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 Process) and nBPA (normed Basic Process Algebra). These results lead to the question if there are any other models in the hierarchy of PRS (Process Rewrite System) whose branching bisimilarity is decidable. It is shown in this paper that the branching bisimilarity for both nOCN (normed One Counter Net) and nPA (normed Process Algebra) 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 and Milner [16,15]. 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 automaton, process algebra, Petri net and their restricted and extended variations. A beautiful classification of the models in terms of PRS (Process Rewrite System) 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 Net). 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 clearer. The weak bisimilarity is undecidable for every model above either nBPA (normed Basic Process Algebra) or nBPP (normed Basic Parallel Process). Srba [17] showed that weak bisimilarity on nPDA (normed Pushdown Automaton) is undecidable by a reduction from the halting problem of Minsky Machine. The undecidability was soon extended to nOCN (normed One Counter Net), a submodel of both nPDA and nPN, by Mayr [14]. Srba also showed that the weak bisimilarity on PA (Process Algebra)

	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 on Process Rewriting System

is undecidable [19] via a reduction from Post's Correspondence Problem. Later several highly undecidable results was 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. 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 question: Is there any other model in the PRS hierarchy whose branching bisimilarity is decidable? The purpose of this paper is to answer the question. Our contributions are as follows:

- We establish the fact that the branching bisimilarity on nOCN is undecidable. This is an improvement of Mayr's result about the undecidability of the weak bisimilarity on nOCN [14]. We also prove that the branching bisimilarity of nPA is undecidable. This is a significant strengthening of Srba's result [19] about the undecidability of the weak bisimilarity on PA. These new results together with the previous results 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 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 proves the result for nOCN and demonstrates Defender's Forcing technique for branching bisimulation game. Section 4 establishes the 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 ℓ, and Δ is a finite set of transition rules. The *processes* defined by $\mathcal{P}$ are generated by the following grammar:

$$P \ ::= \ \epsilon \ \mid \ X \ \mid \ PP' \ \mid \ P \parallel P'.$$

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

$$\frac{X \stackrel{\ell}{\longrightarrow} P \in \Delta}{X \stackrel{\ell}{\longrightarrow} P} \qquad \frac{P \stackrel{\ell}{\longrightarrow} P'}{PQ \stackrel{\ell}{\longrightarrow} P'Q} \qquad \frac{P \stackrel{\ell}{\longrightarrow} P'}{P \parallel Q \stackrel{\ell}{\longrightarrow} P' \parallel Q} \qquad \frac{Q \stackrel{\ell}{\longrightarrow} Q'}{P \parallel Q \stackrel{\ell}{\longrightarrow} P \parallel Q'}$$

The operational semantics is structural, meaning that $PQ \stackrel{\ell}{\longrightarrow} P'Q$, $P \parallel Q \stackrel{\ell}{\longrightarrow} P' \parallel Q$ and $Q \parallel P \stackrel{\ell}{\longrightarrow} Q \parallel P'$ whenever $P \stackrel{\ell}{\longrightarrow} P'$. We write $\Longrightarrow$ for the reflexive transitive closure of $\stackrel{\tau}{\longrightarrow}$, and $\stackrel{\widehat{\ell}}{\Longrightarrow}$ for $\Longrightarrow\stackrel{\ell}{\longrightarrow}\Longrightarrow$ 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 Δ 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 Δ is of the form $pX^i \stackrel{\ell}{\longrightarrow} qX^j$ with $i < 2$. The semantics is structural in the sense that $pX^{i+k} \stackrel{\ell}{\longrightarrow} qX^{j+k}$ whenever $pX^i \stackrel{\ell}{\longrightarrow} qX^j$.

A process P defined in $\mathcal{P}$, respectively $\mathcal{M}$, is *normed* if $\exists \ell_1, \ldots, \ell_n. P \stackrel{\ell_1}{\longrightarrow} \ldots \stackrel{\ell_n}{\longrightarrow} \epsilon$, respectively $\exists \ell_1, \ldots, \ell_n, p. P \stackrel{\ell_1}{\longrightarrow} \ldots \stackrel{\ell_n}{\longrightarrow} p$. We say that $\mathcal{P}/\mathcal{M}$ is normed if all processes defined in it are normed. We write (n)PA for the (normed) process algebra model and (n)OCN for the (normed) one counter net model.

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* $P \stackrel{\ell}{\longrightarrow} P'$*, then* $Q \stackrel{\widehat{\ell}}{\Longrightarrow} Q'$ *and* $P'\mathcal{R}Q'$ *for some* Q'*.*

2. Whenever $P\mathcal{R}Q \stackrel{\ell}{\longrightarrow} Q'$*, then* $P \stackrel{\widehat{\ell}}{\Longrightarrow} P'\mathcal{R}Q'$ *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* $P \stackrel{\ell}{\longrightarrow} P'$*, then either (i)* $Q \Longrightarrow Q'' \stackrel{\ell}{\longrightarrow} Q'$ *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 $P\mathcal{R}Q \stackrel{\ell}{\longrightarrow} Q'$*, then either (i)* $P \Longrightarrow P'' \stackrel{\ell}{\longrightarrow} P'\mathcal{R}Q'$ *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.*

Both $\simeq$ and $\approx$ are congruence relation for our models. 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 $\cong$ be a process equivalence. A silent action $P \xrightarrow{\tau} P'$ is *state preserving* with regards to $\cong$, notation $P \rightarrow P'$, if $P' \cong P$; it is *change-of-state* with regards to $\cong$, notation $P \xrightarrow{\iota} P'$, if $P' \ncong P$. The reflexive and transitive closure of $\rightarrow$ is denoted by $\rightarrow^*$. 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 $Q \not\simeq Q'$.

Bisimilarity has a game theoretic characterization known as *bisimulation game* [22]. Suppose that a pair of processes P, Q, called a *configuration*, are defined in say a process algebra $(\mathcal{C}, \mathcal{A}, \Delta)$. 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:

1. Suppose (P_0, P_1) is the current configuration. 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 $P_{1-i} \stackrel{\widehat{\ell}}{\Longrightarrow} P_{1-i}'$. 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.

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 bound 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 process. The process G defined by the rules $G \stackrel{\tau}{\longrightarrow} GX$ and $GX \stackrel{\tau}{\longrightarrow} G$ is *generating* due to the fact that every process that G may evolve into, say GX^n, is branching bisimilar to G. Add additional 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 $1 : I_1;\ 2 : I_2;\ \ldots;\ m{-}1 : I_{m-1};\ m : \text{halt}$, where for each $i \in \{1, \ldots, m-1\}$ the instruction I_i is in either of the following forms, assuming $1 \leq j, k \leq m$ 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 $(p_j X^{3n}, p'_j X^{3n})$ has a winning strategy if and only if Defender of the branching bisimulation game of $(p_i X^n, p'_i X^n)$ has a winning strategy.*

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

$p_i \xrightarrow{\tau} G'$	$p'_i \xrightarrow{\tau} G'$
$p_i \xrightarrow{a} q_1$	$G' \xrightarrow{a} q'_1,\ G' \xrightarrow{\tau} G'X,\ G'X \xrightarrow{\tau} G'$
$q_1 \xrightarrow{a} q_2$	$q'_1 \xrightarrow{a} q'_2$
$q_1 \xrightarrow{t} t(3)$	$q'_1 \xrightarrow{t} t(1)$
$q_2 \xrightarrow{\tau} G$	$q'_2 \xrightarrow{\tau} G$
$G \xrightarrow{\tau} GX,\ GX \xrightarrow{\tau} G,\ G \xrightarrow{a} q_3$	$q'_2 \xrightarrow{a} q'_3$
$q_3 \xrightarrow{a} p_j$	$q'_3 \xrightarrow{a} p'_j$
$q_3 \xrightarrow{t} t(1)$	$q'_3 \xrightarrow{t} t(1)$
$t(3)X \xrightarrow{c} t_2X,\ t_2X \xrightarrow{c} t_1X,\ t_1X \xrightarrow{c} t(3)$	$t(1)X \xrightarrow{c} t(1)$

Fig. 2. Multiplication Operation on Counter in OCN

do not automatically create a Defender's Forcing situation. The reason is that although $p'_iX^n \rightarrow G'X^n$, 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 $p'_iX^n \rightarrow G'X^n \rightarrow^* G'X^{3n} \xrightarrow{a} q'_1X^{3n}$, 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 $t(3)X^n \simeq t(1)X^{3n}$. 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 $t(1)X^{3n} \simeq t(1)X^{3n}$. 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. *Branching bisimilarity is undecidable on* nOCN.

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. So we can construct a reduction witnessing that "$\mathcal{M}$ halts iff $p_1X \not\simeq p'_1X$". □

4 Undecidability of nPA

The main undecidability result of the paper is proved by reducing PCP (Post's Correspondence Problem) to the branching bisimilarity checking problem on nPA. Suppose Σ is a finite set of symbols and Σ^+ is the set of nonempty finite strings over Σ. The size of Σ 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.

The main theorem of the paper follows immediately from (1).

Theorem 2. *Both* $\simeq$ *and* $\approx$ *are undecidable on* nPA.

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 Σ, 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 ω. 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 $\mathbb{P}_u$ is empty, $\mathbb{P}_u$ is understood to denote ϵ.

$X \xrightarrow{\lambda_U} D \parallel G_v,\ X \xrightarrow{\tau} D;\quad Y \xrightarrow{\tau} D;\quad D \xrightarrow{\tau} D \parallel G_u,\ D \xrightarrow{\lambda_D} C;$ $G_u \xrightarrow{\tau} G_u U_i,\ G_u \xrightarrow{\lambda_U} G_v U_i;\quad G_u \xrightarrow{\tau} G'_v,\ G'_v \xrightarrow{\tau} G'_v V_i,\ G'_v \xrightarrow{\tau} Z;$
$G_v \xrightarrow{\tau} G_v V_i,\ G_v \xrightarrow{\tau} \epsilon,\ G_v \xrightarrow{\lambda_V} Z;\quad Z \xrightarrow{\tau} \epsilon,\ Z \xrightarrow{\lambda_Z} \epsilon;$
$C \xrightarrow{\lambda_I} I,\ C \xrightarrow{\lambda_S} S,\ C \xrightarrow{\tau} C \parallel G,\ C \xrightarrow{\tau} C \parallel G_v;$ $G \xrightarrow{\tau} GU_i,\ G \xrightarrow{\tau} GV_i,\ G \xrightarrow{\tau} \epsilon;$
$I \xrightarrow{\lambda_I} C',\ I \xrightarrow{i} I;\quad S \xrightarrow{\lambda_S} C',\ S \xrightarrow{a} S;\quad C' \xrightarrow{\tau} C' \parallel G',\ C' \xrightarrow{\tau} \epsilon;$ $G' \xrightarrow{\tau} G'U_i,\ G' \xrightarrow{\tau} G'V_i,\ G' \xrightarrow{\tau} G'W,\ G' \xrightarrow{\tau} G_v,\ G' \xrightarrow{\tau} Z;$
$U_i \xrightarrow{\tau} W(u_i, i),\ V_i \xrightarrow{\tau} W(v_i, i);$ $W(a\omega, i) \xrightarrow{a} W(\omega, i),\ W(a\omega, 0) \xrightarrow{a} W(\omega, 0),\ W(\omega, i) \xrightarrow{i} W(\omega, 0),$ $W(a\omega, i) \xrightarrow{\tau} W(\omega, i),\ W(a\omega, 0) \xrightarrow{\tau} W(\omega, 0),\ W(\omega, i) \xrightarrow{\tau} W(\omega, 0),\ W(\epsilon, 0) \xrightarrow{\tau} \epsilon.$
In the above rules, i ranges over $\{1, \ldots, n\}$, a ranges over Σ, 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 throw away what have been produced. The process D for instance can induce circular silent transition sequence of the form

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

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 $P \Longrightarrow P \parallel Q$. Then $P \parallel Q \Longrightarrow P$.*

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

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

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 Σ 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. $I \parallel B\mathbb{PU} \approxeq I \parallel B\mathbb{PV}$ *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. $S \parallel B\mathbb{PU} \approxeq S \parallel B\mathbb{PV}$ *if and only if* $i_1 i_2 \ldots i_l = j_1 j_2 \ldots j_r$.

Proof. Suppose $I \parallel B\mathbb{PU} \simeq I \parallel B\mathbb{PV}$ 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 $I \parallel B\mathbb{PU}$ to $I \parallel \mathbb{U}$ must be simulated essentially by an action sequence from $I \parallel B\mathbb{PV}$ 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* $\mathbb{P}$, $C \parallel Z\mathbb{PU} \approxeq C \parallel Z\mathbb{PV}$ *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 $C \parallel Z\mathbb{PU} \approx C \parallel Z\mathbb{PV}$ 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 $C \parallel Z\mathbb{PU} \xrightarrow{\lambda_S} S \parallel Z\mathbb{PU}$. Defender cannot invoke the action $Z \xrightarrow{\tau} \epsilon$ for otherwise an λ_Z action cannot be performed before an λ_V action. The process constant Z is introduced precisely for this blocking effect. Defender's play must be of the form $C \parallel Z\mathbb{PV} \Longrightarrow C \parallel Q \parallel Z\mathbb{PV} \xrightarrow{\lambda_S} S \parallel Q \parallel Z\mathbb{PV} \Longrightarrow S \parallel Q' \parallel Z\mathbb{PV}$. 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 $(S \parallel Z\mathbb{PU}, S \parallel Q' \parallel Z\mathbb{PV})$. 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 $C \parallel Z\mathbb{PU} \simeq C \parallel Z\mathbb{PV}$. This is done by showing that the relation

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

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: $\mathbb{U}^{-1} = U_{i_2} \ldots U_{i_k}$, $\mathbb{U} = U_{i_1}\mathbb{U}^{-1}$ 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 as follows:

(i) By Defender's Forcing Attacker plays $X \xrightarrow{\lambda_U} D \,\|\, G_v$. Defender proposes $\mathbb{U}$ via the transitions $Y \xrightarrow{\tau} D \xrightarrow{\tau} D \,\|\, G_u \Longrightarrow D \,\|\, G_u\mathbb{U}^{-1} \xrightarrow{\lambda_U} D \,\|\, G_v\mathbb{U}$. The use of an explicit action λ_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 $(D \,\|\, G_v, D \,\|\, G_v\mathbb{U})$.

(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 $(D \,\|\, Q \,\|\, G_v\mathbb{P}_v, D \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U})$.

(iii) To have any chance to win, Attacker must try the action λ_D. Defender does the same action. The configuration becomes $(C \,\|\, Q \,\|\, G_v\mathbb{P}_v, C \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U})$. 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 $C \,\|\, Q \,\|\, G_v\mathbb{P}_v \xrightarrow{\ell} P_1$, Defender can play $C \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U} \Longrightarrow C \,\|\, Q \Longrightarrow C \,\|\, Q \,\|\, G_v\mathbb{P}_v \xrightarrow{\ell} P_1$ and wins. Here $C \,\|\, Q \simeq C \,\|\, Q \,\|\, G_v\mathbb{P}_v$ 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 $C \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U} \simeq C \,\|\, Q$, notice that $C \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U} \Longrightarrow C \,\|\, Q \Longrightarrow C \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{V}$ and that $C \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U} \simeq C \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{V}$ is a corollary of Proposition 1. Thus Attacker would choose $C \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U}$ to continue.

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

(vi) Attacker could choose to do a λ_I action or a λ_S action. But it stands the best chance to play $C \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U} \xrightarrow{\lambda_V} C \,\|\, Q \,\|\, Z\mathbb{P}_v\mathbb{U}$. The counter play from Defender is $C \,\|\, Q \,\|\, G_v\mathbb{P}_v \Longrightarrow C \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{V} \xrightarrow{\lambda_V} C \,\|\, Q \,\|\, Z\mathbb{P}_v\mathbb{V}$.

The last configuration $(C \,\|\, Q \,\|\, Z\mathbb{P}_v\mathbb{V}, C \,\|\, Q \,\|\, Z\mathbb{P}_v\mathbb{U})$ is optimal for Attacker. By Proposition 1 Defender has a winning strategy for the branching bisimulation game of $(C \,\|\, Q \,\|\, Z\mathbb{P}_v\mathbb{V}, C \,\|\, Q \,\|\, Z\mathbb{P}_v\mathbb{U})$. 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 available. 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 $(C \,\|\, Z\mathbb{P}'_v, C \,\|\, Z\mathbb{P}_v\mathbb{P}_u)$, where $\mathbb{P}_u \neq \epsilon$. If there is no solution to INST, Proposition 1 implies $C \,\|\, Z\mathbb{P}'_v \not\approx C \,\|\, Z\mathbb{P}_v\mathbb{P}_u$. It follows that Attacker has a winning strategy for the weak bisimulation game of (X, Y).

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

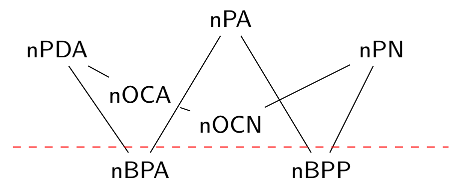

Fig. 4. Decidability Border for Branching Bisimilarity on Normed PRS

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 models above either nBPA or nBPP, and
2. is decidable for both nBPP and nBPA [5,7].

For the weak bisimilarity we have confirmed that the first statement is valid, which slightly strengthens the results obtained in [12]. It has been conjectured that the second statement is also true for the weak bisimialrity. The answers however have remained a secret for us up to now. The picture for the decidability of the branching bisimilarity on the unnormed PRS is similar.

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 ϵ-popping or ϵ-pushing silent moves, the degree of undecidability of the weak bisimilarity goes from the analytic hierarchy 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.

Acknowledgement. The support from NSFC (61033002, ANR 61261130589, 91318301) 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. O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on Infinite Structures. *Handbook of Process Algebra*. Elsevier Science, 2001.
3. S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation Equivalence is Decidable for Basic Parallel Processes. *CONCUR 1993*, Lecture Notes in Computer Science 715, 143–157, 1993.

4. S. Christensen, H. Hüttel, and C. Stirling. Bisimulation Equivalence is Decidable for all Context-free Processes. *CONCUR 1992*, Lecture Notes in Computer Science 630, 138–147, 1992.
5. W. Czerwiński, P. Hofman, and S. Lasota. Decidability of Branching Bisimulation on Normed Commutative Context-free Processes. *CONCUR 2011*, Lecture Notes in Computer Science 6901, 528–542, 2011.
6. R. De Nicola, U. Montanari, and F. Vaandrager. Back and Forth Bisimulations. *CONCUR 1990*, Lecture Notes in Computer Science 458, 152–165, 1990.
7. Y. Fu. Checking Equality and Regularity for Normed BPA with Silent Moves. *ICALP 2013*, Lecture Notes in Computer Science 7966, 238–249, 2013.
8. Y. Hirshfeld and M. Jerrum. Bisimulation Equivanlence is Decidable for Normed Process Algebra. *ICALP 1999*, Lecture Notes in Computer Science 1644, 412–421, 1999.
9. P. Jančar. Undecidability of Bisimilarity for Petri Nets and Some Related Problems. *Theoretical Computer Science*, 148:281–301, 1995.
10. P. Jancar and J. Srba. Highly Undecidable Questions for Process Algebras. *TCS 2004*, IFIP International Federation for Information Processing 155, 507–520, 2004.
11. P. Jančar and J. Srba. Undecidability of Bisimilarity by Defender's Forcing. *Journal of the ACM*, 55:1–26, 2008.
12. A. Kučera and P. Jančar. Equivalence-Checking on Infinite-State Systems: Techniques and Results. *Theory and Practice of Logic Programming*, 6:227–264, 2006.
13. R. Mayr. Process Rewrite Systems. *Information and Computation*, 156:264–286, 2000.
14. R. Mayr. Undecidability of Weak Bisimulation Equivalence for 1-Counter Processes. *ICALP 2003*, Lecture Notes in Computer Science 2719, 570–583, 2003.
15. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
16. D. Park. Concurrency and Automata on Infinite Sequences. *Theoretical Computer Science*, Lecture Notes in Computer Science 104, 167–183, 1981.
17. J. Srba. Undecidability of Weak Bisimilarity for Pushdown Processes. *CONCUR 2002*, Lecture Notes in Computer Science 201, 579–594, 2002.
18. J. Srba. Roadmap of Infinite Results. *EATCS*, 78:163–175, 2002.
19. J. Srba. Undecidability of Weak Bisimilarity for PA-Processes. *Developments in Language Theory*, Lecture Notes in Computer Science 2450, 197–209, 2003.
20. J. Srba. Completeness Results for Undecidable Bisimilarity Problems. Electronic Notes in Computer Science, 98:5–19, 2004.
21. C. Stirling. Decidability of Bisimulation Equivalence for Normed Pushdown Processes. *Theoretical Computer Science*, 195:113–131, 1998.
22. C. Stirling. The Joys of Bisimulation. *Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science 1450, 142–151, 1998.
23. C. Stirling. Decidability of Weak Bisimilarity for a Subset of Basic Parallel Processes. *Foundations of Software Science and Computation Structure*, Lecture Notes in Computer Science 2030, 379–393, 2001.
24. R. van Glabbeek and W. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of ACM*, 43:555–600, 1996.

A Proof of Corollary 1

The proof is a simple application of Lemma 1. For the constant D one has the following circular silent transition sequence:

$$\begin{aligned}
D &\xrightarrow{\tau} D \,\|\, G_u \\
&\Longrightarrow D \,\|\, G_u \mathbb{P}_u \\
&\xrightarrow{\tau} D \,\|\, G'_v \mathbb{P}_u \\
&\Longrightarrow D \,\|\, G'_v \mathbb{P}_v \mathbb{P}_u \\
&\xrightarrow{\tau} D \,\|\, Z \mathbb{P}_v \mathbb{P}_u \\
&\xrightarrow{\tau} D \,\|\, \mathbb{P}_v \mathbb{P}_u \\
&\xrightarrow{\tau} D \,\|\, W \mathbb{P}_v \mathbb{P}_u \\
&\Longrightarrow D \,\|\, \mathbb{P}_u \\
&\Longrightarrow D \,\|\, W \mathbb{P}_u \\
&\Longrightarrow D.
\end{aligned}$$

For the constant C one has

$$\begin{aligned}
C &\xrightarrow{\tau} C \,\|\, G \\
&\Longrightarrow C \,\|\, G \mathbb{P} \\
&\Longrightarrow C \,\|\, \mathbb{P} \\
&\Longrightarrow C \,\|\, W \mathbb{P} \\
&\Longrightarrow C \\
&\xrightarrow{\tau} C \,\|\, G_v \\
&\Longrightarrow C \,\|\, G_v \mathbb{P}_v \\
&\xrightarrow{\tau} C \,\|\, \mathbb{P}_v \\
&\Longrightarrow C.
\end{aligned}$$

Finally for the constant C' one has

$$\begin{aligned}
C' &\xrightarrow{\tau} C' \,\|\, G' \\
&\Longrightarrow C' \,\|\, G' \mathbb{Q} \\
&\xrightarrow{\tau} C' \,\|\, Z \mathbb{Q} \\
&\Longrightarrow C' \\
&\Longrightarrow C' \,\|\, G' \mathbb{Q} \\
&\xrightarrow{\tau} C' \,\|\, G_v \mathbb{Q} \\
&\Longrightarrow C'.
\end{aligned}$$

We are done.

B Proof of Lemma 5

Suppose $\mathbb{U} = U_{i_1}U_{i_2}\dots U_{i_l}$ and $\mathbb{V} = V_{j_1}V_{j_2}\dots V_{j_r}$. We show that

(i) If $u_{i_1}u_{i_2}\dots u_{i_l} = v_{j_1}v_{j_2}\dots v_{j_r}$ then $I \parallel G_v\mathbb{P}\mathbb{U} \simeq I \parallel G_v\mathbb{P}\mathbb{V}$.
(ii) If $i_1i_2\dots i_l = j_1j_2\dots j_r$ then $S \parallel G_v\mathbb{P}\mathbb{U} \simeq S \parallel G_v\mathbb{P}\mathbb{V}$.
(iii) If $I \parallel G_v\mathbb{P}\mathbb{U} \approx I \parallel G_v\mathbb{P}\mathbb{V}$ then $u_{i_1}u_{i_2}\dots u_{i_l} = v_{j_1}v_{j_2}\dots v_{j_r}$.
(iv) If $S \parallel G_v\mathbb{P}\mathbb{U} \approx S \parallel G_v\mathbb{P}\mathbb{V}$ then $i_1i_2\dots i_l = j_1j_2\dots j_r$.

Proof. (ii) Suppose $i_1i_2\dots i_l = j_1j_2\dots j_r$. The proof is given by the following case analysis:

- If Attacker chooses the transition $S \parallel G_v\mathbb{P}\mathbb{U} \xrightarrow{\lambda_S} C' \parallel G_v\mathbb{P}\mathbb{U}$, Defender can win by playing $S \parallel G_v\mathbb{P}\mathbb{V} \xrightarrow{\lambda_S} C' \parallel G_v\mathbb{P}\mathbb{V}$. This is because $C' \parallel G_v\mathbb{P}\mathbb{U} \simeq C' \simeq C' \parallel G_v\mathbb{P}\mathbb{V}$ by (3) of Corollary 1.
- If Attacker plays a transition caused by an action of G_v, Defender does the same action. Suppose the resulting configuration is $(S \parallel Z\mathbb{P}\mathbb{U}, S \parallel Z\mathbb{P}\mathbb{V})$. Attacker would not play a λ_S action for the same reason. If it plays an action caused by Z, Defender follows suit.
- If both Attacker and Defender play in the optimal manner, the game will reach the configuration $(S \parallel \mathbb{U}, S \parallel \mathbb{V})$. By (3) of Corollary 1 Attacker would lose if it plays a λ_S action. It would not win if it plays an action from Σ. Finally if Attacker decides to play say i_1 or skip it, Defender copycats the action. By the assumption $i_1i_2\dots i_l = j_1j_2\dots j_r$, Attacker would not win in this case either.

This completes the proof of (ii).

(iv) Suppose that $S \parallel G_v\mathbb{P}\mathbb{U} \approx S \parallel G_v\mathbb{P}\mathbb{V}$ and without loss of generality that

$$|i_1i_2\dots i_l| \geq |j_1j_2\dots j_r|. \tag{2}$$

Now $S \parallel G_v\mathbb{P}\mathbb{V} \xrightarrow{\lambda_V} S \parallel Z\mathbb{P}\mathbb{V}$ must be bisimulated by $S \parallel G_v\mathbb{P}\mathbb{U} \stackrel{\lambda_V}{\Longrightarrow} S \parallel Z\mathbb{P}'\mathbb{P}\mathbb{U}$ for some $\mathbb{P}'$. Notice that if $\mathbb{P}'$ is not empty, there would be no hope that $S \parallel Z\mathbb{P}'\mathbb{P}\mathbb{U} \approx S \parallel Z\mathbb{P}\mathbb{V}$. So the simulation must be of the form $S \parallel G_v\mathbb{P}\mathbb{U} \xrightarrow{\lambda_V} S \parallel Z\mathbb{P}\mathbb{U}$. Let

$$S \parallel Z\mathbb{P}\mathbb{U} \xrightarrow{\lambda_Z} S \parallel \mathbb{P}\mathbb{U} \stackrel{k_1}{\Longrightarrow} \dots \stackrel{k_m}{\Longrightarrow} S \parallel \mathbb{U}$$

be the longest sequence of actions of the form $\lambda_Z, k_1, \dots, k_m$ such that $k_1, \dots, k_m \in \mathcal{N}$. In the light of (2) the simulation from $\mathbb{P}\mathbb{V}$ must be of the form

$$S \parallel Z\mathbb{P}\mathbb{V} \xrightarrow{\lambda_Z} S \parallel \mathbb{P}\mathbb{V} \stackrel{k_1}{\Longrightarrow} \dots \stackrel{k_m}{\Longrightarrow} S \parallel \mathbb{V}.$$

By similar argument one shows that $S \parallel \mathbb{U} \approx S \parallel \mathbb{V}$ implies $i_1i_2\dots i_l = j_1j_2\dots j_r$.

The proof of (i) and (iii) can be done in the same fashion. □

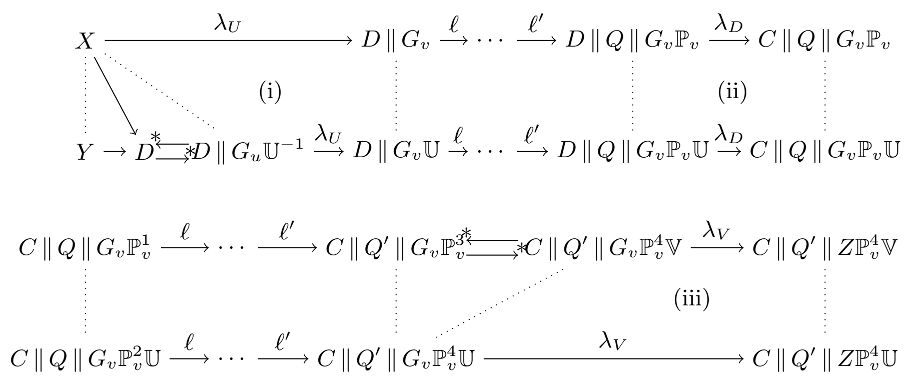

Fig. 5. Defender's Strategy

C Proof of Proposition 1

Suppose $\mathbb{U} = U_{i_1}U_{i_2}\ldots U_{i_l}$ and $\mathbb{V} = V_{j_1}V_{j_2}\ldots V_{j_r}$. We have seen that $i_1i_2\ldots i_l = j_1j_2\ldots j_r$ and $u_{i_1}u_{i_2}\ldots u_{i_l} = v_{j_1}v_{j_2}\ldots v_{j_r}$ imply $C \parallel Z\mathbb{P}\mathbb{U} \simeq C \parallel Z\mathbb{P}\mathbb{V}$ for all $\mathbb{P}$. The following lemma says that if there is some $\mathbb{P}$ such that $C \parallel Z\mathbb{P}\mathbb{U} \simeq C \parallel Z\mathbb{P}\mathbb{V}$, then $i_1i_2\ldots i_l = j_1j_2\ldots j_r$ and $u_{i_1}u_{i_2}\ldots u_{i_l} = v_{j_1}v_{j_2}\ldots v_{j_r}$.

Lemma 8. *If* $C \parallel Z\mathbb{P}_v^1 \approx C \parallel Z\mathbb{P}_v^2\mathbb{P}_u^2$ *for some* $\mathbb{P}_v^1, \mathbb{P}_v^2, \mathbb{P}_u^2$ *with* $\mathbb{P}_u^2 \neq \epsilon$*, then* INST *has a solution.*

Proof. Suppose $C \parallel Z\mathbb{P}'_v \xrightarrow{\lambda_I} I \parallel Z\mathbb{P}'_v$ is simulated by

$$C \parallel Z\mathbb{P}_v^2\mathbb{P}_u^2 \Longrightarrow C \parallel Q \parallel Z\mathbb{P}_v^2\mathbb{P}_u^2 \xrightarrow{\lambda_I} I \parallel Q \parallel Z\mathbb{P}_v^2\mathbb{P}_u^2 \Longrightarrow I \parallel Q' \parallel Z\mathbb{P}_v^2\mathbb{P}_u^2 \approx I \parallel Z\mathbb{P}_v^1$$

for some Q, Q'. It is easy to see that Q' contains neither G nor G_v. Moreover Q' contains no processes of the form $W(\omega, i)$ for $\omega \neq \epsilon$. The only nontrivial components Q' may contain are processes of the form $W(\epsilon, i)$. It follows that $I \parallel Q' \approx I$. Consequently $I \parallel Z\mathbb{P}_v^1 \approx I \parallel Z\mathbb{P}_v^2\mathbb{P}_u^2$. Using similar argument one derives that $S \parallel Z\mathbb{P}_v^1 \approx S \parallel Z\mathbb{P}_v^2\mathbb{P}_u^2$. We are done by applying Lemma 5. □

D Proof of Lemma 6

Defender's strategy is composed of three substrategies (see Fig. 5). We now give the details of the substrategies.

(i) By Defender's Forcing with delayed justification, Attacker chooses to play $X \xrightarrow{\lambda_U} D \parallel G_v$. Defender responds with the following transition sequence

$$Y \xrightarrow{\tau} D \Longrightarrow^* D \parallel G_u\mathbb{U}^{-1} \xrightarrow{\lambda_U} D \parallel G_v\mathbb{U},$$

noticing that $Y \simeq D \simeq D \,\|\, G_u\mathbb{U}^{-1}$ according to (1) of Corollary 1. The following case analysis implies that if Attacker plays optimal, it would continue from the configuration $(D \,\|\, G_v, D \,\|\, G_v\mathbb{U})$.

(a) If Attacker sets the configuration to be $(D \,\|\, G_v, D \,\|\, G_v\mathbb{U})$, we are done.

(b) Otherwise assume w.l.o.g. that Attacker sets it to be $(X, D \,\|\, G_v\mathbb{U}^{-1})$.
   - By Defender's Forcing, Attacker would not play $D \,\|\, G_v\mathbb{U}^{-1} \xrightarrow{\ell} Q$ since it can be matched by $X \xrightarrow{\tau} D \Longrightarrow D \,\|\, G_v\mathbb{U}^{-1} \xrightarrow{\ell} Q$.
   - Attacker would not play $X \xrightarrow{\tau} D$ either since Defender can win by palying $D \,\|\, G_u\mathbb{U}^{-1} \Longrightarrow D$.
   - If Attacker plays $X \xrightarrow{\lambda_U} D \,\|\, G_v$, Defender responds with the transition $D \,\|\, G_u\mathbb{U}^{-1} \xrightarrow{\lambda_U} D \,\|\, G_v\mathbb{U}$.

(ii) Now suppose the current configuration is $(D \,\|\, Q \,\|\, G_v\mathbb{P}_v, D \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U})$. Attacker would choose neither $G_v \xrightarrow{\tau} \epsilon$ nor $G_v \xrightarrow{\lambda_V} Z$ since $D \,\|\, Q \,\|\, \mathbb{P}_v \simeq D \,\|\, Q \,\|\, \mathbb{P}_v\mathbb{U}$ and $D \,\|\, Q \,\|\, Z\mathbb{P}_v \simeq D \,\|\, Q \,\|\, Z\mathbb{P}_v\mathbb{U}$ by (1) of Corollary 1. The other cases are as follows:
   - Attacker plays $D \,\|\, Q \,\|\, G_v\mathbb{P}_v \xrightarrow{\ell} D \,\|\, Q' \,\|\, G_v\mathbb{P}_v$. Defender responds by playing $D \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U} \xrightarrow{\ell} D \,\|\, Q' \,\|\, G_v\mathbb{P}_v\mathbb{U}$.
   - Attacker plays $D \,\|\, Q \,\|\, G_v\mathbb{P}_v \xrightarrow{\tau} D \,\|\, G_v \,\|\, Q \,\|\, G_v\mathbb{P}_v$. Defender responds by playing $D \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U} \xrightarrow{\tau} D \,\|\, G_v \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U}$.
   - Attacker plays $D \,\|\, Q \,\|\, G_v\mathbb{P}_v \xrightarrow{\tau} D \,\|\, Q \,\|\, G_vV_i\mathbb{P}_v$. Defender responds by playing $D \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U} \xrightarrow{\tau} D \,\|\, Q \,\|\, G_vV_i\mathbb{P}_v\mathbb{U}$.
   - Attacker plays $D \,\|\, Q \,\|\, G_v\mathbb{P}_v \xrightarrow{\lambda_D} C \,\|\, Q \,\|\, G_v\mathbb{P}_v$. Defender counter plays $D \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U} \xrightarrow{\lambda_D} C \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U}$. In this case Attacker can only choose $(C \,\|\, Q \,\|\, G_v\mathbb{P}_v, C \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U})$ as the next configuration.

   In these cases Attacker will eventually choose to play an λ_D action to have any chance to win at all.
   If Attacker chooses $D \,\|\, Q \,\|\, G_v\mathbb{P}_v\mathbb{U}$ to play, the situations are symmetric.

(iii) For generality suppose $(C \,\|\, Q \,\|\, G_v\mathbb{P}^1_v, C \,\|\, Q \,\|\, G_v\mathbb{P}^2_v\mathbb{U})$ is the current configuration. Attacker would not choose any transition of the form

$$C \,\|\, Q \,\|\, G_v\mathbb{P}^1_v \xrightarrow{\ell} P_1$$

since the following response

$$C \,\|\, Q \,\|\, G_v\mathbb{P}^2_v\mathbb{U} \Longrightarrow C \,\|\, Q \Longrightarrow C \,\|\, Q \,\|\, G_v\mathbb{P}^1_v \xrightarrow{\ell} P_1 \qquad (3)$$

is a winning move for Defender. To see that none of the silent transitions appearing in (3) are change-of-state, first notice that $C \,\|\, Q \simeq C \,\|\, Q \,\|\, G_v\mathbb{P}^1_v$ by (2) of Corollary 1. The equivalence $C \,\|\, Q \,\|\, G_v\mathbb{P}^2_v\mathbb{U} \simeq C \,\|\, Q$ is derived as follows: One has that

$$C \,\|\, Q \,\|\, G_v\mathbb{P}^2_v\mathbb{U} \Longrightarrow C \,\|\, Q \Longrightarrow C \,\|\, Q \,\|\, G_v\mathbb{P}^2_v\mathbb{V}.$$

It is easy to see that Proposition 1 implies $C \,\|\, G_v\mathbb{P}^2_v\mathbb{U} \simeq C \,\|\, G_v\mathbb{P}^2_v\mathbb{V}$. It then follows from $C \,\|\, Q \,\|\, G_v\mathbb{P}^2_v\mathbb{U} \simeq C \,\|\, Q \,\|\, G_v\mathbb{P}^2_v\mathbb{V}$ and Lemma 1 that $C \,\|\, Q \,\|\, G_v\mathbb{P}^2_v\mathbb{U} \simeq C \,\|\, Q$.
Now suppose Attacker chooses $C \,\|\, Q \,\|\, G_v\mathbb{P}^2_v\mathbb{U}$ to play.

(a) If Attacker plays some $C \parallel Q \parallel G_v \mathbb{P}_v^2 \mathbb{U} \xrightarrow{\ell} P_2$ caused by either an action of Q or $C \xrightarrow{\tau} C \parallel G$ or $C \xrightarrow{\tau} C \parallel G_v$, Defender plays the same action, reaching to a configuration of the same shape.

(b) If Attacker plays $C \parallel Q \parallel G_v \mathbb{P}_v^2 \mathbb{U} \xrightarrow{\lambda_I} I \parallel Q \parallel G_v \mathbb{P}_v^2 \mathbb{U}$, Defender replies

$$C \parallel Q \parallel G_v \mathbb{P}_v^1 \Longrightarrow C \parallel Q \Longrightarrow C \parallel Q \parallel G_v \mathbb{P}_v^2 \mathbb{V} \xrightarrow{\lambda_I} I \parallel Q \parallel G_v \mathbb{P}_v^2 \mathbb{V}.$$

Suppose Attacker chooses $(I \parallel Q \parallel G_v \mathbb{P}_v^2 \mathbb{U}, I \parallel Q \parallel G_v \mathbb{P}_v^2 \mathbb{V})$ to be the next configuration. Now $C \parallel Q \parallel G_v \mathbb{P}_v^1 \simeq C \parallel Q \parallel G_v \mathbb{P}_v^2 \mathbb{V}$ by (2) of Corollary 1 and $I \parallel Z \mathbb{P}_v^2 \mathbb{V} \simeq I \parallel Z \mathbb{P}_v^2 \mathbb{U}$ by Lemma 5. It follows that

$$I \parallel Q \parallel G_v \mathbb{P}_v^2 \mathbb{V} \simeq I \parallel Q \parallel G_v \mathbb{P}_v^2 \mathbb{U}.$$

So in this case Defender wins.

(c) The situation is similar if Attacker chooses to play an λ_S action.

Attacker will not win if it keeps doing (a). Eventually it must do (b) or (c).

By applying the three substrategies consecutively we see that Attacker's optimal strategy is to reach a configuration of the form $(C \parallel Q \parallel G_v \mathbb{P}_v \mathbb{U}, C \parallel Q \parallel G_v \mathbb{P}_v \mathbb{V})$. This is however also a win situation for Defender because of the equivalence

$$C \parallel G_v \mathbb{P}_v \mathbb{U} \simeq C \parallel G_v \mathbb{P}_v \mathbb{V},$$

the simple proof of which is as follows:

– If $C \parallel G_v \mathbb{P}_v \mathbb{U}$ performs a λ_I or λ_S action, then $C \parallel G_v \mathbb{P}_v \mathbb{V}$ does the same. We are done by Lemma 5.
– If $C \parallel G_v \mathbb{P}_v \mathbb{U}$ does an action induced by $G_v \xrightarrow{\tau} \epsilon$, the process $C \parallel G_v \mathbb{P}_v \mathbb{V}$ follows suit. We are done by Corollary 1.
– If $C \parallel G_v \mathbb{P}_v \mathbb{U}$ acts using $G_v \xrightarrow{\lambda_V} Z$, then $C \parallel G_v \mathbb{P}_v \mathbb{V}$ copycats the action. We are done by Proposition 1.
– If $C \parallel G_v \mathbb{P}_v \mathbb{U} \xrightarrow{\tau} C \parallel G_v V_i \mathbb{P}_v \mathbb{U}$, then $C \parallel G_v \mathbb{P}_v \mathbb{V} \xrightarrow{\tau} C \parallel G_v V_i \mathbb{P}_v \mathbb{V}$. We get a pair of processes of the same shape.

This completes the proof.

E Proof of Lemma 7

Attacker's winning strategy is very much similar to the optimal strategy described in Section D. It is outlined in Fig. 6, where Attacker's moves are marked in red. We explain the strategy in the following.

– Attacker plays $X \xrightarrow{\lambda_U} D \parallel G_v$. Defender's optimal response would be

$$Y \Longrightarrow D \parallel G_u \mathbb{P}_u^1 \xrightarrow{\lambda_U} D \parallel G_v \mathbb{P}_u^2 \Longrightarrow D \parallel Q_1 \parallel G_v \mathbb{P}_v^1 \mathbb{P}_u^2$$

for some Q_1, $\mathbb{P}_v^1$, $\mathbb{P}_u^1$ and $\mathbb{P}_u^2$ such that $D \Longrightarrow D \parallel Q_1$ and $\mathbb{P}_u^2 = U_i \mathbb{P}_u^1$ for some $i \in \mathcal{N}$. Defender would not remove G_v using the rule $G_v \xrightarrow{\tau} \epsilon$ because that would make it unable to reply Attacker's next move $D \parallel G_v \xrightarrow{\lambda_V} D \parallel Z$.

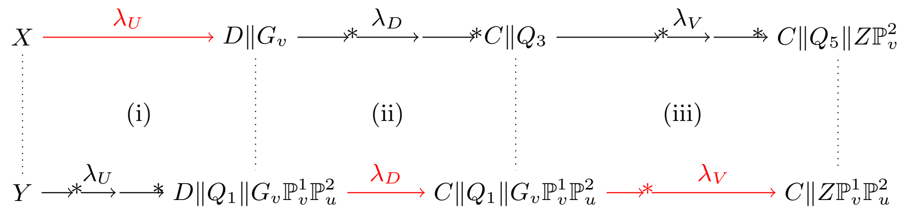

Fig. 6. Attacker's Strategy

- Attacker then plays $D \parallel Q_1 \parallel G_v\mathbb{P}_v^1\mathbb{P}_u^2 \xrightarrow{\lambda_D} C \parallel Q_1 \parallel G_v\mathbb{P}_v^1\mathbb{P}_u^2$. Defender's response must be of the form $D \parallel G_v \Longrightarrow D \parallel Q_2 \xrightarrow{\lambda_D} C \parallel Q_2 \Longrightarrow C \parallel Q_3$ for some Q_2, Q_3.
- It is easy to see that $Q_1 \Longrightarrow \epsilon$. So the following is a valid move of Attacker:

$$C \parallel Q_1 \parallel G_v\mathbb{P}_v^1\mathbb{P}_u^2 \Longrightarrow C \parallel G_v\mathbb{P}_v^1\mathbb{P}_u^2 \xrightarrow{\lambda_V} C \parallel Z\mathbb{P}_v^1\mathbb{P}_u^2.$$

  Defender's response must be a transition sequence of the following form

$$C \parallel Q_3 \Longrightarrow C \parallel Q_4 \parallel G_v\mathbb{P}_v^2 \xrightarrow{\lambda_V} C \parallel Q_4 \parallel Z\mathbb{P}_v^2 \Longrightarrow C \parallel Q_5 \parallel Z\mathbb{P}_v^2$$

  for some $\mathbb{P}_v^2$, Q_4 and Q_5. Now Q_5 must be a parallel composition of processes that can be generated by C or D. W.l.o.g. assume that

$$Q_5 = Q^C \parallel Q_1^D \parallel \ldots \parallel Q_k^D$$

  where Q^C is generated by C and D_i^D is generated by D for $i \in \{1, \ldots, k\}$. There are following cases:
  (a) If Q_i contains an occurrence of G_u, then Attacker wins because the process $C \parallel Q_5 \parallel Z\mathbb{P}_v^2$ can do a λ_U action that cannot be simulated by the process $C \parallel Z\mathbb{P}_v^1\mathbb{P}_u^2$.
  (b) If Q_i contains an occurrence of Z, then Attacker also wins since the process $C \parallel Q_5 \parallel Z\mathbb{P}_v^2$ can do two consecutive λ_Z actions whereas the process $C \parallel Z\mathbb{P}_v^1\mathbb{P}_u^2$ can do only one such action.
  (c) If for each $i \in \{1, \ldots, k\}$ the process Q_i contains neither G_u nor Z then $C \simeq C \parallel Q_i^D$ for all i. By Lemma 4 we must also have $C \parallel Q^C \simeq C$. It follows that $C \parallel Q_5 \simeq C$. So in this case the configuration the game reached is essentially $(C \parallel Z\mathbb{P}_v^2, C \parallel Z\mathbb{P}_v^1\mathbb{P}_u^2)$. By Lemma 8, $C \parallel Z\mathbb{P}_v^2 \not\approx C \parallel Z\mathbb{P}_v^1\mathbb{P}_u^2$, meaning that Attacker has a winning strategy.

We are done.