Refining the Undecidability Border of Weak Bisimilarity

Mojmír Křetínský¹ Vojtěch Řehák² Jan Strejček³

Faculty of Informatics, Masaryk University
Botanická 68a, 60200 Brno,
Czech Republic
{kreţinsky,rehak,strejcek}@fi.muni.cz

Abstract
Weak bisimilarity is one of the most studied behavioural equivalences. This equivalence is undecidable for pushdown processes (PDA), process algebras (PA), and multiset automata (MSA, also known as parallel pushdown processes, PPDA). Its decidability is an open question for basic process algebras (BPA) and basic parallel processes (BPP). We move the undecidability border towards these classes by showing that the equivalence remains undecidable for weakly extended versions of BPA and BPP. In fact, we show that the weak bisimulation equivalence problem is undecidable even for normed subclasses of BPA and BPP extended with a finite constraint system.

Key words: weak bisimulation, infinite-state systems, decidability

1 Introduction

Equivalence checking is one of the main streams in verification of concurrent systems. It aims at demonstrating some semantic equivalence between two systems, one of which is usually considered as representing the specification, the other as its implementation or refinement. The semantic equivalences are designed to correspond to the system behaviours at the desired level of abstraction; the most prominent ones being strong and weak bisimulations.

Current software systems often exhibit an evolving structure and/or operate on unbounded data types. Hence automatic verification of such systems usually requires modelling them as infinite-state ones. Various specification

¹ Supported by the Grant Agency of the Czech Republic, grant No. 201/03/1161.
² Supported by the research centre “Institute for Theoretical Computer Science (ITI)”, project No. 1M0021620808.
³ Supported by the Academy of Sciences of the Czech Republic, grant No. 1ET408050503.
formalisms have been developed with their respective advantages and limitations. Petri nets (PN), pushdown processes (PDA), and process algebras like BPA, BPP, or PA all serve to exemplify this. Here we employ the classes of infinite-state systems defined by term rewrite systems and called Process Rewrite Systems (PRS) as introduced by Mayr [12]. PRS subsume a variety of the formalisms studied in the context of formal verification (e.g. all the models mentioned above). The relevance of various subclasses of PRS for modelling and analysing programs is shown, for example, in [5]; for automatic verification we refer to surveys [2,22].

The relative expressive power of various process classes has been studied, especially with respect to strong bisimulation; see [3,16] and also [12] showing the strictness of the hierarchy of PRS classes. Adding a finite-state control unit to the PRS rewriting mechanism results in so-called state-extended PRS (sePRS) classes, see for example [8]. We have extended the PRS hierarchy by sePRS classes and refined this extended hierarchy by introducing restricted state extensions of two types: PRS with a weak finite-state control unit (wPRS, inspired by weak automata [17]) [11,10] and PRS with a finite constraint system (fcPRS) [24].

Research on the expressive power of process classes has been accompanied by exploring algorithmic boundaries of various verification problems. In this paper we focus on the equivalence checking problem taking weak bisimilarity as the notion of behavioral equivalence.

**State of the art:** Regarding sequential systems, i.e. those without parallel composition, the weak bisimilarity problem is undecidable for PDA even for the normed case [19]. However, it is conjectured [13] that weak bisimilarity is decidable for basic process algebras (BPA); the best known lower bound is EXPTIME-hardness [13].

Considering parallel systems, even strong bisimilarity is undecidable for multiset automata (MSA, also known as parallel pushdown processes or state-extended BPP) [16] using the technique introduced in [6]. However, it is conjectured [7] that the weak bisimilarity problem is decidable for basic parallel processes (BPP); the best known lower bound is PSPACE-hardness [20].

For the simplest systems combining both parallel and sequential operators, called PA processes [1], the weak bisimilarity problem is undecidable [21]. It is an open question for the normed PA; the best known lower bound is EXPTIME-hardness [13].

**Our contribution:** We move the undecidability border of the weak bisimilarity problem towards the classes of BPA and BPP, where the problem is conjectured to be decidable. Section 3 shows undecidability of the considered problem for the weakly extended versions of BPA (wBPA) and BPP (wBPP). In Section 4, we strengthen the result for even more restricted systems, namely for normed fcBPA and normed fcBPP systems. In fact, the result is not new for wBPA due to the following reasons: Mayr [13] has shown that adding a
2 Preliminaries

We recall the definitions of labelled transition system and weak bisimilarity. Then we define the syntax of process rewrite systems, (weak) finite-state unit extensions of PRS, and PRS with finite constraint systems. Their semantics is given in terms of labelled transition systems.

Definition 2.1 Let $\text{Act} = \{a, b, \ldots\}$ be a set of actions such that $\text{Act}$ contains a distinguished silent action $\tau$. A labelled transition system is a pair $(S, \rightarrow)$, where $S$ is a set of states and $\rightarrow \subseteq S \times \text{Act} \times S$ is a transition relation. We write $s_1 \rightarrow a s_2$ instead of $(s_1, a, s_2) \in \rightarrow$. The transition relation is extended to finite words over $\text{Act}$ in the standard way. Further, we extend the relation to language $L \subseteq \text{Act}^*$ such that $s_1 L \rightarrow s_2$ if $s_1 w \rightarrow s_2$ for some $w \in L$. Moreover, we write $s_1 \rightarrow^* s_2$ instead of $s_1 \text{Act}^* \rightarrow s_2$. The weak transition relation $\Rightarrow \subseteq S \times \text{Act} \times S$ is defined as $\tau \Rightarrow = \tau^*$ and $a \Rightarrow = \tau^* a \tau^*$ for all $a \neq \tau$.

Definition 2.2 A binary relation $R$ on states of a labelled transition system is a weak bisimulation iff whenever $(s_1, s_2) \in R$ then for any $a \in \text{Act}$:

- if $s_1 \rightarrow a s_1'$ then $s_2 \Rightarrow s_2'$ for some $s_2'$ such that $(s_1', s_2') \in R$ and
- if $s_2 \rightarrow a s_2'$ then $s_1 \Rightarrow a s_1'$ for some $s_1'$ such that $(s_1', s_2') \in R$.

States $s_1$ and $s_2$ are weakly bisimilar, written $s_1 \approx s_2$, iff $(s_1, s_2) \in R$ for some weak bisimulation $R$.

We use a characterization of weak bisimilarity in terms of a bisimulation game, see e.g. [23]. This is a two-player game between an attacker and a defender played in rounds on pairs of states of a considered labelled transition system. In a round starting at a pair of states $(s_1, s_2)$, the attacker first chooses $i \in \{1, 2\}$, an action $a \in \text{Act}$, and a state $s_1'$ such that $s_i \rightarrow a s_i'$. The defender then has to choose a state $s_{3-i}$ such that $s_{3-i} \Rightarrow s_{3-i}'$. The states $s_1', s_2'$ form a pair of starting states for the next round. A play is a maximal sequence of pairs of states chosen by players in the given way. The defender is the winner of every infinite play. A finite game is lost by the player who cannot make any choice satisfying the given conditions. It can be shown that two states $s_1, s_2$ of a labelled transition system are not weakly bisimilar if and only if the attacker has a winning strategy for the bisimulation game starting in these states.

Let $\text{Const} = \{X, \ldots\}$ be a set of process constants. The set of process
terms (ranged over by \( t, \ldots \)) is defined by the abstract syntax

\[
t ::= \varepsilon \mid X \mid t.t \mid t\|t
\]

where \( \varepsilon \) is the empty term, \( X \in \text{Const} \) is a process constant; and ‘\( . \)’ and ‘\( \| \)’ mean sequential and parallel composition respectively. We always work with equivalence classes of terms modulo commutativity and associativity of ‘\( \| \)’, associativity of ‘\( . \)’, and neutrality of \( \varepsilon \), i.e. \( \varepsilon.t = t.\varepsilon = t\|\varepsilon = t \). We distinguish four classes of process terms as:

1 – terms consisting of a single process constant only, in particular \( \varepsilon \not\in 1 \),

\( S \) – sequential terms - terms without parallel composition, e.g. \( X.Y.Z \),

\( P \) – parallel terms - terms without sequential composition, e.g. \( X\|Y\|Z \),

\( G \) – general terms - terms with arbitrarily nested sequential and parallel compositions, e.g. \((X.(Y\|Z))\|W) \).

**Definition 2.3** Let \( \alpha, \beta \) be classes of process terms \( \alpha, \beta \in \{1, S, P, G\} \) such that \( \alpha \subseteq \beta \). An \( (\alpha, \beta)\)-PRS (process rewrite system) \( \Delta \) is a finite set of rewrite rules of the form \( t_1 \overset{a}{\rightarrow} t_2 \), where \( t_1 \in \alpha \smallsetminus \{\varepsilon\} \), \( t_2 \in \beta \) are process terms and \( a \in \text{Act} \) is an action.

Given a PRS \( \Delta \), let \( \text{Const}(\Delta) \) and \( \text{Act}(\Delta) \) be the respective (finite) sets of all constants and all actions which occur in the rewrite rules of \( \Delta \).

An \( (\alpha, \beta)\)-PRS \( \Delta \) determines a labelled transition system where states are process terms \( t \in \beta \) over \( \text{Const}(\Delta) \). The transition relation \( \longrightarrow \) is the least relation satisfying the following inference rules (recall that ‘\( \| \)’ is commutative):

\[
\frac{(t_1 \overset{a}{\rightarrow} t_2) \in \Delta}{t_1 \overset{a}{\longrightarrow} t_2}
\]

\[
\frac{t_1 \overset{a}{\rightarrow} t_2}{t_1\|t \overset{a}{\rightarrow} t_2\|t}
\]

\[
\frac{t_1 \overset{a}{\rightarrow} t_2}{t_1.t \overset{a}{\rightarrow} t_2.t}
\]

The formalism of process rewrite systems can be extended to include a finite-state control unit in the following way.

**Definition 2.4** Let \( M = \{m, n, \ldots\} \) be a set of control states. Let \( \alpha, \beta \) be classes of process terms \( \alpha, \beta \in \{1, S, P, G\} \) such that \( \alpha \subseteq \beta \). An \( (\alpha, \beta)\)-sePRS (state extended process rewrite system) \( \Delta \) is a finite set of rewrite rules of the form \( (m, t_1) \overset{a}{\rightarrow} (n, t_2) \), where \( t_1 \in \alpha \smallsetminus \{\varepsilon\} \), \( t_2 \in \beta \), \( m, n \in M \), and \( a \in \text{Act} \).

\( M(\Delta) \) denotes the finite set of control states which occur in \( \Delta \).

An \( (\alpha, \beta)\)-sePRS \( \Delta \) determines a labelled transition system where states are the pairs of the form \( (m, t) \) such that \( m \in M(\Delta) \) and \( t \in \beta \) is a process term over \( \text{Const}(\Delta) \). The transition relation \( \longrightarrow \) is the least relation satisfying the following inference rules:

\[
\frac{(m, t_1) \overset{a}{\rightarrow} (n, t_2) \in \Delta}{(m, t_1) \overset{a}{\rightarrow} (n, t_2)}
\]

\[
\frac{(m, t_1) \overset{a}{\rightarrow} (n, t_2)}{(m, t_1.t) \overset{a}{\rightarrow} (n, t_2.t)}
\]
To shorten our notation we write $mt$ in lieu of $(m, t)$.

**Definition 2.5** An $(\alpha, \beta)$-sePRS $\Delta$ is called a *process rewrite system with a weak finite-state control unit* or just a *weakly extended process rewrite system*, written $(\alpha, \beta)$-wPRS, if there exists a partial order $\leq$ on $M(\Delta)$ such that every rule $(m, t_1) \xrightarrow{a} (n, t_2)$ of $\Delta$ satisfies $m \leq n$.

Finally, we recall the extension of *process rewrite systems with finite constraint systems* introduced in [24]. This extension has been directly motivated by constraint systems used in concurrent constraint programming (CCP), for example, see [18].

**Definition 2.6** A *finite constraint system* is a bounded lattice $(C, \geq, \wedge, \tt, \ff)$, where $C$ is a finite set of constraints, $\geq$ (called entailment) is a partial ordering on this set, $\wedge$ is the least upper bound operation, and $\tt$ (true), $\ff$ (false) are the least and the greatest elements of $C$ respectively $(\ff \geq \tt$ and $\tt \neq \ff)$.

**Example 2.7** An example of a constraint system given by its Hasse diagram.

```
     tt
     |   |
   ff   m   n
     |   |
     tt
```

**Definition 2.8** Let $\alpha, \beta$ be classes of process terms, $\alpha, \beta \in \{1, S, P, G\}$, such that $\alpha \subseteq \beta$. Let $(C(\Delta), \geq, \wedge, \tt, \ff)$ be a finite constraint system. An $(\alpha, \beta)$-fcPRS (PRS with a finite constraint system) $\Delta$ is a finite set of rewrite rules of the form $(m, t_1) \xrightarrow{a} (n, t_2)$, where $t_1 \in \alpha$, $t_1 \neq \epsilon$, $t_2 \in \beta$ are process terms, $a \in \text{Act}$, and $m, n \in C(\Delta)$ are constraints.

An $(\alpha, \beta)$-fcPRS $\Delta$ determines a labelled transition system where states are the pairs of the form $(m, t)$ such that $m \in C(\Delta) \setminus \{\ff\}$ and $t \in \beta$ is a process term over $\text{Const}(\Delta)$. The transition relation $\longrightarrow$ is the least relation satisfying the following inference rules:

\[
\begin{align*}
(m, t_1) \xrightarrow{a} (n, t_2) \in \Delta & \quad \text{if } o \geq m \text{ and } o \wedge n \neq \ff, \\
(o, t_1) \xrightarrow{a} (o \wedge n, t_2) & \\
(o, t_1 \parallel t) \xrightarrow{a} (p, t_2 \parallel t) & , \\
(o, t_1) \xrightarrow{a} (p, t_2) \\
(o, t_1.t) \xrightarrow{a} (p, t_2.t) & .
\end{align*}
\]

To shorten our notation we write $mt$ in lieu of $(m, t)$.

As in CCP, the constraint system describes possible behaviour of a *store*. The constraint $m$ in a state $mt$ represents a current value of the store. The two side conditions of the first inference rule are also very close to principles used in CCP. The first one $(o \geq m)$ ensures the rule $(mt_1 \xrightarrow{a} nt_2) \in \Delta$ can
be used only if the current value of the store \( o \) entails \( m \) (it is similar to \( \text{ask}(m) \) in CCP). The second condition \((o \land n \neq \text{ff})\) guarantees that the store stays consistent after the application of the rule (analogous to a consistency requirement when processing \( \text{tell}(n) \) in CCP).

An important observation is that the value of a store can move in a lattice only upwards as \( o \land n \) always entails \( o \). Intuitively, partial information can only be added to the store, but never retracted (the store is monotonic).

We note that an execution of a transition which starts in a state with \( o \) on the store and which is generated by a rule \((m_{t_1} \xrightarrow{a} nt_2) \in \Delta\) implies that for every subsequent value of the store \( p \) the conditions \( p \geq m \) and \( p \land n \neq \text{ff} \) are satisfied (and thus the use of the rule cannot be forbidden by a value of the store in the future). The first condition \( p \geq m \) comes from the monotonic behaviour of the store. The second condition comes from two following facts: the constraint \( n \) of the rule can only change the store in the first application of the rule; and \( p \land n = p \) holds for any subsequent state \( p \) of the store.

**Definition 2.9** An \((\alpha, \beta)\)-fcPRS \( \Delta \) is normed in a state \( m_0t_0 \) of \( \Delta \) if and only if, for all states \( mt \) satisfying \( m_0t_0 \xrightarrow{*} mt \), it holds that \( mt \xrightarrow{*} o\varepsilon \) for some \( o \in C(\Delta) \).

Some classes of \((\alpha, \beta)\)-PRS correspond to widely known models as finite-state systems (FS), basic process algebras (BPA), basic parallel processes (BPP), process algebras (PA), pushdown processes (PDA, see [4] for justification), and Petri nets (PN). The other \((\alpha, \beta)\)-PRS classes were introduced and named as PAD, PAN, and PRS by Mayr [12]. The correspondence between \((\alpha, \beta)\)-PRS classes and the acronyms is given in Figure 1. Instead of \((\alpha, \beta)\)-sePRS, \((\alpha, \beta)\)-wPRS, and \((\alpha, \beta)\)-fcPRS we use the prefixes ‘se-’, ‘w-’, and ‘fc-’ in connection with the acronym for the corresponding \((\alpha, \beta)\)-PRS class. For example, we use \( \text{wBPA} \) and \( \text{wBPP} \) rather than \((1, S)\)-wPRS and \((1, P)\)-wPRS, respectively. Finally, we note that \( \text{seBPP} \) are also known as multiset automata (MSA) or parallel pushdown processes (PPDA).

Figure 1 depicts relations between the expressive power of the considered classes. The expressive power of a class is measured by the set of labelled transition systems that are definable (up to strong bisimulation equivalence) by the class. A solid line between two classes means that the upper class is strictly more expressive than the lower one. A dotted line means that the upper class is at least as expressive as the lower class (and the strictness is just our conjecture). Details can be found in [11,10].

### 3 Undecidability of Weak Bisimilarity

In this section, we show that weak bisimilarity is undecidable for the classes \( \text{wBPA} \) and \( \text{wBPP} \). More precisely, we study the following problems for extended \((\alpha, \beta)\)-PRS classes.
Problem: Weak bisimilarity problem for an extended (α, β)-PRS class

Instance: An extended (α, β)-PRS system ∆ and two of its states mt, m′

Question: Are the two states mt and m′ weakly bisimilar?

3.1 wBPA

In [13] Mayr studied the question of how many control states are needed in PDA to make weak bisimilarity undecidable.

Theorem 3.1 ([13], Theorem 29) Weak bisimilarity is undecidable for pushdown automata with only 2 control states.

The proof is done by a reduction of Post’s correspondence problem to the weak bisimilarity problem for PDA. The constructed PDA has only two control states, p and q. Quick inspection of the construction shows that the resulting pushdown automata are in fact wBPA systems as there is no transition rule changing q to p and each rule has only one process constant on the left hand
Theorem 3.2 Weak bisimilarity is undecidable for wBPA systems with only 2 control states.

3.2 wBPP

We show that the non-halting problem for Minsky 2-counter machines can be reduced to the weak bisimilarity problem for wBPP. First, let us recall the notions of Minsky 2-counter machine and the non-halting problem.

A Minsky 2-counter machine, or a machine for short, is a finite sequence

\[ N = l_1 : i_1, l_2 : i_2, \ldots, l_{n-1} : i_{n-1}, l_n : \text{halt} \]

where \( n \geq 1, l_1, l_2, \ldots, l_n \) are labels, and each \( i_j \) is an instruction for

- increment: \( c_k := c_k + 1; \text{ goto } l_r \), or
- test-and-decrement: \( \text{ if } c_k > 0 \text{ then } c_k := c_k - 1; \text{ goto } l_r \text{ else goto } l_s \)

where \( k \in \{1, 2\} \) and \( 1 \leq r, s \leq n \).

The semantics of a machine \( N \) is given by a labelled transition system the states of which are configurations of the form \( (l_j, v_1, v_2) \) where \( l_j \) is a label of an instruction to be executed and \( v_1, v_2 \) are nonnegative integers representing current values of counters \( c_1 \) and \( c_2 \), respectively. The transition relation is the smallest relation satisfying the following conditions: if \( i_j \) is an instruction of the form

- \( c_1 := c_1 + 1; \text{ goto } l_r \), then \( (l_j, v_1, v_2) \xrightarrow{\text{inc}} (l_r, v_1 + 1, v_2) \) for all \( v_1, v_2 \geq 0 \);
- \( \text{if } c_1 > 0 \text{ then } c_1 := c_1 - 1; \text{ goto } l_r \text{ else goto } l_s \), then \( (l_j, v_1 + 1, v_2) \xrightarrow{\text{dec}} (l_r, v_1, v_2) \) and \( (l_j, 0, v_2) \xrightarrow{\text{zero}} (l_s, 0, v_2) \) for all \( v_1, v_2 \geq 0 \);

and the analogous condition for instructions manipulating \( c_2 \). We say that the (computation of) machine \( N \) halts if there are numbers \( v_1, v_2 \geq 0 \) such that \( (l_1, 0, 0) \xrightarrow{*} (l_n, v_1, v_2) \). Let us note that the system is deterministic, i.e. for every configuration there is at most one transition leading from the configuration.

The non-halting problem is to decide whether a given machine \( N \) does not halt. The problem is undecidable [15].

Let us fix a machine \( N = l_1 : i_1, l_2 : i_2, \ldots, l_{n-1} : i_{n-1}, l_n : \text{halt} \). We construct a wBPP system \( \Delta \) such that its states \( \text{sim} L_1 \) and \( \text{sim} L'_1 \) are weakly bisimilar if and only if \( N \) does not halt. Roughly speaking, we create a set of wBPP rules allowing us to simulate the computation of \( N \) by two separate sets of terms. If the instruction \( \text{halt} \) is reached in the computation of \( N \), the corresponding term from one set can perform the action \( \text{halt} \), while the corresponding term from the other set can perform the action \( \text{halt}' \). Therefore, the starting terms are weakly bisimilar if and only if the machine does not halt.
The wBPP system $\Delta$ we are going to construct uses five control states, namely $\text{sim}, \text{check}_1, \text{check}'_1, \text{check}_2, \text{check}'_2$. We associate each label $l_j$ and each counter $c_k$ with process constants $L_j, L_j', X_k, Y_k$ respectively. A parallel composition of $x$ copies of $X_k$ and $y$ copies of $Y_k$, written $X^x_k \parallel Y^y_k$, represents the fact that the counter $c_k$ has the value $x - y$. Hence, terms $\text{sim} L_j \parallel X^x_k \parallel Y^y_k$, and $\text{sim} L_j' \parallel X^x_k \parallel Y^y_k$ are associated with a configuration $(l_j, x_1 - y_1, x_2 - y_2)$ of the machine $N$. Some rules contain auxiliary process constants. In what follows, $\beta$ stands for a term of the form $\beta = X^x_k \parallel Y^y_k$. The control states $\text{check}_k, \text{check}'_k$ for $k \in \{1, 2\}$ are intended for testing emptiness of the counter $c_k$. The only rules applicable in these control states are:

\[
\begin{align*}
\text{check}_1 X_1 & \leftrightarrow \text{check}_1 \varepsilon \\
\text{check}'_1 Y_1 & \leftrightarrow \text{check}'_1 \varepsilon
\end{align*}
\]

\[
\begin{align*}
\text{check}_2 X_2 & \leftrightarrow \text{check}_2 \varepsilon \\
\text{check}'_2 Y_2 & \leftrightarrow \text{check}'_2 \varepsilon
\end{align*}
\]

One can readily confirm that $\text{check}_k \beta \approx \text{check}'_k \beta$ if and only if the value of $c_k$ represented by $\beta$ equals zero.

In what follows we construct a set of wBPP rules for each instruction of the machine $N$. At the same time we argue that the only chance for the attacker to win is to simulate the machine without cheating as every cheating can be punished by the defender’s victory. This attacker’s strategy is winning if and only if the machine halts.

**Halt:** $l_n: \text{halt}$

Halt instruction is translated into the following two rules:

\[
\begin{align*}
\text{sim} L_n & \leftrightarrow \text{sim} \varepsilon \\
\text{sim} L'_n & \rightarrow \text{sim} \varepsilon
\end{align*}
\]

Clearly, the states $\text{sim} L_n \parallel \beta$ and $\text{sim} L'_n \parallel \beta$ are not weakly bisimilar.

**Increment:** $l_j: c_k := c_k + 1; \text{ goto } l_r$

For each such an instruction of the machine $N$ we add the following rules to $\Delta$:

\[
\begin{align*}
\text{sim} L_j & \leftrightarrow \text{sim} L_r \parallel X_k \\
\text{sim} L'_j & \leftrightarrow \text{sim} L'_r \parallel X_k
\end{align*}
\]

Obviously, every round of the bisimulation game starting at states $\text{sim} L_j \parallel \beta$ and $\text{sim} L'_j \parallel \beta$ ends up in states $\text{sim} L_r \parallel X_k \parallel \beta$ and $\text{sim} L'_r \parallel X_k \parallel \beta$.

**Test-and-decrement:** $l_j: \text{if } c_k > 0 \text{ then } c_k := c_k - 1; \text{ goto } l_r \text{ else } \text{ goto } l_s$

For any such instruction of the machine $N$ we add two sets of rules to $\Delta$, one for the $c_k > 0$ case and the other for the $c_k = 0$ case. The wBPP formalism has no power to rewrite a process constant corresponding to a label $l_j$ and to
check whether \( c_k > 0 \) at the same time. Therefore, in the bisimulation game it is the attacker who has to decide whether \( c_k > 0 \) holds or not, i.e. whether he will play an action \( \text{dec} \) or an action \( \text{zero} \). We show that whenever the attacker tries to cheat, the defender can win the game.

At this point our construction of wBPP rules uses a variant of the technique called \textit{defender’s choice} [9]. In a round starting at the pair of states \( s_1, s_2 \), the attacker is forced to choose one specific transition (indicated by a wavy arrow henceforth). If he chooses a different transition, say \( s_k \to s \) where \( k \in \{1, 2\} \), then there exists a transition \( s_{3-k} \to s \) that enables the defender to reach the same state and win the play. The name of this technique refers to the fact that after the attacker chooses the specific transition, the defender can choose an arbitrary transition with the same label. These transitions are indicated by solid arrows. The dotted arrows stand for auxiliary transitions which compel the attacker to play the specific transition.

First, we discuss the following rules designed for the \( c_k > 0 \) case:

\[
\begin{align*}
\text{sim} L_j \xrightarrow{\text{dec}} & \text{sim} A_{k,r} \quad \text{sim} A_{k,r} \xrightarrow{\text{dec}} \text{check}_k \varepsilon \quad \text{sim} B_{k,r} \xrightarrow{\text{dec}} \text{sim} L_r \parallel Y_k \\
\text{sim} L_j \xrightarrow{\text{dec}} & \text{sim} B_{k,r} \quad \text{sim} A_{k,r} \xrightarrow{\text{dec}} \text{sim} L'_r \parallel Y_k \quad \text{sim} B_{k,r} \xrightarrow{\text{dec}} \text{sim} L'_r \parallel Y_k \\
\text{sim} L'_j \xrightarrow{\text{dec}} & \text{sim} A_{k,r} \quad \text{sim} A_{k,r} \xrightarrow{\text{dec}} \text{check}'_k \varepsilon \quad \text{sim} B_{k,r} \xrightarrow{\text{dec}} \text{check}'_k \varepsilon \\
\text{sim} L'_j \xrightarrow{\text{dec}} & \text{sim} B_{k,r} \quad \text{sim} A_{k,r} \xrightarrow{\text{dec}} \text{check}_k \varepsilon \\
\text{check}_k \varepsilon \xrightarrow{\text{dec}} & \text{sim} C_{k,r} \\
\text{sim} L_r \parallel Y_k \varepsilon \xrightarrow{\text{dec}} & \text{sim} L'_r \parallel Y_k \\
\text{sim} L'_r \parallel Y_k \varepsilon \xrightarrow{\text{dec}} & \text{check}'_k \varepsilon
\end{align*}
\]

The situation can be depicted as follows.

Let us assume that in a round starting at states \( \text{sim} L_j \parallel \beta, \text{sim} L'_j \parallel \beta \) the attacker decides to perform the action \( \text{dec} \). Due to the principle of defender’s choice employed here, the attacker has to play the transition \( \text{sim} L'_j \parallel \beta \xrightarrow{\text{dec}} \text{sim} C_{k,r} \parallel \beta \), while the defender can choose between the transitions leading from \( \text{sim} L_j \parallel \beta \) either to \( \text{sim} A_{k,r} \parallel \beta \) or to \( \text{sim} B_{k,r} \parallel \beta \). Thus, the round will finish either in states \( \text{sim} A_{k,r} \parallel \beta, \text{sim} C_{k,r} \parallel \beta \) or in states \( \text{sim} B_{k,r} \parallel \beta, \text{sim} C_{k,r} \parallel \beta \). Using the defender’s choice again, one can easily see that the next round ends up in \( \text{check}_k \beta \) or \( \text{sim} L_r \parallel Y_k \parallel \beta \), and \( \text{sim} L'_r \parallel Y_k \parallel \beta \) or \( \text{check}'_k \beta \). The exact combination
Křetínský et al.

is chosen by the defender. The defender will not choose any pair of states where one control state is \( \text{sim} \) and the other is not as such states are clearly not weakly bisimilar. Hence, the two considered rounds of the bisimulation game end up in a pair of states either \( \text{sim} L_r \parallel Y_k \parallel \beta \), \( \text{sim} L'_r \parallel Y_k \parallel \beta \) or \( \text{check}_k \beta \), \( \text{check}'_k \beta \).

The latter pair is weakly bisimilar iff the value of \( c_k \) represented by \( \beta \) is zero, i.e. iff the attacker cheats when he decides to play an action \( \text{dec} \). This means that if the attacker cheats, the defender wins. If the attacker plays the action \( \text{dec} \) correctly, the only chance for either player to force a win is to finish these two rounds in states \( \text{sim} L_r \parallel Y_k \parallel \beta \), \( \text{sim} L'_r \parallel Y_k \parallel \beta \) corresponding to the correct simulation of an test-and-decrement instruction with a label \( l_j \).

Now, we focus on the following rules designed for the \( c_k = 0 \) case:

\[
\begin{align*}
\text{sim} L_j \overset{\text{zero}}{\leftrightarrow} \text{sim} D_{k,s} & \quad \text{sim} D_{k,s} \overset{\text{zero}}{\leftrightarrow} \text{check}_k \varepsilon & \quad \text{sim} E_{k,s} \overset{\text{zero}}{\leftrightarrow} \text{sim} L_s \\
\text{sim} L_j \overset{\text{zero}}{\leftrightarrow} \text{sim} E_{k,s} & \quad \text{sim} D_{k,s} \overset{\text{zero}}{\leftrightarrow} \text{sim} L'_s & \quad \text{sim} E_{k,s} \overset{\text{zero}}{\leftrightarrow} \text{sim} L'_s \\
\text{sim} L'_j \overset{\text{zero}}{\leftrightarrow} \text{sim} D_{k,s} & \quad \text{sim} D_{k,s} \overset{\text{zero}}{\leftrightarrow} \text{sim} G_k & \quad \text{sim} E_{k,s} \overset{\text{zero}}{\leftrightarrow} \text{sim} G_k \\
\text{sim} L'_j \overset{\text{zero}}{\leftrightarrow} \text{sim} E_{k,s} & \quad \text{sim} F_{k,s} \overset{\text{zero}}{\leftrightarrow} \text{sim} L'_s & \quad \text{sim} G_k \overset{\tau}{\rightarrow} \text{sim} G_k \parallel Y_k \\
\text{sim} L'_j \overset{\text{zero}}{\leftrightarrow} \text{sim} F_{k,s} & \quad \text{sim} F_{k,s} \overset{\text{zero}}{\leftrightarrow} \text{sim} G_k & \quad \text{sim} G_k \overset{\tau}{\rightarrow} \text{check}'_k Y_k
\end{align*}
\]

The situation can be depicted as follows.

\[\text{Let us assume that the attacker decides to play the action zero. The defender’s choice technique allows the defender to control the two rounds of the bisimulation game starting at states } \text{sim} L_j \parallel \beta \text{ and } \text{sim} L'_j \parallel \beta. \text{ The two rounds end up in a pair of states } \text{sim} L_s \parallel \beta, \text{sim} L'_s \parallel \beta \text{ or in a pair of the form } \text{check}_k \beta, \text{check}'_k Y^m \parallel \beta \text{ where } m \geq 1; \text{ all the other choices of the defender lead to his loss. As in the previous case, the latter possibility is designed to punish any possible attacker’s cheating. The attacker is cheating if he plays}\]
the action zero and the value of $c_k$ represented by $\beta$, say $v_k$, is positive. In such a case, the defender can control the two rounds to end up in states $\text{check}_k\beta, \text{check}_k'Y_k^\alpha\|\beta$ which are weakly bisimilar. If the attacker plays correctly, i.e. the value of $c_k$ represented by $\beta$ is zero, then the defender has to control the two discussed rounds to end up in states $\text{sim}_L\|\beta, \text{sim}_L'\|\beta$ as the states $\text{check}_k\beta, \text{check}_k'Y_k^\alpha\|\beta$ are not weakly bisimilar for any $m \geq 1$. To sum up, the attacker’s cheating can be punished by defender’s victory. If the attacker plays correctly, the only chance for both players to win is to end up after the two rounds in states $\text{sim}_L\|\beta, \text{sim}_L'\|\beta$ corresponding to the correct simulation of the considered instruction.

It has been argued that if each of the two players wants to win, then both players will correctly simulate the computation of the machine $N$. The computation is finite if and only if the machine halts. The states $\text{sim}_L^1$ and $\text{sim}_L'^1$ are not weakly bisimilar in this case. If the machine does not halt, the play is infinite and the defender wins. Hence, the two states are weakly bisimilar in this case. In other words, the states $\text{sim}_L^1$ and $\text{sim}_L'^1$ of the constructed wBPP $\Delta$ are weakly bisimilar if and only if the Minsky 2-counter machine $N$ does not halt. Hence, we have proved the following theorem.

**Theorem 3.3** Weak bisimilarity is undecidable for wBPP systems.

### 4 Weak Bisimilarity for More Restricted Classes

Here, we strengthen the results of the previous section. We will show that weak bisimilarity remains undecidable for both fcBPP and fcBPA systems. Moreover, this holds even for their respective normed versions (i.e. if, in an instance of the weak bisimilarity problem, a given fcBPP/fcBPA system is normed in both given states). Hence, weak bisimilarity is undecidable for normed wBPP and normed wBPA as well.

#### 4.1 Normed fcBPP

In this subsection, we show that weak bisimilarity is undecidable for normed fcBPP systems.

Let $\Delta$ be the wBPP system constructed in Subsection 3.2. We recall that given any fixed Minsky machine $N$, we have constructed a wBPP system $\Delta$ such that its states $\text{sim}_L^1$ and $\text{sim}_L'^1$ are weakly bisimilar if and only if $N$ does not halt.

Based on $\Delta$, we now construct a fcBPP $\Delta'$ and two of its states $\text{sim}_L^1\|D$ and $\text{sim}_L'^1\|D$ such that they satisfy the same condition as given in the previous paragraph and moreover $\Delta'$ is normed in both of the states $\text{sim}_L^1\|D$ and $\text{sim}_L'^1\|D$.

---

4 We do not have to consider the case when $\beta$ represents a negative value of $c_k$ as such a state is reachable in the game starting in states $\text{sim}_L^1, \text{sim}_L'^1$ only by unpunished cheating.
The constraint system of $\Delta'$ is defined as follows.

Let $\text{Const}(\Delta') = \{D\} \cup \text{Const}(\Delta)$ and $\text{Act}(\Delta') = \{\text{norm}\} \cup \text{Act}(\Delta)$, where $D \notin \text{Const}(\Delta)$ is a fresh process constant and $\text{norm} \notin \text{Act}(\Delta)$ is a fresh action.

The set $\Delta'$ consists of all the rewrite rules in $\Delta$ and the following rules:

1. $\text{tt}D \xrightarrow{\text{norm}} \text{del}D$,
2. $\text{del}X \xrightarrow{\tau} \text{del} \varepsilon$ for all $X \in \text{Const}(\Delta')$,
3. $\text{del}X \xrightarrow{a} \text{del}X$ for all $X \in \text{Const}(\Delta')$ and $a \in \text{Act}(\Delta)$.

The process constant $D$ enables the $\text{norm}$ action changing the value of the store onto $\text{del}$. Starting in the state $\text{sim}L_1 \parallel D$ or $\text{sim}L'_1 \parallel D$, every reachable state includes the process constant $D$ or the current value of the store has been already changed onto $\text{del}$. Whenever the value of the store is set to $\text{del}$, the rules of type (2) can be used to make the state normed. Hence, $\Delta'$ is normed in both of the states $\text{sim}L_1 \parallel D$ and $\text{sim}L'_1 \parallel D$.

The rewrite rules of the type (3) have been introduced as the result of the fact that one cannot forbid any further applications of the original rules taken from $\Delta$ in the considered fcBPP systems.

Using the $\text{norm}$ action in the game, weakly bisimilar states are received. As only the attacker can decide for the action, this reconstruction of $\Delta$ onto $\Delta'$ does not change the winning strategies discussed in Subsection 3.2. Hence the Theorem 3.3 can be strengthen as follows.

**Theorem 4.1** Weak bisimilarity is undecidable for normed fcBPP systems.

### 4.2 Normed fcBPA

In this subsection, we show that the problem remains undecidable for normed fcBPA. Our proof is a slightly extended translation of the proof for PDA of [13] into fcBPA framework. We used the notation of [13] to make the proof comparable.

The proof is based on a reduction of Post’s correspondence problem, which is known to be undecidable [14].
Problem: Post's Correspondence problem (PCP)

Instance: A non-unary alphabet Σ and two ordered sets of words
\[ A = \{u_1, \ldots, u_n\} \text{ and } B = \{v_1, \ldots, v_n\} \text{ where } u_i, v_i \in \Sigma^+ \]

Question: Do there exist finitely many indices \( i_1, \ldots, i_m \in \{1, \ldots, n\} \)
such that \( u_{i_1} \cdots u_{i_m} = v_{i_1} \cdots v_{i_m} \)?

Given any instance of PCP we now construct a normed fcBPA \( \Delta \) and two
of its states \( p_{TB}, p_{T'B} \) such that \( p_{TB} \) and \( p_{T'B} \) are weakly bisimilar if and
only if the instance of PCP has a solution.

A constraint system of \( \Delta \) contains elements \( tt, p, check_1, check_2, del \), and \( ff \) that are ordered as follows.

\[
\begin{array}{c}
\text{ff} \\
\text{del} \\
check_1 \quad \check_2 \\
p \\
tt
\end{array}
\]

We use process constants \( T, T', T_1, T'_1, T_2, T'_2, G_l, G_r, B \) and \( U_i, V_i \) for each
\( 1 \leq i \leq n \). Actions of \( \Delta \) are \( a, b, c, \tau, \text{norm}, 1, \ldots, n \) and the letters of \( \Sigma \). In
what follows, \( \mathcal{U} \) stands for a sequential term of process constants of \( \{U_i \mid 1 \leq i \leq n\} \) and similarly \( \mathcal{V} \) stands for a sequential term of process constants of
\( \{V_i \mid 1 \leq i \leq n\} \).

Now, we construct a set of rewrite rules \( \Delta \). The rules of types (1)–(10)
are exactly the same as those of Mayr’s proof and forms a defender’s choice
construction.

\[
\begin{align*}
(1) & \quad pT \xrightarrow{a} pT_1 \\
(2) & \quad pT \xrightarrow{\tau} pG_r \\
(3) & \quad pT' \xrightarrow{\tau} pG_r \\
(4) & \quad pG_r \xrightarrow{a} pG_r V_i \quad \text{for all } i \in \{1, \ldots, n\} \\
(5) & \quad pG_r \xrightarrow{a} pT'_1 \\
(6) & \quad pT_1 \xrightarrow{a} pG_l \\
(7) & \quad pT'_1 \xrightarrow{a} pG_l B \\
(8) & \quad pT'_1 \xrightarrow{a} pT'_2 \\
(9) & \quad pG_l \xrightarrow{\tau} pG_l U_i \quad \text{for all } i \in \{1, \ldots, n\} \\
(10) & \quad pG_l \xrightarrow{\tau} pT_2
\end{align*}
\]
If there is a solution of the instance of PCP, the defender can use these rules to finish the first two rounds of the bisimulation game (starting in $pTB$ and $pT'_B$) in states $pT_2 UB$ and $pT'_2 VB$, where $U$ and $V$ form a solution of the PCP instance. The discussed first two rounds of the bisimulation game are depicted in Figure 2. We use the same notation for arrows as in Subsection 3.2.

The following six rules form two subsequent rounds of the bisimulation game and allow attacker to decide whether to check equality of indices or equality of the words of $U$ and $V$. In the first case, the attacker uses action $b$ leading to the constraint $\text{check}_1$, while the second possibility is labelled by $c$ and ends in the constraint $\text{check}_2$. The rewrite rules are as follows.

\begin{align*}
(11) & \quad pT_2 \xrightarrow{a} pT_3 & (12) & \quad pT'_2 \xrightarrow{a} pT'_3 \\
(13) & \quad pT_3 \xrightarrow{b} \text{check}_1 \varepsilon & (14) & \quad pT'_3 \xrightarrow{b} \text{check}_1 \varepsilon \\
(15) & \quad pT_3 \xrightarrow{c} \text{check}_2 \varepsilon & (16) & \quad pT'_3 \xrightarrow{c} \text{check}_2 \varepsilon
\end{align*}

Now, we list the rules that serve for the checking phases mentioned in the previous paragraph. In rules (19) and (20), we use a short notation that can
be easily expressed by standard rules. The rewrite rules are as follows.

\[
\begin{align*}
(17) & \quad \text{check}_1U_i \xrightarrow{i} \text{check}_1\varepsilon \quad \text{for all } i \in \{1, \ldots, n\} \\
(18) & \quad \text{check}_1V_i \xrightarrow{i} \text{check}_1\varepsilon \quad \text{for all } i \in \{1, \ldots, n\} \\
(19) & \quad \text{check}_2U_i \xrightarrow{vi} \text{check}_2\varepsilon \quad \text{for all } i \in \{1, \ldots, n\} \\
(20) & \quad \text{check}_2V_i \xrightarrow{vi} \text{check}_2\varepsilon \quad \text{for all } i \in \{1, \ldots, n\}
\end{align*}
\]

Finally, we add rules that make the system normed. The construction of rules (21) and (22) is also discussed in Remark 30 of [13]. The rules of type (21) enables the norm action changing the value of the store onto \(\text{del}\). In any state, whenever the value of the store is set to \(\text{del}\), the rules of type (3) can be used to make the state normed. Hence, \(\Delta\) is normed in all of its states. The rules of type (23) make all states composed of the constraint \(\text{del}\) and a non-empty term weakly bisimilar.

\[
\begin{align*}
(21) & \quad \text{tt}X \xrightarrow{\text{norm}} \text{del}X \quad \text{for all } X \in \text{Const}(\Delta) \\
(22) & \quad \text{del}X \xrightarrow{x} \text{del}\varepsilon \quad \text{for all } X \in \text{Const}(\Delta) \\
(23) & \quad \text{del}X \xrightarrow{x} \text{del}X \quad \text{for all } X \in \text{Const}(\Delta) \text{ and } x \in \text{Act}(\Delta)
\end{align*}
\]

Hence, we have strengthen the Mayr’s result [13], Theorem 29 (also reformulated as Theorem 3.2 of this paper) as follows.

**Theorem 4.2** Weak bisimilarity is undecidable for normed \(fc\text{BPA}\) systems.

## 5 Conclusion

First, we have shown that the weak bisimilarity problem remains undecidable for weakly extended versions of BPP (wBPP) and BPA (wBPA) process classes.

We note that the result for wBPA is just our interpretation (in terms of weakly extended systems) of Mayr’s proof showing that the problem is undecidable for PDA with two control states ([13], Theorem 29).

In terms of parallel systems, our technique used for wBPA is new. To mimic the computation of a Minsky 2-counter machine, one has to be able to maintain its state information – the label of a current instruction and the values of the counters \(c_1\) and \(c_2\). As a finite-state unit of wBPP is weak, it cannot be used to store even a part of such often changing information. Hence, contrary to the constructions in more expressive systems (PN [6] and MSA [16]) we have made a term part to manage it as follows. In a test-and-decrement instruction a process constant \(L_j\), which represents a label of the instruction, has to be changed and one of the counters \(c_1, c_2\) has to be decreased at the same time (assuming its value is positive). As two process
constants cannot be rewritten by one wBPP rewrite rule, we introduce new process constants $Y_1$ and $Y_2$ to represent “inverse elements” to $X_1$ and $X_2$ respectively and we make a term $X_k^x \parallel Y_k^y$ to represent the counter $c_k$ the value of which is $x - y$. We note that the weak state unit allows for controlling the correct order of the successive stages in the progress of a bisimulation game.

Moreover, we have shown that our undecidability results hold even for more restricted classes fcBPA and fcBPP and remain valid also for the normed versions of fcBPP and fcBPA. Hence, they hold for normed wBPP and normed wBPA as well.

We recall that the decidability of weak bisimilarity is an open question for BPA and BPP. Note that these problems are conjectured to be decidable (see [13] and [7] respectively) in which case our results would establish a fine undecidability border of weak bisimilarity.

**Acknowledgements.** We would like to thank Jiří Srba for valuable suggestions and comments.

**References**


