Deciding Bisimulation-Like Equivalences with Finite-State Processes*

Petr Jančar[1], Antonín Kučera[2], and Richard Mayr[3]

[1] Dept. of Computer Science FEI, Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava, Czech Republic, Petr.Jancar@vsb.cz
[2] Faculty of Informatics MU, Botanická 68a, 602 00 Brno, Czech Republic, tony@fi.muni.cz
[3] Institut für Informatik, Technische Universität München, Arcisstr. 21, D-80290 München, Germany, mayrri@informatik.tu-muenchen.de

Abstract. We design a general method for proving decidability of bisimulation-like equivalences between infinite-state processes and finite-state ones. We apply this method to the class of PAD processes, which strictly subsumes PA and pushdown (PDA) processes, showing that a large class of bisimulation-like equivalences (including e.g. strong and weak bisimilarity) is decidable between PAD and finite-state processes. On the other hand, we also demonstrate that no 'reasonable' bisimulation-like equivalence is decidable between state-extended PA processes and finite-state ones. Furthermore, weak bisimilarity with finite-state processes is shown to be undecidable even for state-extended BPP (which are also known as 'parallel pushdown processes').

1 Introduction

In this paper we study the decidability of bisimulation-like equivalences between infinite-state processes and finite-state ones. First we examine this problem in a general setting, extracting its core in a form of two rather special subproblems (which are naturally not decidable in general). A special variant of this method which works for strong bisimilarity has been described in [10]; here we extend and generalize the concept, obtaining a universal mechanism for proving decidability of bisimulation-like equivalences between infinite-state and finite-state processes. Then we apply the designed method to the class of PAD processes (defined in [16]), which properly subsumes all PA and pushdown processes. We prove that a large class of bisimulation-like equivalences (including e.g. strong and weak bisimilarity) is decidable between PAD and finite-state processes, utilizing previously established results on decidability of the model-checking problem for EF logic [15,17]. We also provide several undecidability results to complete the picture—we show that any 'reasonable' bisimulation-like equivalence is undecidable between state-extended PA processes and finite-state ones. Moreover, even for state-extended BPP processes (which are a natural subclass of Petri nets) weak bisimilarity with finite-state processes is undecidable.

* The first author is supported by the Grant Agency of the Czech Republic, grant No. 201/97/0456. The second author is supported by a Post-Doc grant GA ČR No. 201/98/P046 and by a Research Fellowship granted by The Alexander von Humboldt Foundation.

Decidability of bisimulation-like equivalences has been intensively studied for various process classes (see e.g. [19] for a complete survey). The majority of the results are about the decidability of strong bisimilarity, e.g. [3,6,5,22,4,13,8].

Strong bisimilarity with finite-state processes is known to be decidable for (labelled) Petri nets [12], PA and pushdown processes [10]. Another positive result of this kind is presented in [14], where it is shown that weak bisimilarity is decidable between BPP and finite-state processes. However, weak bisimilarity with finite-state processes is undecidable for Petri nets [9]. In [21] it is shown that the problem of equivalence-checking with finite-state systems can be reduced to the model-checking problem for the modal μ-calculus. Thus, in this paper we obtain original positive results for PAD (and hence also PA and PDA) processes, and an undecidability result for state-extended BPP processes. Moreover, all positive results are proved using the same general strategy, which can also be adapted to previously established ones.

2 Definitions

Transition systems are widely accepted as a structure which can exactly define the operational semantics of processes. In the rest of this paper we understand processes as (being associated with) nodes in transition systems of certain types.

Definition 1. *A transition system (TS) $\mathcal{T}$ is a triple $(S, Act, \rightarrow)$ where S is a set of* states*, Act is a finite set of* actions *(or* labels*) and $\rightarrow \subseteq S \times Act \times S$ is a* transition relation*.*

We defined *Act* as a finite set; this is a little bit nonstandard, but we can allow this as all classes of processes we consider generate transition systems of this type. As usual, we write $s \xrightarrow{a} t$ instead of $(s, a, t) \in \rightarrow$ and we extend this notation to elements of Act^* in an obvious way (we sometimes write $s \rightarrow^* t$ instead of $s \xrightarrow{w} t$ if $w \in Act^*$ is irrelevant). A state t is *reachable* from a state s if $s \rightarrow^* t$.

Let $Var = \{X, Y, Z, \ldots\}$ be a countably infinite set of *variables*. The class of *process expressions*, denoted $\mathcal{E}$, is defined by the following abstract syntax equation:

$$E ::= \lambda \mid X \mid E \| E \mid E.E$$

Here X ranges over *Var* and λ is a constant that denotes the empty expression. In the rest of this paper we do not distinguish between expressions related by *structural congruence* which is the smallest congruence relation over process expressions such that the following laws hold: associativity for '.' and '$\|$', commutativity for '$\|$', and 'λ' as a unit for '.' and '$\|$'.

A *process rewrite system* [16] is specified by a finite set Δ of *rules* which are of the form $E \xrightarrow{a} F$, where E, F are process expressions and a is an element of a finite set *Act*. Each process rewrite system determines a unique transition system where states are process expressions, *Act* is the set of labels, and transitions are defined by Δ and the following inference rules (remember that '$\|$' is commutative):

$$\frac{(E \stackrel{a}{\to} F) \in \Delta}{E \stackrel{a}{\to} F} \qquad \frac{E \stackrel{a}{\to} E'}{E.F \stackrel{a}{\to} E'.F} \qquad \frac{E \stackrel{a}{\to} E'}{E \| F \stackrel{a}{\to} E' \| F}$$

The classes of BPA, BPP, PA, and PAD systems are subclasses of process rewrite systems obtained by certain restrictions on the form of the expressions which can appear at the left-hand and the right-hand side of rules. To specify those restrictions, we first define the classes of *sequential* and *parallel* expressions, composed of all process expressions which do not contain the '$\|$' and the '.' operator, respectively. BPA, BPP, and PA allow only a single variable at the left-hand side of rules, and a sequential, parallel, and general process expression at the right-hand side, respectively. Note that each transition $E \stackrel{a}{\to} F$ is due to some rule $X \stackrel{a}{\to} G$ of Δ (i.e. X is rewritten by G within E, yielding the expression F). Generally, there can be more than one rule of Δ with this property—if e.g. $\Delta = \{X \stackrel{a}{\to} X \| Y, Y \stackrel{a}{\to} Y \| Y\}$, then the transition $X \| Y \stackrel{a}{\to} X \| Y \| Y$ can be derived in one step in two different ways. For each transition $E \stackrel{a}{\to} F$ we denote the set of all rules of Δ which allow to derive the transition in one step by $\mathit{Step}(E \stackrel{a}{\to} F)$.

The PA class strictly subsumes BPA and BPP systems; a proper extension of PA is the class of PAD systems (see [16]), where sequential expressions are allowed at the left-hand side and general ones at the right-hand side of rules. The PAD class strictly subsumes not only PA but also PDA processes (see below). This is demonstrated in [16].

Another way how to extend a PA system is to add a finite-state control unit to it. A *state-extended PA system* is a triple (Δ, Q, BT) where Δ is a PA system, Q is a finite set of *states*, and $\mathit{BT} \subseteq \Delta \times Q \times Q$ is a set of *basic transitions*. The transition system generated by a state-extended PA system (Δ, Q, BT) has $Q \times \mathcal{E}$ as the set of states (its elements are called *state-extended PA processes*, or *StExt(PA)* processes for short), *Act* is the set of labels, and the transition relation is determined by

$$(p, E) \stackrel{a}{\to} (q, F) \text{ iff } E \stackrel{a}{\to} F \text{ and } (X \stackrel{a}{\to} G, p, q) \in \mathit{BT} \text{ for some } X \stackrel{a}{\to} G \in \mathit{Step}(E \stackrel{a}{\to} F)$$

Natural subclasses of *StExt(PA)* systems are *StExt(BPA)* and *StExt(BPP)*, which are also known as pushdown (PDA) and parallel pushdown (PPDA) systems, respectively. Each *StExt(BPA)* system can also be seen as a PAD system; however, the classes of *StExt(BPP)* and PAD systems are semantically incomparable (w.r.t. *strong bisimilarity*, which is defined in the next section—see also [16]).

3 A General Method for Bisimulation-Like Equivalences

In this section we design a general method for proving decidability of bisimulation-like equivalences between infinite-state processes and finite-state ones.

Definition 2. *Let $R : \mathit{Act} \to 2^{\mathit{Act}^*}$ be a (total) function, assigning to each action its corresponding set of* responses*. We say that R is* closed under substitution *if the following conditions hold:*

- $a \in R(a)$ *for each* $a \in \mathit{Act}$
- *If* $b_1 b_2 \ldots b_n \in R(a)$ *and* $w_1 \in R(b_1), w_2 \in R(b_2), \ldots, w_n \in R(b_n)$*, then also* $w_1 w_2 \ldots w_n \in R(a)$.

In order to simplify our notation, we adopt the following conventions in this section:

– $\mathcal{G} = (G, Act, \rightarrow)$ always denotes a (general) transition system.
– $\mathcal{F} = (F, Act, \rightarrow)$ always denotes a finite-state transition system with k states.
– R always denotes a function from *Act* to 2^{Act^*} which is closed under substitution.
– N always denotes a decidable binary predicate defined for pairs (s, t) of nodes in transition systems (which will be clear from the context). Moreover, N is reflexive, symmetric, and transitive.
– We write $s \stackrel{a}{\Rightarrow} t$ if $s \stackrel{w}{\rightarrow} t$ for some $w \in R(a)$.

Note that $\mathcal{G}$ and $\mathcal{F}$ have the same set of actions *Act*. All definitions and propositions which are formulated for $\mathcal{G}$ should be considered as general; if we want to state some specific property of finite-state transition systems, we refer to $\mathcal{F}$. We also assume that $\mathcal{G}$, $\mathcal{F}$, R, and N are defined in a 'reasonable' way so that we can allow natural decidability assumptions on them (e.g. it is decidable whether $g \stackrel{a}{\rightarrow} g'$ for any given $g, g' \in G$ and $a \in Act$, or whether $w \in R(a)$ for a given $w \in Act^*$, etc.)

Definition 3. *A relation $P \subseteq G \times G$ is an* R-N-bisimulation *if whenever $(s, t) \in P$, then $N(s, t)$ is true and for each $a \in Act$:*

– *If $s \stackrel{a}{\rightarrow} s'$, then $t \stackrel{a}{\Rightarrow} t'$ for some $t' \in G$ such that $(s', t') \in P$.*
– *If $t \stackrel{a}{\rightarrow} t'$, then $s \stackrel{a}{\Rightarrow} s'$ for some $s' \in G$ such that $(s', t') \in P$.*

States $s, t \in G$ are R-N-bisimilar*, written $s \stackrel{RN}{\sim} t$, if there is an R-N-bisimulation relating them.*

Various special versions of *R-N*-bisimilarity appeared in the literature, e.g. *strong* and *weak* bisimilarity (see [20,18]). The corresponding versions of R (denoted by S and W, respectively) are defined as follows:

– $S(a) = \{a\}$ for each $a \in Act$
– $W(a) = \begin{cases} \{\tau^i \mid i \in \mathbb{N}_0\} & \text{if } a = \tau \\ \{\tau^i a \tau^j \mid i, j \in \mathbb{N}_0\} & \text{otherwise} \end{cases}$

The 'τ' is a special (silent) action, usually used to model an internal communication. As the predicate N is not employed in the definitions of strong and weak bisimilarity, we can assume it is always true (we use T to denote this special case of N).

The concept of *R-N*-bisimilarity covers many equivalences, which have not been explicitly investigated so far; for example, we can define the function R like this:

– $K(a) = \{a^i \mid i \in \mathbb{N}_0\}$ for each $a \in Act$.
– $L(a) = \{w \in Act^* \mid w \text{ begins with } a\}$.
– $M(a) = \begin{cases} Act^* & \text{if } a = \tau \\ \{w \in Act^* \mid w \text{ contains at least one } a\} & \text{otherwise} \end{cases}$

The predicate N can also have various forms. We have already mentioned the 'T' (always true). Another natural example is the I predicate: $I(s, t)$ is true iff s and t have the same sets of initial actions (the set of initial actions of a state $g \in G$ is $\{a \in Act \mid g \stackrel{a}{\rightarrow} g' \text{ for some } g' \in G\}$). It is easy to see that e.g. $\stackrel{ST}{\sim}$ coincides with $\stackrel{SI}{\sim}$, while $\stackrel{WI}{\sim}$ refines $\stackrel{WT}{\sim}$.

To the best of our knowledge, the only bisimulation-like equivalence which cannot be seen as *R-N*-bisimilarity is *branching bisimilarity* introduced in [23]. This relation also places requirements on 'intermediate' nodes that extended transitions pass through,

and this brings further difficulties. Therefore we do not consider branching bisimilarity in our paper.

R-N-bisimilarity can also be defined in terms of the so-called *R-N-bisimulation game*. Imagine that there are two tokens initially placed in states s and t such that $N(s,t)$ is true. Two players, Al and Ex, now start to play a game consisting of a (possibly infinite) sequence of rounds, where each round is performed as follows:

1. Al chooses one of the two tokens and moves it along an arbitrary (but single!) transition, labelled by some $a \in Act$.
2. Ex has to respond by moving the other token along a finite sequence of transitions in such a way that the corresponding sequence of labels belongs to $R(a)$ and the predicate N is true for the states where the tokens lie after Ex finishes his move.

Al wins the *R-N*-bisimulation game, if after a finite number of rounds Ex cannot respond to Al's final attack. Now it is easy to see that the states s and t are *R-N*-bisimilar iff Ex has a universal defending strategy (i.e. Ex can play in such a way that Al cannot win).

A natural way how to approximate *R-N*-bisimilarity is to define the family of relations $\overset{RN}{\sim}_i \subseteq G \times G$ for each $i \in \mathbb{N}_\mathbf{0}$ as follows: $s \overset{RN}{\sim}_i t$ iff $N(s,t)$ is true and Ex has a defending strategy within the first i rounds in the *R-N*-bisimulation game. However, $\overset{RN}{\sim}_i$ does not have to be an equivalence relation. Moreover, it is not necessarily true that $s \overset{RN}{\sim} t \iff s \overset{RN}{\sim}_i t$ for each $i \in \mathbb{N}_\mathbf{0}$. A simple counterexample is the weak bisimilarity (i.e. *W-T*-bisimilarity) and its approximations.

Now we show how to overcome those drawbacks; to do this, we introduce the *extended R-N*-bisimulation relation:

Definition 4. *A relation $P \subseteq G \times G$ is an* extended *R-N-bisimulation if whenever $(s,t) \in P$, then $N(s,t)$ is true and for each $a \in Act$:*

- *If $s \overset{a}{\Rightarrow} s'$, then $t \overset{a}{\Rightarrow} t'$ for some $t' \in G$ such that $(s',t') \in P$.*
- *If $t \overset{a}{\Rightarrow} t'$, then $s \overset{a}{\Rightarrow} s'$ for some $s' \in G$ such that $(s',t') \in P$.*

States $s,t \in G$ are extended R-N-bisimilar *if there is an extended R-N-bisimulation relating them.*

Naturally, we can also define the extended *R-N*-bisimilarity by means of the extended *R-N*-bisimulation game; we simply allow Al to use the 'long' moves (i.e. Al can play the same kind of moves as Ex). Moreover, we can define the family of approximations of extended *R-N*-bisimilarity in the same way as in case of *R-N*-bisimilarity—for each $i \in \mathbb{N}_\mathbf{0}$ we define the relation $\overset{RN}{\simeq}_i \subseteq G \times G$ as follows: $s \overset{RN}{\simeq}_i t$ iff $N(s,t)$ is true and Ex has a defending strategy within the first i rounds in the extended *R-N*-bisimulation game where tokens are initially placed in s and t.

Lemma 1. *Two states s,t of $\mathcal{G}$ are R-N-bisimilar iff s and t are extended R-N-bisimilar.*

Lemma 2. *The following properties hold:*

1. *$\overset{RN}{\simeq}_i$ is an equivalence relation for each $i \in \mathbb{N}_\mathbf{0}$.*
2. *Let s,t be states of $\mathcal{G}$. Then $s \overset{RN}{\sim}_i t$ for each $i \in \mathbb{N}_\mathbf{0}$ iff $s \overset{RN}{\simeq}_i t$ for each $i \in \mathbb{N}_\mathbf{0}$.*

Now we examine some special features of *R-N*-bisimilarity on finite-state transition systems (remember that $\mathcal{F}$ is a finite-state TS with k states).

Lemma 3. *Two states s, t of $\mathcal{F}$ are R-N-bisimilar iff $s \overset{RN}{\simeq}_{k-1} t$.*

Proof. As $\mathcal{F}$ has k states and $\overset{RN}{\simeq}_{i+1}$ refines $\overset{RN}{\simeq}_i$ for each $i \in \mathbb{N}_0$, we have that $\overset{RN}{\simeq}_{k-1} = \overset{RN}{\simeq}_k$, hence $\overset{RN}{\simeq}_{k-1} = \overset{RN}{\sim}$.

Theorem 1. *States $g \in G$, $f \in F$ are R-N-bisimilar iff $g \overset{RN}{\simeq}_k f$ and for each state g' reachable from g there is a state $f' \in F$ such that $g' \overset{RN}{\simeq}_k f'$.*

Proof.
'$\Longrightarrow$': Obvious.
'$\Longleftarrow$': We prove that the relation $P = \{(g', f') \mid g \rightarrow^* g' \text{ and } g' \overset{RN}{\simeq}_k f'\}$ is an extended *R-N*-bisimulation. Let $(g', f') \in P$ and let $g' \overset{a}{\Rightarrow} g''$ for some $a \in Act$ (the case when $f' \overset{a}{\Rightarrow} f''$ is handled is the same way). By definition of $\overset{RN}{\simeq}_k$, there is f'' such that $f' \overset{a}{\Rightarrow} f''$ and $g'' \overset{RN}{\simeq}_{k-1} f''$. It suffices to show that $g'' \overset{RN}{\simeq}_k f''$; as $g \rightarrow^* g''$, there is a state $\overline{f}$ of $\mathcal{F}$ such that $g'' \overset{RN}{\simeq}_k \overline{f}$. By transitivity of $\overset{RN}{\simeq}_{k-1}$ we have $\overline{f} \overset{RN}{\simeq}_{k-1} f''$, hence $\overline{f} \overset{RN}{\simeq}_k f''$ (due to Lemma 3). Now $g'' \overset{RN}{\simeq}_k \overline{f} \overset{RN}{\simeq}_k f''$ and thus $g'' \overset{RN}{\simeq}_k f''$ as required. Clearly $(g, f) \in P$ and the proof is complete. □

Remark 1. We have already mentioned that the equivalence $s \overset{RN}{\sim} t \Longleftrightarrow s \overset{RN}{\simeq}_i t$ *for each* $i \in \mathbb{N}_0$ is generally invalid (e.g. in case of weak bisimilarity). However, as soon as we assume that t is a state in a finite-state transition system, the equivalence becomes true. This is an immediate consequence of the previous theorem. Moreover, the second part of Lemma 2 says that we could also use the $\overset{RN}{\sim}_i$ approximations in the right-hand side of the equivalence.

The previous theorem in fact says that one can use the following strategy to decide whether $g \overset{RN}{\sim} f$:

1. Decide whether $g \overset{RN}{\simeq}_k f$ (if not, then $g \overset{RN}{\not\sim} f$).
2. Check whether g can reach a state g' such that $g' \overset{RN}{\not\simeq}_k f'$ for *any* state f' of $\mathcal{F}$ (if there is such a g' then $g \overset{RN}{\not\sim} f$; otherwise $g \overset{RN}{\sim} f$).

However, none of these tasks is easy in general. Our aim is to examine both subproblems in detail, keeping the general setting. Thus we cannot expect any 'universal' (semi)decidability result, because even the problems $g \overset{WT}{\simeq}_1 f$ and $g \overset{WT}{\not\simeq}_1 f$ are not semidecidable in general (see Section 5).

As $\mathcal{F}$ has finitely many states, the extended transition relation $\Rightarrow$ is finite and effectively constructible. This allows us to "extract" from $\mathcal{F}$ the information which is relevant for the first k moves in the extended *R-N*-bisimulation game by means of branching trees with depth at most k, whose arcs are labelled by elements of *Act* and nodes are labelled by elements of $F \cup \{\bot\}$, where $\bot \notin F$. The aim of following definition is to describe all such trees up to isomorphism (remember that *Act* is a *finite* set).

Definition 5. *For each $i \in \mathbb{N}_0$ we define the set of* Trees *with* depth *at most i (denoted Tree$_i$) inductively as follows:*

– *A Tree with depth* 0 *is any tree with no arcs and a single node (the root) which is labelled by an element of $F \cup \{\bot\}$.*

– *A Tree with depth at most $i+1$ is any directed tree with root r whose nodes are labelled by elements of $F \cup \{\bot\}$, arcs are labelled by elements of Act, which satisfies the following conditions:*
  – *If $r \xrightarrow{a} s$, then the subtree rooted by s is a Tree with depth at most i.*
  – *If $r \xrightarrow{a} s$ and $r \xrightarrow{a} s'$, then the subtrees rooted by s and s' are not isomorphic.*

It is clear that the set *Tree*$_j$ is finite and effectively constructible for any $j \in \mathbb{N}_{\mathbf{0}}$. As each Tree can be seen as a transition system, we can also speak about *Tree-processes* which are associated with roots of Trees (we do not distinguish between Trees and Tree-processes in the rest of this paper).

Now we introduce special rules which replace the standard ones whenever we consider an extended *R-N*-bisimulation game with initial state (g, p), where $g \in G$ and p is a Tree process (formally, these rules determine is a new (different) game—however, it does not deserve a special name in our opinion).

– Al and Ex are allowed to play only 'short' moves consisting of exactly one transition whenever playing within the Tree process p (transitions of Trees correspond to extended transitions of $\mathcal{F}$).
– The predicate $N(g', p')$, where $g' \in G$ and p' a state of the Tree process p, is evaluated as follows:
  – if *label*$(p') \neq \bot$, then $N(g', p') = N(g', \mathit{label}(p'))$
  – if *label*$(p') = \bot$ and $N(g', f) =$ *true* for some $f \in F$, then $N(g', p') =$ *false*
  – if *label*$(p') = \bot$ and $N(g', f) =$ *false* for any $f \in F$, then $N(g', p') =$ *true*

Whenever we write $g \overset{RN}{\simeq}_i p$, where $g \in G$ and p is a Tree process, we mean that Ex has a defending strategy within the first i rounds in the 'modified' extended *R-N*-bisimulation game. The importance of Tree processes is clarified by the two lemmas below:

Lemma 4. *Let g be a state of $\mathcal{G}$, $j \in \mathbb{N}_{\mathbf{0}}$. Then $g \overset{RN}{\simeq}_j p$ for some $p \in$ Tree$_j$*

Lemma 5. *Let f be a state of $\mathcal{F}$, $j \in \mathbb{N}_{\mathbf{0}}$, and $p \in$ Tree$_j$ such that $f \overset{RN}{\simeq}_j p$. Then for any state g of $\mathcal{G}$ we have that $g \overset{RN}{\simeq}_j f$ iff $g \overset{RN}{\simeq}_j p$.*

Now we can extract the core of both subproblems which appeared in the previously mentioned general strategy in a (hopefully) nice way by defining two new and rather special problems—the Step-problem and the Reach-problem:

The Step-problem
Instance: (g, a, j, p) where g is a state of $\mathcal{G}$, $a \in$ *Act*, $0 \leq j < k$, and $p \in$ *Tree*$_j$.
Question: Is there a state g' of $\mathcal{G}$ such that $g \overset{a}{\Rightarrow} g'$ and $g' \overset{RN}{\simeq}_j p$?

The oracle which for any state g'' of $\mathcal{G}$ answers whether $g'' \overset{RN}{\simeq}_j p$ can be used.

The Reach-problem
Instance: (g, p) where g is a state of $\mathcal{G}$ and p is a Tree-process of depth $\leq k$.
Question: Is there a state g' of $\mathcal{G}$ such that $g \rightarrow^* g'$ and $g' \overset{RN}{\simeq}_k p$?

The oracle which for any state g'' of $\mathcal{G}$ answers whether $g'' \overset{RN}{\simeq}_k p$ can be used.

Formally, the transition system $\mathcal{F}$ should also be present in instances of both problems, as it determines the sets Tree_j and the constant k; we prefer the simplified form to make the following proofs more readable.

Theorem 2. *If the Step-problem is decidable (with possible usage of the mentioned oracle), then $\overset{RN}{\simeq}_k$ is decidable between any states g and f of $\mathcal{G}$ and $\mathcal{F}$, respectively.*

Proof. We prove by induction on j that $\overset{RN}{\simeq}_j$ is decidable for any $0 \leq j \leq k$. First, $\overset{RN}{\simeq}_0$ is decidable because the predicate N is decidable. Let us assume that $\overset{RN}{\simeq}_j$ is decidable (hence the mentioned oracle can be used). It remains to prove that if the Step-problem is decidable, then $\overset{RN}{\simeq}_{j+1}$ is decidable as well. We introduce two auxiliary finite sets:

- The set of **C**ompatible **S**teps, denoted CS^f_j, is composed exactly of all pairs of the form (a, p) where $a \in \mathit{Act}$ and $p \in \mathit{Tree}_j$, such that $f \overset{a}{\Rightarrow} f'$ for some f' with $f' \overset{RN}{\simeq}_j p$.
- The set of **INC**ompatible **S**teps, denoted INCS^f_j, is a complement of CS^f_j w.r.t. $\mathit{Act} \times \mathit{Tree}_j$.

The sets CS^f_j and INCS^f_j are effectively constructible. By definition, $g \overset{RN}{\simeq}_{j+1} f$ iff $N(g, f)$ is true and the following conditions hold:

1. If $f \overset{a}{\Rightarrow} f'$, then $g \overset{a}{\Rightarrow} g'$ for some g' with $g' \overset{RN}{\simeq}_j f'$.
2. If $g \overset{a}{\Rightarrow} g'$, then $f \overset{a}{\Rightarrow} f'$ for some f' with $g' \overset{RN}{\simeq}_j f'$.

The first condition in fact says that (g, a, j, p) is a positive instance of the Step-problem for any $(a, p) \in \mathit{CS}^f_j$ (see Lemma 4 and 5). It can be checked effectively due to the decidability of the Step-problem.

The second condition does *not* hold iff $g \overset{a}{\Rightarrow} g'$ for some g' such that $g' \overset{RN}{\simeq}_j p$ where (a, p) is an element of INCS^f_j (due to Lemma 4 and 5). This is clearly decidable due to the decidability of the Step-problem again. □

It is worth mentioning that the Step-problem is generally semidecidable (provided it is possible to enumerate all finite paths starting in g). However, it does not suffice for semidecidability of $\overset{RN}{\simeq}_i$ or $\overset{RN}{\not\simeq}_i$ between states of $\mathcal{G}$ and $\mathcal{F}$.

Theorem 3. *Decidability of the Step-problem and the Reach-problem (with possible usage of the indicated oracles) implies decidability of the problem whether for each g' reachable from a given state g of $\mathcal{G}$ there is a state f' of $\mathcal{F}$ with $g' \overset{RN}{\simeq}_k f'$.*

Proof. First, the oracle indicated in the definition of Reach-problem can be used because we already know that decidability of the Step-problem implies decidability of $\overset{RN}{\simeq}_k$ between states of $\mathcal{G}$ and $\mathcal{F}$ (see the previous theorem). To complete the proof, we need to define one auxiliary set:

- The set of **INC**ompatible **T**rees, denoted INCT, is composed of all $p \in \mathit{Tree}_k$ such that $f \overset{RN}{\not\simeq}_k p$ for each state f of $\mathcal{F}$.

The set *INCT* is finite and effectively constructible. The state g can reach a state g' such that $g' \overset{RN}{\not\simeq}_k f$ for any state f of $\mathcal{F}$ (i.e. g is a *negative* instance of the problem specified in the second part of this theorem) iff (g, p) is a positive instance of the Reach problem for some $p \in$ *INCT* (due to Lemma 4 and 5). □

4 Applications

In this section we show that the Step and Reach problems can be reduced to the model checking problem for the branching-time temporal logic EF. In this way we elegantly prove that a large class of *R-N*-bisimulation equivalences is decidable between PAD processes and finite-state ones (the class includes all versions of *R-N*-bisimulation equivalences we defined in this paper and many others). First we define the logic EF (more exactly an extended version of EF with constraints on sequences of actions). The formulae have the following syntax:

$$\Phi ::= \mathit{true} \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \langle a \rangle \Phi \mid \Diamond_C \Phi$$

where a is an atomic action and C is a unary predicate on sequences of atomic actions. Let $\mathcal{T} = (S, \mathit{Act}, \rightarrow)$ be a transition system. The denotation $[\![\Phi]\!]$ of a formula Φ is a set of states of $\mathcal{T}$, which is defined as follows (sequences of actions are denoted by w):

$$[\![\mathit{true}]\!] := S,\ [\![\neg\Phi]\!] := S - [\![\Phi]\!],\ [\![\Phi_1 \wedge \Phi_2]\!] := [\![\Phi_1]\!] \cap [\![\Phi_2]\!]$$
$$[\![\langle a \rangle \Phi]\!] := \{s \in S \mid \exists s' \in S.\ s \xrightarrow{a} s' \in [\![\Phi]\!]\}$$
$$[\![\Diamond_C \Phi]\!] := \{s \in S \mid \exists w, s'.\ s \xrightarrow{w} s' \wedge C(w) \wedge s' \in [\![\Phi]\!]\}$$

The predicates C are used to express constraints on sequences of actions. For every *R-N*-bisimulation we define predicates C_a s.t. for every action a and every sequence w we have $C_a(w) \iff w \in R(a)$. Let EF_R be the fragment of EF that contains only constraints C_a for R and the *true* constraint.

An instance of the *model checking problem* is given by a state s in S and an EF_R-formula Φ. The question is whether $s \in [\![\Phi]\!]$. This property is also denoted by $s \models \Phi$.

Let us fix a general TS $\mathcal{G} = (G, \mathit{Act}, \rightarrow)$ and a finite-state TS $\mathcal{F} = (F, \mathit{Act}, \rightarrow)$ with k states in the same way as in the previous section. We show how to encode the Step and the Reach problems by EF_R formulae. The first difficulty is the N predicate. Although it is decidable, this fact is generally of no use as we do not know anything about the strategy of the model-checking algorithm. Instead, we restrict our attention to those predicates which can be encoded by EF_R formulae in the following sense: for each $f \in F$ there is an EF_R formula Ψ_f such that for each $g \in G$ we have that $g \models \Psi_f$ iff $N(g, f)$ is true. In this case we also define the formula $\Psi_\perp := \bigwedge_{f \in F} \neg\Psi_f$. A concrete example of a predicate which can be encoded by EF_R formulae is e.g. the 'I' predicate defined in the previous section. Now we design the family of $\Phi_{j,p}$ formulae, where $0 \leq j \leq k$ and $p \in \mathit{Tree}_j$, in such a way that for each $g \in G$ the equivalence $g \overset{RN}{\simeq}_j p \iff g \models \Phi_{j,p}$ holds. Having these formulae, the Step and the Reach problems can be encoded in a rather straightforward way:

- (g, a, j, p) is a positive instance of the Step problem iff $g \models \Diamond_{C_a}(\Phi_{j,p})$
- (g, p) is a positive instance of the Reach problem iff $g \models \Diamond(\Phi_{k,p})$

The family of $\Phi_{j,p}$ formulae is defined inductively on j as follows:

– $\Phi_{0,p} := \Psi_f$, where $f = \mathit{label}(p)$

– $\Phi_{j+1,p} := \Psi_f \wedge \left(\bigwedge_{a \in Act} \bigwedge_{p' \in S(p,a)} \Diamond_{C_a} \Phi_{j,p'} \right) \wedge \left(\bigwedge_{a \in Act} (\neg \Diamond_{C_a} (\bigwedge_{p' \in S(p,a)} \neg \Phi_{j,p'})) \right)$,

where $f = \mathit{label}(p)$ and $S(p,a) = \{p' \mid p \xrightarrow{a} p'\}$. If the set $S(p,a)$ is empty, any conjunction of the form $\bigwedge_{p' \in S(p,a)} \Theta_{p'}$ is replaced by *true*.

The decidability of model checking with the logic EF_R depends on the constraints that correspond to R. It has been shown in [15] that model checking PA-processes with the logic EF is decidable for the class of *decomposable constraints*. This result has been generalized to PAD processes in [17]. These constraints are called decomposable, because they can be decomposed w.r.t. sequential and parallel composition. The formal definition is as follows: A set of decomposable constraints $\mathcal{DC}$ is a finite set of unary predicates on finite sequences of actions that contains the predicates $true$ and $false$ and satisfies the following conditions.

1. For every $C \in \mathcal{DC}$ there is a finite index set I and a finite set of decomposable constraints $\{C_i^1, C_i^2 \in \mathcal{DC} \mid i \in I\}$ s.t.
$$\forall w, w_1, w_2.\ w_1 w_2 = w \ \Rightarrow\ (C(w) \iff \bigvee_{i \in I} C_i^1(w_1) \wedge C_i^2(w_2))$$
2. For every $C \in \mathcal{DC}$ there is a finite index set J and a finite set of decomposable constraints $\{C_i^1, C_i^2 \in \mathcal{DC} \mid i \in J\}$ s.t.
$$\forall w_1, w_2.(\ (\exists w \in \mathit{interleave}(w_1, w_2).\, C(w)) \iff \bigvee_{i \in J} (C_i^1(w_1) \wedge C_i^2(w_2)))$$
Here $w \in \mathit{interleave}(w_1, w_2)$ iff w is an arbitrary interleaving of w_1 and w_2.

It is easy to see that the closure of a set of decomposable constraints under disjunction is again a set of decomposable constraints. All the previously mentioned examples of functions R can be expressed by decomposable constraints. However, there are also functions R that are closed under substitution, but which yield non-decomposable constraints. For example, let $Act = \{a, b\}$ and $R(a) := \{w \mid \#_a w > \#_b w\}$ and $R(b) := \{b\}$, where $\#_a w$ is the number of actions a in w. On the other hand, there are decomposable constraints that are not closed under substitution like $R(a) := \{a^i \mid 1 \leq i \leq 5\}$. Now we can formulate a very general decidability theorem:

Theorem 4. *The problem $g \overset{RN}{\sim} f$, where R yields a set of constraints contained in a set $\mathcal{DC}$ of decomposable constraints, N is expressible in EF_R, g is a PAD processes, and f is a finite-state process, is decidable.*

5 Undecidability Results

Intuitively, any 'nontrivial' equivalence with finite-state processes should be undecidable for a class of processes having 'full Turing power', which can be formally expressed as e.g. the ability to simulate Minsky counter machines. Any such machine $\mathcal{M}$ can be easily 'mimicked' by a *StExt(PA)* process $P(\mathcal{M})$. A construction of the $P(\mathcal{M})$ process is described in [10]. If we label each transition in $P(\mathcal{M})$ by an action a then it is can either perform the action a boundedly many times and stop (its behaviour can be defined as a^n for some n) or do a forever (its behaviour being a^ω); this depends

on whether the corresponding counter machine $\mathcal{M}$ halts or not. Notice that a^ω is the behaviour of the 1-state transition system $(\{s\}, \{a\}, \{(s, a, s)\})$. When we declare as *reasonable* any equivalence which distinguishes between (processes with) behaviours a^ω and a^n, we can conclude:

Theorem 5. *Any reasonable equivalence between StExt(PA) processes and finite-state ones is undecidable.*

It is obvious that (almost) any *R-N*-bisimilarity is reasonable in the above sense, except for some trivial cases. For weak bisimilarity, we can even show that none of the problems $g \overset{WT}{\simeq}_1 f$, $g \overset{WT}{\not\simeq}_1 f$ is semidecidable when g is a *StExt(PA)* process.

Once seeing that *StExt(PA)* are strong enough to make our equivalences undecidable, it is natural to ask what happens when we add finite-state control parts to processes from subclasses of PA, namely to BPA and BPP. The *StExt(BPA)* (i.e. PDA) processes have been examined in the previous section. In the case of *StExt(BPP)*, strong bisimilarity with finite-state processes is decidable [12]. Here we demonstrate that the problem for weak bisimilarity is undecidable; the proof is obtained by a modification of the one which has been used for labelled Petri nets in [9].

It can be easily shown that a labelled Petri net where each transition t has exactly one input place is equivalent to a BPP process (the corresponding transition systems are isomorphic)—see e.g. [7]. Similarly, if any transition has at most one unbounded place among its input places, then it is easy to transform the net into an equivalent *StExt(BPP)* process (the marking of bounded places is modelled by finite control states); let us call such nets as *StExt(BPP)*-nets.

The idea of the mentioned construction from [9] looks as follows. First, a 7-state transition system $\mathcal{F}$ is fixed. Then it is shown how to construct a net $N_\mathcal{M}$ for any two-counter machine $\mathcal{M}$ such that $N_\mathcal{M}$ is weakly bisimilar to $\mathcal{F}$ iff $\mathcal{M}$ does not halt for zero input. Therefore, if the net $N_\mathcal{M}$ were always a *StExt(BPP)*-net, we would be done. In fact, it is not the case but $N_\mathcal{M}$ can be suitably transformed. The description of the transformation is omitted due to the lack of space; it can be found in [11]. Now we can conclude:

Theorem 6. *Weak bisimilarity is undecidable between StExt(BPP) processes and finite-state ones.*

6 Conclusions, Future Work

A complete summary of the results on decidability of bisimulation-like equivalences with finite-state processes is given in the table below. As we want to make clear what results have been previously obtained by other researchers, our table contains more columns than it is necessarily needed (e.g., the positive result for PAD and $\overset{RN}{\sim}$, where R and N have the above indicated properties, 'covers' all positive results for BPA, BPP, PA, and PDA). The results obtained in this paper are in boldface. We also add a special row which indicates decidability of the model-checking problem for EF. Note that although model-checking EF logic is undecidable for *StExt(BPP)* processes and Petri nets, strong bisimilarity with finite-state systems is decidable. The original proof

in [12] in fact demonstrates decidability of the Reach problem (the Step problem is trivially decidable), hence our general strategy applies also in this case.

	BPA	BPP	PA	*StExt(BPA)*	*StExt(BPP)*	*StExt(PA)*	PAD	PN
$\overset{ST}{\sim}$	Yes [6]	Yes [5]	Yes [10]	Yes [10]	Yes [12]	No [10]	**YES**	Yes [12]
$\overset{WT}{\sim}$	**YES**	Yes [14]	**YES**	**YES**	**NO**	No [10]	**YES**	No [9]
$\overset{RN}{\sim}$	**YES**	**YES**	**YES**	**YES**	**NO**	No [10]	**YES**	No [9]
EF	Yes	Yes	Yes	Yes	No	No	Yes	No

References

1. *Proceedings of CONCUR'96*, volume 1119 of *LNCS*. Springer-Verlag, 1996.
2. *Proceedings of CONCUR'97*, volume 1243 of *LNCS*. Springer-Verlag, 1997.
3. J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. *JACM*, 40:653–682, 1993.
4. I. Černá, M. Křetínský, and A. Kučera. Bisimilarity is decidable in the union of normed BPA and normed BPP processes. *ENTCS*, 6, 1997.
5. S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation is decidable for all basic parallel processes. In *Proceedings of CONCUR'93*, volume 715 of *LNCS*, pages 143–157. Springer-Verlag, 1993.
6. S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, 121:143–148, 1995.
7. J. Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. In *Proceedings of FCT'95*, volume 965 of *LNCS*, pages 221–232. Springer-Verlag, 1995.
8. P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301, 1995.
9. P. Jančar and J. Esparza. Deciding finiteness of Petri nets up to bisimilarity. In *Proceedings of ICALP'96*, volume 1099 of *LNCS*, pages 478–489. Springer-Verlag, 1996.
10. P. Jančar and A. Kučera. Bisimilarity of processes with finite-state systems. *ENTCS*, 9, 1997.
11. P. Jančar, A. Kučera, and R. Mayr. Deciding bisimulation-like equivalences with finite-state processes. Technical report TUM-I9805, Technische Universität München, 1998.
12. P. Jančar and F. Moller. Checking regular properties of Petri nets. In *Proceedings of CONCUR'95*, volume 962 of *LNCS*, pages 348–362. Springer-Verlag, 1995.
13. A. Kučera. How to parallelize sequential processes. In *Proceedings of CONCUR'97* [2], pages 302–316.
14. R. Mayr. Weak bisimulation and model checking for basic parallel processes. In *Proceedings of FST&TCS'96*, volume 1180 of *LNCS*, pages 88–99. Springer-Verlag, 1996.
15. R. Mayr. Model checking PA-processes. In *Proceedings of CONCUR'97* [2], pages 332–346.
16. R. Mayr. Process rewrite systems. *ENTCS*, 7, 1997.
17. R. Mayr. *Decidability and Complexity of Model Checking Problems for Infinite-State Systems*. PhD thesis, TU-München, 1998.
18. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
19. F. Moller. Infinite results. In *Proceedings of CONCUR'96* [1], pages 195–216.
20. D.M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings* 5^{th} *GI Conference*, volume 104 of *LNCS*, pages 167–183. Springer-Verlag, 1981.
21. B. Steffen and A. Ingólfsdóttir. Characteristic formulae for processes with divergence. *Information and Computation*, 110(1):149–163, 1994.
22. C. Stirling. Decidability of bisimulation equivalence for normed pushdown processes. In *Proceedings of CONCUR'96* [1], pages 217–232.
23. R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Information Processing Letters*, 89:613–618, 1989.