{\ell}}{\Longrightarrow} Q'$ and $P'\mathcal{R}Q'$ for some $Q'$.*

*2. Whenever $P\mathcal{R}Q$ and $Q \xrightarrow{\ell} Q'$, then $P \overset{\widehat{\ell}}{\Longrightarrow} P'$ and $P'\mathcal{R}Q'$ for some $P'$.*

*The* weak bisimilarity $\approx$ *is the largest weak bisimulation.*

**Definition 2.** *A relation $\mathcal{R}$ is a* branching bisimulation *if the following hold:*

*1. Whenever $P\mathcal{R}Q$ and $P \xrightarrow{\ell} P'$, then either (i) $Q \Longrightarrow Q'' \xrightarrow{\ell} Q'$ and $P'\mathcal{R}Q'$ and $P\mathcal{R}Q''$ for some $Q', Q''$ or (ii) $\ell = \tau$ and $P'\mathcal{R}Q$.*

*2. Whenever $P\mathcal{R}Q$ and $Q \xrightarrow{\ell} Q'$, then either (i) $P \Longrightarrow P'' \xrightarrow{\ell} P'$ and $P'\mathcal{R}Q'$ and $P''\mathcal{R}Q$ for some $P', P''$ or (ii) $\ell = \tau$ and $P\mathcal{R}Q'$.*

*The* branching bisimilarity $\simeq$ *is the largest branching bisimulation.*

The following lemma, first noticed by van Glabbeek and Weijland [24], plays a fundamental role in the study of branching bisimilarity.

**Lemma 1.** *If* $P \Longrightarrow P' \Longrightarrow P'' \simeq P$ *then* $P' \simeq P$.

Let $\approxeq$ be a process equivalence. A silent action $P \xrightarrow{\tau} P'$ is *state preserving* with regards to $\approxeq$, notation $P \to P'$, if $P' \approxeq P$; it is *change-of-state* with regards to $\approxeq$, notation $P \xrightarrow{\iota} P'$, if $P' \not\approxeq P$. The reflexive and transitive closure of $\to$ is denoted by $\to^*$. Branching bisimilarity strictly refines weak bisimilarity in the sense that only state preserving silent actions can be ignored; a change-of-state must be explicitly bisimulated. Suppose that $P \simeq Q$ and $P \xrightarrow{\ell} P'$ is matched by the transition sequence $Q \xrightarrow{\tau} \cdots \xrightarrow{\tau} Q_i \xrightarrow{\tau} \cdots \xrightarrow{\tau} Q'' \xrightarrow{\ell} Q'$. By definition one has $P \simeq Q''$. It follows from Lemma 1 that $P \simeq Q_i$, meaning that all silent actions in $Q \Longrightarrow Q''$ are necessarily state preserving. This property fails for the weak bisimilarity as the following example demonstrates.

*Example 1.* Consider the transition system $\{P \xrightarrow{b} \epsilon,\ P \xrightarrow{\tau} P' \xrightarrow{a} \epsilon,\ P \xrightarrow{a} \epsilon;\ Q \xrightarrow{b} \epsilon,\ Q \xrightarrow{\tau} Q' \xrightarrow{a} \epsilon\}$. One has $P \approx Q$. However $P \not\simeq Q$ since $Q \not\simeq Q'$.

A game theoretic characterization of bisimilarity is by *bisimulation game* [22]. Suppose that a pair of processes $P, Q$, called a *configuration*, are defined in say a process algebra $(\mathcal{C}, \mathcal{A}, \varDelta)$. A *branching bisimulation game* for the configuration $(P, Q)$ is played between *Attacker* and *Defender*. The game is played in *rounds*. A new configuration is chosen after each round. Every round consists of three steps defined as follows, assuming $(P_0, P_1)$ is the current configuration:

1. Attacker chooses $i \in \{0,1\}$, $\ell \in \mathcal{A}$ and some process $P_i'$ such that $P_i \xrightarrow{\ell} P_i'$.
2. Defender may respond in either of the following manner:
   - Choose some $P_{1-i}', P_{1-i}''$ such that $P_{1-i} \Longrightarrow P_{1-i}'' \xrightarrow{\ell} P_{1-i}'$.
   - Do nothing in the case that $\ell = \tau$.
3. Attacker decides which of $(P_i, P_{1-i}'')$, $(P_i', P_{1-i}')$ is the new configuration if Defender has played. Otherwise the new configuration must be $(P_i', P_{1-i})$.

In a *weak bisimulation game* a round consists of two steps. The first step is the same as above. In the second step Defender chooses some $P_{1-i}'$ and some transition sequence $P_{1-i} \overset{\widehat{\ell}}{\Longrightarrow} P_{1-i}'$. The game then continues with $(P_i', P_{1-i}')$.

Defender wins a game if it never gets stuck; otherwise Attacker wins. We say that Defender/Attacker has a *winning strategy* if it can always win no matter how the opponent plays. The following lemma is well known, a clever use of which often simplifies bisimulation argument considerably.

**Lemma 2.** *Defender has a winning strategy in the branching, respectively weak, bisimulation game starting from the configuration* $(P, Q)$ *if and only if* $P \simeq Q$, *respectively* $P \approx Q$.

Attacker has a winning strategy for the branching bisimulation game of the pair $P, Q$ defined in Example 1. It simply chooses $P \xrightarrow{a} \epsilon$. If Defender chooses $Q \xrightarrow{\tau} Q' \xrightarrow{a} \epsilon$, Attacker chooses the configuration $(P, Q')$ and wins. Defender can win the weak bisimulation game of $(P, Q)$ though.

## 3 Defender's Forcing with Delayed Justification

A powerful technique for proving lower bounds for bisimilarity checking problem is Defender's Forcing described by Jančar and Srba in [11]. The basic idea is to force Attacker to make a particular choice in a bisimulation game by introducing enough copycat rules. An application of the technique to weak bisimulation game should be careful since both Attacker and Defender can take advantage of silent transitions. The design of a branching bisimulation game is even more subtle. In such a game a sequence of silent transitions used by Defender, except possibly the last one, must all be state preserving. A useful technique, motivated by Lemma 1, is to make use of generating processes. The process $G$ defined by the rules $G \xrightarrow{\tau} GX$ and $GX \xrightarrow{\tau} G$ is *generating* due to the fact that every process that $G$ may evolve into, say $GX^n$, is branching bisimilar to $G$. The presence of other transition rules for $G$ and $X$ would not change the fact that $G \simeq GX^n$ for all $n$. This technique has already been used in the design of weak bisimulation games [11,14]. The relations these games give rise to are not branching bisimulation because a state-preserving transition may be simulated by a change-of-state silent transition. In what follows we use a small example to expose the subtlety of branching bisimulation game and the technique to apply Defender's Forcing in such a game.

Mayr proved in [14] a general result that the weak bisimilarity is undecidable for any model that subsumes nOCN. The lower bound is achieved by reducing from the halting problem of Minsky machine. A Minsky machine $\mathcal{M}$ with two counters $c_1, c_2$ is a program of the form $1 : I_1;\ 2 : I_2;\ \ldots;\ m{-}1 : I_{m-1};\ m :$ halt, where for each $i \in \{1, \ldots, m-1\}$ the instruction $I_i$ is in either of the following forms, assuming $j, k \in \{1, \ldots, m-1\}$ 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 $k$.

By encoding a pair of numbers $(n_1, n_2)$ by Gödel number of the form $2^{n_1}3^{n_2}$, Mayr implemented the increment and decrement operations on the counters by multiplying and dividing by 2 and 3 respectively. The central part of Mayr's proof is to show that it is possible to encode these operations and test for divisibility by constant into weak bisimulation games on nOCN. We shall show that Mayr's reduction can be strengthened to produce reductions to branching bisimulation games on nOCN. For every instruction "$i : I_i$" of a Minsky machine $\mathcal{M}$ a pair of states $p_i, p'_i$ are introduced. Suppose "$i : c_2 := c_2 + 1$; goto $j$" is the i-th instruction of $\mathcal{M}$. The instruction is translated to the rules given in Fig. 2. The model defined in Fig. 2 is open-ended. Transition rules associated to $p_j$ and $p'_j$ are not given. We have however the following interesting property.

**Lemma 3.** *Let $n = 2^{n_1}3^{n_2}$ for some $n_1, n_2$. Defender of the branching bisimulation game of $(p_jX^{3n}, p'_jX^{3n})$ has a winning strategy if and only if Defender of the branching bisimulation game of $(p_iX^n, p'_iX^n)$ has a winning strategy.*

*Proof.* The crucial point here is that the copycat rules $p_i \xrightarrow{\tau} G'$ and $p'_i \xrightarrow{\tau} G'$, which syntactically identify what $p_iX^n$ and $p'_iX^n$ may reach in one silent step,

| | |
|---|---|
| $p_i \xrightarrow{\tau} G'$ | $p'_i \xrightarrow{\tau} G'$ |
| $p_i \xrightarrow{a} q_1$ | $G' \xrightarrow{a} q'_1,\ G' \xrightarrow{\tau} G'X,\ G'X \xrightarrow{\tau} G'$ |
| $q_1 \xrightarrow{a} q_2$ | $q'_1 \xrightarrow{a} q'_2$ |
| $q_1 \xrightarrow{t} t_3$ | $q'_1 \xrightarrow{t} t_1$ |
| $q_2 \xrightarrow{\tau} G$ | $q'_2 \xrightarrow{\tau} G$ |
| $G \xrightarrow{\tau} GX,\ GX \xrightarrow{\tau} G,\ G \xrightarrow{a} q_3$ | $q'_2 \xrightarrow{a} q'_3$ |
| $q_3 \xrightarrow{a} p_j$ | $q'_3 \xrightarrow{a} p'_j$ |
| $q_3 \xrightarrow{t} t_1$ | $q'_3 \xrightarrow{t} t_1$ |
| $t_3X \xrightarrow{c} t''X,\ t''X \xrightarrow{c} t'X,\ t'X \xrightarrow{c} t_3$ | $t_1X \xrightarrow{c} t_1$ |

**Fig. 2.** Multiplication Operation on Counter in OCN

do not automatically create a Defender's Forcing situation. The reason is that although $p'_iX^n \to G'X^n$, since $p'_iX^n \xrightarrow{\tau} G'X^n$ is the only action of $p'_iX^n$, it might well be that $p_iX^n \xrightarrow{\iota} G'X^n$. For branching bisimulation syntactical Defender's Forcing is insufficient. One needs Defender's Forcing that works at semantic level. Let's take a look at the development of the game in some detail.

1. If Attacker plays $p_iX^n \xrightarrow{\tau} G'X^n$, Defender plays $p'_iX^n \xrightarrow{\tau} G'X^n$. By Lemma 1 this response is equivalent to any other response from Defender.
2. If Attacker chooses the action $p_iX^n \xrightarrow{a} q_1X^n$, Defender responds with $p'_iX^n \to G'X^n \to^* G'X^{3n} \xrightarrow{a} q'_1X^{3n}$, making use of Lemma 1. Attacker's optimal move is to choose $(q_1X^n, q'_1X^{3n})$ to be the next configuration.
3. Now Attacker would not do a $t$ action since $t_3X^n \simeq t_1X^{3n}$. It chooses the action $a$ and the new configuration $(q_2X^n, q'_2X^{3n})$.
4. Then we come to another semantic Defender's Forcing. If Attacker plays $q_2X^n \xrightarrow{\tau} GX^n$, Defender plays $q'_2X^n \xrightarrow{\tau} GX^{3n}$; and vice versa.
5. If Attacker chooses the transition $q'_2X^{3n} \xrightarrow{a} q'_3X^{3n}$, Defender's response is $q_2X^n \xrightarrow{\tau} GX^n \Longrightarrow GX^{3n} \xrightarrow{a} q_3X^{3n}$, exploiting again Lemma 1. Attacker's nontrivial choice of the new configuration is $(q_3X^{3n}, q'_3X^{3n})$.
6. Finally Attacker would not choose a $t_1$ action since $t_1X^{3n} \simeq t_1X^{3n}$. So after an $a$ action, the configuration becomes $(q_jX^{3n}, q'_jX^{3n})$.

It is easy to see that the configuration $(q_jX^{3n}, q'_jX^{3n})$ is optimal for both Attacker and Defender. If $q_jX^{3n} \simeq q'_jX^{3n}$ then Defender's Forcing described above is justified. If $q_jX^{3n} \not\simeq q'_jX^{3n}$ the forcing is ineffective since Attacker can choose to play $p_iX^n \xrightarrow{\tau} G'X^n$ and wins. □

The main result of the section follows easily from Lemma 3 and its proof.

**Theorem 1.** *On* nOCN *every relation $\mathcal{R}$ satisfying $\simeq \subseteq \mathcal{R} \subseteq \approx$ is undecidable.*

*Proof.* Dividing a number by a constant can be encoded in similar fashion. The rest of Mayr's reduction does not refer to any silent transitions. It follows that we can construct a reduction witnessing that "$\mathcal{M}$ halts iff $p_1X \not\simeq p'_1X$". As a matter of fact the reduction supports the stronger correspondence stated as follows: "$\mathcal{M}$ halts iff $p_1X \not\approx p'_1X$". □

## 4 Undecidability of nPA

Following [19], our main undecidability result is proved by reducing PCP (Post's Correspondence Problem) to the branching bisimilarity checking problem on nPA. Suppose $\Sigma$ is a finite set of symbols and $\Sigma^+$ is the set of nonempty finite strings over $\Sigma$. The size of $\Sigma$ is at least two. PCP is defined as follows.

Post's Correspondence Problem
*Input:* $\{(u_1, v_1), (u_2, v_2) \ldots (u_n, v_n) \mid u_i, v_i \in \Sigma^+\}$.
*Problem:* Are there $i_1, i_2, \ldots i_m \in \{1, 2, \ldots, n\}$ with $m \geq 1$ such that $u_{i_1} u_{i_2} \ldots u_{i_m} = v_{i_1} v_{i_2} \ldots v_{i_m}$?

We will fix a PCP instance INST=$\{(u_1, v_1), (u_2, v_2) \ldots (u_n, v_n) \mid u_i, v_i \in \Sigma^+\}$ in this section. Our task is to construct a normed process algebra $\mathcal{G}$=$(\mathcal{C}, \mathcal{A}, \Delta)$ containing two process constants $X, Y$ that render true the following equivalence.

$$\text{“INST has a solution” iff } X \simeq Y \text{ iff } X \approx Y. \tag{1}$$

We will prove (1) by validating the following statements:

– “If INST has a solution then $X \simeq Y$”. This is Lemma 6 of Section 4.4.
– “If INST has no solution then $X \not\approx Y$”. This is Lemma 7 of Section 4.4.

As $X \simeq Y$ implies $X \approx Y$, the main theorem of the paper follows from (1).

**Theorem 2.** *On* nPA *every relation $\mathcal{R}$ satisfying $\simeq \subseteq \mathcal{R} \subseteq \approx$ is undecidable.*

In the rest of the section, we firstly define $\mathcal{G}$, and then argue in several steps how the game based on $\mathcal{G}$ works in Defender's favour if INST has a solution.

### 4.1 The nPA Game

The construction of $\mathcal{G} = (\mathcal{C}, \mathcal{A}, \Delta)$ from INST is based on Srba's reduction [19]. Substantial amount of redesigning effort is necessary to make it work for the *branching* bisimilarity on the *normed* PA. The set $\mathcal{A}$ of actions is defined by

$$\mathcal{A} = \Lambda \cup \mathcal{N} \cup \Sigma \cup \{\tau\},$$

where $\Lambda = \{\lambda_U, \lambda_V, \lambda_D, \lambda_I, \lambda_S, \lambda_Z\}$, $\mathcal{N} = \{1, \ldots, n\}$ and $\Sigma, n$ are from INST. The set $\mathcal{C}$ of process constants is defined by

$$\begin{aligned}
\mathcal{C} &= \{X, Y, Z, I, S, C, C', D, G, G', G_u, G_v, G'_v\} \cup \mathcal{U} \cup \mathcal{V} \cup \mathcal{W}, \\
\mathcal{U} &= \{U_i \mid i \in \mathcal{N}\}, \\
\mathcal{V} &= \{V_i \mid i \in \mathcal{N}\}, \\
\mathcal{W} &= \{W(\omega, i), W(\omega, 0) \mid \omega \in (\mathcal{SF}(u_i) \cup \mathcal{SF}(v_i)) \text{ and } i \in \mathcal{N}\},
\end{aligned}$$

where for each $\omega \in \Sigma^*$, the notation $\mathcal{SF}(\omega)$ stands for the set of suffixes of $\omega$. The set of transition rules is given in Fig. 3. It is clear from these rules that $\mathcal{G}$ is indeed normed. In particular $P \Longrightarrow \epsilon$ for all $P \in \mathcal{U} \cup \mathcal{V} \cup \mathcal{W}$.

We write $\mathbb{P}_u$, respectively $\mathbb{P}_v$, for a sequential composition of members of $\mathcal{U}$, respectively $\mathcal{V}$. Similarly we write $\mathbb{P}$, respectively $\mathbb{Q}$, for a sequential composition of members of $\mathcal{U} \cup \mathcal{V}$, respectively $\mathcal{U} \cup \mathcal{V} \cup \mathcal{W}$. If for example the sequence $u$ is empty, $\mathbb{P}_u$ is understood to denote $\epsilon$.

$$X \xrightarrow{\lambda_U} D \,\|\, G_v,\ X \xrightarrow{\tau} D; \quad Y \xrightarrow{\tau} D; \quad D \xrightarrow{\tau} D \,\|\, G_u,\ D \xrightarrow{\lambda_D} C;$$
$$G_u \xrightarrow{\tau} G_uU_i,\ G_u \xrightarrow{\lambda_U} G_vU_i; \quad G_u \xrightarrow{\tau} G'_v,\ G'_v \xrightarrow{\tau} G'_vV_i,\ G'_v \xrightarrow{\tau} Z;$$

$$G_v \xrightarrow{\tau} G_vV_i,\ G_v \xrightarrow{\tau} \epsilon,\ G_v \xrightarrow{\lambda_V} Z; \quad Z \xrightarrow{\tau} \epsilon,\ Z \xrightarrow{\lambda_Z} \epsilon;$$

$$C \xrightarrow{\lambda_I} I,\ C \xrightarrow{\lambda_S} S,\ C \xrightarrow{\tau} C \,\|\, G,\ C \xrightarrow{\tau} C \,\|\, G_v;$$
$$G \xrightarrow{\tau} GU_i,\ G \xrightarrow{\tau} GV_i,\ G \xrightarrow{\tau} \epsilon;$$

$$I \xrightarrow{\lambda_I} C',\ I \xrightarrow{i} I; \quad S \xrightarrow{\lambda_S} C',\ S \xrightarrow{a} S; \quad C' \xrightarrow{\tau} C' \,\|\, G',\ C' \xrightarrow{\tau} \epsilon;$$
$$G' \xrightarrow{\tau} G'U_i,\ G' \xrightarrow{\tau} G'V_i,\ G' \xrightarrow{\tau} G'W,\ G' \xrightarrow{\tau} G_v,\ G' \xrightarrow{\tau} Z;$$

$$U_i \xrightarrow{\tau} W(u_i,i),\ V_i \xrightarrow{\tau} W(v_i,i);$$
$$W(a\omega,i) \xrightarrow{a} W(\omega,i),\ W(a\omega,0) \xrightarrow{a} W(\omega,0),\ W(\omega,i) \xrightarrow{i} W(\omega,0),$$
$$W(a\omega,i) \xrightarrow{\tau} W(\omega,i),\ W(a\omega,0) \xrightarrow{\tau} W(\omega,0),\ W(\omega,i) \xrightarrow{\tau} W(\omega,0),\ W(\epsilon,0) \xrightarrow{\tau} \epsilon.$$

In the above rules, $i$ ranges over $\{1,\ldots,n\}$, $a$ ranges over $\Sigma$, and $W$ ranges over $\mathcal{W}$.

**Fig. 3.** Transition Rules for the nPA Game

## 4.2 Defender's Generator

To explain how the reduction works we start with the generators introduced by the process algebra. A generator should be able to not only produce what is necessary but also do away with what has been produced. The process $D$ for instance can induce circular silent transition sequence of the form

$$D \xrightarrow{\tau} D \,\|\, G_u \Longrightarrow D \,\|\, G_u\mathbb{P}_u \xrightarrow{\tau} D \,\|\, G'_v\mathbb{P}_u \Longrightarrow D \,\|\, G'_v\mathbb{P}_v\mathbb{P}_u \Longrightarrow D.$$

By Lemma 1 all the processes appearing in the above sequence are branching bisimilar. Notice that the only reason the process constant $G'_v$ is introduced is to make available the above circular sequence. The constant $G'_v$ is necessary because $G_u$ cannot reach $G_v$ via silent moves. Similar circular silent transition sequences are also available for $C$ and $C'$.

**Lemma 4.** *Suppose $P \in \{D, C, C'\}$ and $P \Longrightarrow P \,\|\, Q$. Then $P \,\|\, Q \Longrightarrow P$.*

**Corollary 1.** *The following equalities are valid for all $\mathbb{P}_u, \mathbb{P}_v, \mathbb{P}, \mathbb{Q}$.*

1. $D \simeq D \,\|\, G_u\mathbb{P}_u \simeq D \,\|\, G'_v\mathbb{P}_v\mathbb{P}_u \simeq D \,\|\, Z\mathbb{P}_v\mathbb{P}_u \simeq D \,\|\, \mathbb{P}_v\mathbb{P}_u \simeq D \,\|\, W\mathbb{P}_v\mathbb{P}_u$;
2. $C \simeq C \,\|\, G\mathbb{P} \simeq C \,\|\, \mathbb{P} \simeq C \,\|\, W\mathbb{P} \simeq C \,\|\, G_v\mathbb{P}_v$;
3. $C' \simeq C' \,\|\, G'\mathbb{Q} \simeq C' \,\|\, G_v\mathbb{Q} \simeq C' \,\|\, Z\mathbb{Q} \simeq C' \,\|\, \mathbb{Q}$.

It has been observed that generating transitions are the most tricky ones in decidability proofs [23,5,7]. Here they are used to Defender's advantage. A generator can start everything all over again from scratch. This gives Defender the ability to copy Attacker if the latter does not make a particular move.

The bisimulation game of $(X, Y)$ is played in two phases. The generating phase comes first. During this phase Defender tries to produce a pair $\mathbb{P}_u, \mathbb{P}_v$, via Defender's Forcing using the generators, that encode a solution to INST. Next comes the checking phase in which Attacker tries to reject the pair $\mathbb{P}_u, \mathbb{P}_v$. In the light of the delayed effect of Defender's Forcing in branching bisimulation games, we will look at the two phases in reverse order.

### 4.3 Checking Phase

The processes $U_i, V_i$ play two roles. One is to announce $u_i$, respectively $v_i$; the other is to reveal the index $i$. The first role can be suppressed by composing $U_i$, respectively $V_i$, with $S$ while the second can be discharged by composing with $I$ [19]. Since $I, S$ are normed, Attacker can choose to remove $I$, respectively $S$. In our game the removal can be done by playing $I \xrightarrow{\lambda_I} C'$, respectively $S \xrightarrow{\lambda_S} C'$. According to (3) of Corollary 1 however Attacker would lose immediately if it plays $I \xrightarrow{\lambda_I} C'$, respectively $S \xrightarrow{\lambda_S} C'$, in a branching bisimulation game starting from $(I \parallel \mathbb{Q}, I \parallel \mathbb{Q}')$, respectively $(S \parallel \mathbb{Q}, S \parallel \mathbb{Q}')$. Notice that it is important for a process constant $W$ to ignore the string/index information by doing silent transitions. Otherwise the interleaving between actions in $\Sigma$ and actions in $\mathcal{N}$ would defeat Defender's attempt to prove string/index equality.

**Lemma 5.** *Suppose* $\mathbb{U} = U_{i_1} U_{i_2} \ldots U_{i_l}$, $\mathbb{V} = V_{j_1} V_{j_2} \ldots V_{j_r}$ *and* $B \in \{\epsilon, Z, G_v\}$. *The following statements are valid, where* $\approxeq \in \{\simeq, \approx\}$.

1. $I \parallel B\mathbb{P}\mathbb{U} \approxeq I \parallel B\mathbb{P}\mathbb{V}$ *if and only if* $u_{i_1} u_{i_2} \ldots u_{i_l} = v_{j_1} v_{j_2} \ldots v_{j_r}$.
2. $S \parallel B\mathbb{P}\mathbb{U} \approxeq S \parallel B\mathbb{P}\mathbb{V}$ *if and only if* $i_1 i_2 \ldots i_l = j_1 j_2 \ldots j_r$.

*Proof.* Suppose $I \parallel B\mathbb{P}\mathbb{U} \simeq I \parallel B\mathbb{P}\mathbb{V}$ and w.l.o.g. $|u_{i_1} u_{i_2} \ldots u_{i_l}| \geq |v_{j_1} v_{j_2} \ldots v_{j_r}|$. An action sequence from $I \parallel B\mathbb{P}\mathbb{U}$ to $I \parallel \mathbb{U}$ must be simulated essentially by an action sequence from $I \parallel B\mathbb{P}\mathbb{V}$ to $I \parallel \mathbb{V}$. But then $u_{i_1} u_{i_2} \ldots u_{i_l} = v_{j_1} v_{j_2} \ldots v_{j_r}$ can be derived from $I \parallel \mathbb{U} \simeq I \parallel \mathbb{V}$. The converse implication follows from the discussion in the above. The second equivalence can be proved similarly. □

The following proposition, in which $\approxeq \in \{\simeq, \approx\}$, says that the constant $C$ can be used to check both string equality and index equality by Attacker's forcing.

**Proposition 1.** *If* $\mathbb{U} = U_{i_1} U_{i_2} \ldots U_{i_l}$ *and* $\mathbb{V} = V_{j_1} V_{j_2} \ldots V_{j_r}$, *then for all* $\mathbb{P}$, $C \parallel Z\mathbb{P}\mathbb{U} \approxeq C \parallel Z\mathbb{P}\mathbb{V}$ *iff* $i_1 i_2 \ldots i_l = j_1 j_2 \ldots j_r$ *and* $u_{i_1} u_{i_2} \ldots u_{i_l} = v_{j_1} v_{j_2} \ldots v_{j_r}$.

*Proof.* In one direction we prove that $C \parallel Z\mathbb{P}\mathbb{U} \approx C \parallel Z\mathbb{P}\mathbb{V}$ implies $i_1 i_2 \ldots i_l = j_1 j_2 \ldots j_r$ and $u_{i_1} u_{i_2} \ldots u_{i_l} = v_{j_1} v_{j_2} \ldots v_{j_r}$. If $i_1 i_2 \ldots i_l \neq j_1 j_2 \ldots j_r$, then Attacker chooses $C \parallel Z\mathbb{P}\mathbb{U} \xrightarrow{\lambda_S} S \parallel Z\mathbb{P}\mathbb{U}$. Defender cannot invoke the action $Z \xrightarrow{\tau} \epsilon$ for otherwise an $\lambda_Z$ action cannot be performed before an $\lambda_V$ action. The process constant $Z$ is introduced precisely for this blocking effect. Defender's play must be of the form $C \parallel Z\mathbb{P}\mathbb{V} \Longrightarrow C \parallel Q \parallel Z\mathbb{P}\mathbb{V} \xrightarrow{\lambda_S} S \parallel Q \parallel Z\mathbb{P}\mathbb{V} \Longrightarrow S \parallel Q' \parallel Z\mathbb{P}\mathbb{V}$. If $Q'$ can perform any one of $\{\lambda_V, \lambda_Z\} \cup \mathcal{N}$, Attacker wins since $S$ can do none of those. If $Q'$ can do none of those actions, then $S \simeq S \parallel Q'$. By Lemma 5 Attacker has a winning strategy for the weak bisimulation game $(S \parallel Z\mathbb{P}\mathbb{U}, S \parallel Q' \parallel Z\mathbb{P}\mathbb{V})$. If $u_{i_1} u_{i_2} \ldots u_{i_l} \neq v_{j_1} v_{j_2} \ldots v_{j_r}$, the argument is similar.

Conversely we prove that $i_1 i_2 \ldots i_l = j_1 j_2 \ldots j_r \wedge u_{i_1} u_{i_2} \ldots u_{i_l} = v_{j_1} v_{j_2} \ldots v_{j_r}$ implies $C \parallel Z\mathbb{P}\mathbb{U} \simeq C \parallel Z\mathbb{P}\mathbb{V}$. This is done by showing that the relation

$$\left\{ (C \parallel Q \parallel Z\mathbb{P}\mathbb{U}, C \parallel Q \parallel Z\mathbb{P}\mathbb{V}) \;\middle|\; \begin{array}{l} i_1 i_2 \ldots i_l = j_1 j_2 \ldots j_r \\ u_{i_1} u_{i_2} \ldots u_{i_l} = v_{j_1} v_{j_2} \ldots v_{j_r}. \end{array} \right\} \cup \simeq$$

is a branching bisimulation. □

## 4.4 Generating Phase

Suppose that INST has a solution $i_1, i_2, \ldots, i_k$. Fix the following abbreviations: $\mathbb{U}^- = U_{i_2} \ldots U_{i_k}$, $\mathbb{U} = U_{i_1}\mathbb{U}^-$ and $\mathbb{V} = V_{i_1}V_{i_2} \ldots V_{i_k}$. We will argue that Defender has a winning strategy in the branching bisimulation game of $(X, Y)$. Defender's basic idea is to produce the pair $\mathbb{U}, \mathbb{V}$ by forcing. Its strategy and Attacker's counter strategy are described below.

(i) By Defender's Forcing Attacker plays $X \xrightarrow{\lambda_U} D \parallel G_v$. Defender proposes $\mathbb{U}$ via the transitions $Y \xrightarrow{\tau} D \xrightarrow{\tau} D \parallel G_u \Longrightarrow D \parallel G_u\mathbb{U}^- \xrightarrow{\lambda_U} D \parallel G_v\mathbb{U}$. The use of an explicit action $\lambda_U$ guarantees that $\mathbb{U}$ is *nonempty*. Now Attacker has a number of configurations to choose from. But by (1) of Corollary 1, it all boils down to choosing $(D \parallel G_v, D \parallel G_v\mathbb{U})$.

(ii) Due to (1) of Corollary 1 Attacker would not remove $G_v$ using either $G_v \xrightarrow{\tau} \epsilon$ or $G_v \xrightarrow{\lambda_V} Z$. It can generate an element of $\mathcal{V}$ using $G_v$. It can do an action induced by $D$ or a descendant of $D$. Defender simply copycats Attacker's actions. The configuration stays in the form $(D \parallel Q \parallel G_v\mathbb{P}_v, D \parallel Q \parallel G_v\mathbb{P}_v\mathbb{U})$.

(iii) To have any chance to win, Attacker must try the action $\lambda_D$. Defender does the same action. The configuration becomes $(C \parallel Q \parallel G_v\mathbb{P}_v, C \parallel Q \parallel G_v\mathbb{P}_v\mathbb{U})$. At this point if Attacker plays a harmless action, Defender can copycat the action; and the configuration stays in the same shape.

(iv) An important observation is that if Attacker plays $C \parallel Q \parallel G_v\mathbb{P}_v \xrightarrow{\ell} P_1$, Defender can play $C \parallel Q \parallel G_v\mathbb{P}_v\mathbb{U} \Longrightarrow C \parallel Q \Longrightarrow C \parallel Q \parallel G_v\mathbb{P}_v \xrightarrow{\ell} P_1$ and wins. Here $C \parallel Q \simeq C \parallel Q \parallel G_v\mathbb{P}_v$ by (2) of Corollary 1. To see that the assumptions $i_1i_2 \ldots i_l = j_1j_2 \ldots j_r$ and $u_{i_1}u_{i_2} \ldots u_{i_l} = v_{j_1}v_{j_2} \ldots v_{j_r}$ imply $C \parallel Q \parallel G_v\mathbb{P}_v\mathbb{U} \simeq C \parallel Q$, notice that $C \parallel Q \parallel G_v\mathbb{P}_v\mathbb{U} \Longrightarrow C \parallel Q \Longrightarrow C \parallel Q \parallel G_v\mathbb{P}_v\mathbb{V}$ and that $C \parallel Q \parallel G_v\mathbb{P}_v\mathbb{U} \simeq C \parallel Q \parallel G_v\mathbb{P}_v\mathbb{V}$ is a corollary of Proposition 1. Thus Attacker would choose $C \parallel Q \parallel G_v\mathbb{P}_v\mathbb{U}$ to continue.

(v) Attacker would not play $C \parallel Q \parallel G_v\mathbb{P}_v\mathbb{U} \xrightarrow{\tau} C \parallel Q \parallel \mathbb{P}_v\mathbb{U}$ because it would lose right away according to (2) of Corollary 1.

(vi) By Lemma 5 Attacker would not do a $\lambda_I$ action or a $\lambda_S$ action. It stands the best chance to play $C \parallel Q \parallel G_v\mathbb{P}_v\mathbb{U} \xrightarrow{\lambda_V} C \parallel Q \parallel Z\mathbb{P}_v\mathbb{U}$. The counter play from Defender is $C \parallel Q \parallel G_v\mathbb{P}_v \Longrightarrow C \parallel Q \parallel G_v\mathbb{P}_v\mathbb{V} \xrightarrow{\lambda_V} C \parallel Q \parallel Z\mathbb{P}_v\mathbb{V}$.

The last configuration $(C \parallel Q \parallel Z\mathbb{P}_v\mathbb{V}, C \parallel Q \parallel Z\mathbb{P}_v\mathbb{U})$ is optimal for Attacker. By Proposition 1 Defender has a winning strategy for the branching bisimulation game of $(C \parallel Q \parallel Z\mathbb{P}_v\mathbb{V}, C \parallel Q \parallel Z\mathbb{P}_v\mathbb{U})$. Hence the following lemma.

**Lemma 6.** *If INST has a solution then $X \simeq Y$.*

The converse of Lemma 6 also holds. In fact a stronger result is obtainable. In the weak bisimulation game of $(X, Y)$, Attacker has a strategy to force the game to reach a configuration that is essentially of the form $(C \parallel Z\mathbb{P}'_v, C \parallel Z\mathbb{P}_v\mathbb{P}_u)$, where $\mathbb{P}_u \neq \epsilon$. If there is no solution to INST, Proposition 1 implies $C \parallel Z\mathbb{P}'_v \not\approx C \parallel Z\mathbb{P}_v\mathbb{P}_u$. It follows that Attacker has a winning strategy for the weak bisimulation game of $(X, Y)$.

**Lemma 7.** *If INST has no solution then $X \not\approx Y$.*

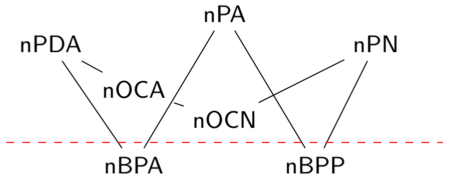

**Fig. 4.** Decidability Border for Branching Bisimilarity on Normed PRS

## 5 Conclusion

Putting together the results derived in this paper, we see that there is a decidability border in the normed PRS hierarchy, see Fig. 4. The branching bisimilarity

1. is undecidable on all normed models above either nBPA or nBPP, and
2. is decidable for both nBPP and nBPA [5,7].

We have confirmed that the first statement is valid for the weak bisimilarity, which slightly strengthens the results obtained in [12]. In fact the statement is valid for every relation between the branching bisimilarity and the weak bisimilarity. It has been conjectured that the second statement is also true for the weak bisimilarity. The answers however have remained a secret for us up to now.

Tighter complexity bounds, or even completeness characterizations, would be very welcome. Another avenue for further study is based on the observation that although the undecidability results of both the present paper and the paper of Jančar and Srba [11] are about the same models, the degrees of undecidability are most likely to be different. In [11] it is pointed out that by constraining the silent actions of nPDA, say to $\epsilon$-popping or $\epsilon$-pushing silent moves, the degree of undecidability of the weak bisimilarity goes from the analytic hierarchy down to the arithmetic hierarchy. It is therefore a reasonable hope that the same restriction may lead to decidable results for the branching bisimilarity on some PRS models. Further studies are called for.

Complete proofs of the results stated in this extended abstract can be found in the full paper [25].

**Acknowledgement.** We gratefully acknowledge the support of the National Science Foundation of China (61033002, ANR 61261130589, 91318301). We thank the anonymous referees and Patrick Totzke for their constructive suggestions.

## References

1. Baeten, J.C.M., Bergstra, J.A., Klop, J.W.: Decidability of Bisimulation Equivalence for Processes Generating Context-free Languages. In: de Bakker, J.W., Nijman, A.J., Treleaven, P.C. (eds.) PARLE 1987. LNCS, vol. 259, pp. 94–111. Springer, Heidelberg (1987)
2. Burkart, O., Caucal, D., Moller, F., Steffen, B.: Verification on Infinite Structures. In: Handbook of Process Algebra. Elsevier Science (2001)
3. Christensen, S., Hirshfeld, Y., Moller, F.: Bisimulation Equivalence is Decidable for Basic Parallel Processes. In: Best, E. (ed.) CONCUR 1993. LNCS, vol. 715, pp. 143–157. Springer, Heidelberg (1993)

4. Christensen, S., Hüttel, H., Stirling, C.: Bisimulation Equivalence is Decidable for all Context-free Processes. In: Cleaveland, W.R. (ed.) CONCUR 1992. LNCS, vol. 630, pp. 138–147. Springer, Heidelberg (1992)
5. Czerwiński, W., Hofman, P., Lasota, S.: Decidability of Branching Bisimulation on Normed Commutative Context-free Processes. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 528–542. Springer, Heidelberg (2011)
6. De Nicola, R., Montanari, U., Vaandrager, F.: Back and Forth Bisimulations. In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR 1990. LNCS, vol. 458, pp. 152–165. Springer, Heidelberg (1990)
7. Fu, Y.: Checking Equality and Regularity for Normed BPA with Silent Moves. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part II. LNCS, vol. 7966, pp. 238–249. Springer, Heidelberg (2013)
8. Hirshfeld, Y., Jerrum, M.: Bisimulation Equivanlence is Decidable for Normed Process Algebra. In: Wiedermann, J., Van Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 412–421. Springer, Heidelberg (1999)
9. Jančar, P.: Undecidability of Bisimilarity for Petri Nets and Some Related Problems. Theoretical Computer Science 148, 281–301 (1995)
10. Jančar, P., Brics, J.S.: Highly Undecidable Questions for Process Algebras. In: Levy, J.-J., Mayr, E.W., Mitchell, J.C. (eds.) TCS 2004. IFIP, vol. 155, pp. 507–520. Springer, Boston (2004)
11. Jančar, P., Srba, J.: Undecidability of Bisimilarity by Defender's Forcing. Journal of the ACM 55, 1–26 (2008)
12. Kučera, A., Jančar, P.: Equivalence-Checking on Infinite-State Systems: Techniques and Results. Theory and Practice of Logic Programming 6, 227–264 (2006)
13. Mayr, R.: Process Rewrite Systems. Information and Computation 156, 264–286 (2000)
14. Mayr, R.: Undecidability of Weak Bisimulation Equivalence for 1-Counter Processes. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 570–583. Springer, Heidelberg (2003)
15. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
16. Park, D.: Concurrency and Automata on Infinite Sequences. In: Deussen, P. (ed.) GI-TCS 1981. LNCS, vol. 104, pp. 167–183. Springer, Heidelberg (1981)
17. Srba, J.: Undecidability of Weak Bisimilarity for Pushdown Processes. In: Brim, L., Jančar, P., Křetínský, M., Kučera, A. (eds.) CONCUR 2002. LNCS, vol. 2421, pp. 579–594. Springer, Heidelberg (2002)
18. Srba, J.: Roadmap of Infinite Results. EATCS 78, 163–175 (2002)
19. Srba, J.: Undecidability of Weak Bisimilarity for PA-Processes. In: Ito, M., Toyama, M. (eds.) DLT 2002. LNCS, vol. 2450, pp. 197–208. Springer, Heidelberg (2003)
20. Srba, J.: Completeness Results for Undecidable Bisimilarity Problems. Electronic Notes in Computer Science 98, 5–19 (2004)
21. Stirling, C.: Decidability of Bisimulation Equivalence for Normed Pushdown Processes. Theoretical Computer Science 195, 113–131 (1998)
22. Stirling, C.: The Joys of Bisimulation. In: Brim, L., Gruska, J., Zlatuška, J. (eds.) MFCS 1998. LNCS, vol. 1450, pp. 142–151. Springer, Heidelberg (1998)
23. Stirling, C.: Decidability of Weak Bisimilarity for a Subset of Basic Parallel Processes. In: Honsell, F., Miculan, M. (eds.) FOSSACS 2001. LNCS, vol. 2030, pp. 379–393. Springer, Heidelberg (2001)
24. van Glabbeek, R., Weijland, W.: Branching Time and Abstraction in Bisimulation Semantics. Journal of ACM 43, 555–600 (1996)
25. Yin, Q., Fu, Y., He, C., Huang, M., Tao, X.: Branching Bisimilarity Checking for PRS (2014), `http://arxiv.org/abs/1402.0050`