Deciding Bisimulation-Like Equivalences with Finite-State Processes

Petr Jančar [a,1], Antonín Kučera [b,2], Richard Mayr [c]

[a] *Dept. of Computer Science, FEI, Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava, Czech Republic. E-mail:* `Petr.Jancar@vsb.cz`

[b] *Faculty of Informatics, Masaryk University, Botanická 68a, 60200 Brno, Czech Republic. E-mail:* `tony@fi.muni.cz`

[c] *Institut für Informatik, Technische Universität München, Arcisstr. 21, D-80290 München, Germany. E-mail:* `mayrri@informatik.tu-muenchen.de`

Abstract

We show that characteristic formulae for finite-state systems up to bisimulation-like equivalences (e.g., strong and weak bisimilarity) can be given in the simple branching-time temporal logic *EF*. Since *EF* is a very weak fragment of the modal μ-calculus, model checking with *EF* is decidable for many more classes of infinite-state systems. This yields 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').

Key words: concurrency, bisimulation, characteristic formulae, infinite-state systems

[1] Supported by the Grant Agency of the Czech Republic, grant number 201/97/0456.

[2] Supported by a Research Fellowship granted by the Alexander von Humboldt Foundation and by a Post-Doc grant GA ČR No. 201/98/P046.

1 Introduction

We study the decidability of bisimulation-like equivalences between infinite-state processes and finite-state ones. The motivation is that the intended behavior of a process is often easy to specify (by a finite-state system), but a 'real' implementation can contain components which are essentially infinite-state (e.g., counters, buffers). The aim of formal verification is to check if the finite-state specification and the infinite-state implementation are semantically equivalent (i.e., bisimilar). 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 [14]; 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. We show that finite-state processes can be encoded up to bisimilarity in formulae of the temporal logic *EF* (more precisely, in a slightly extended version of *EF* which can also express constraints on sequences of atomic actions). Such a formula is called a *characteristic formula* for the given finite-state process. The characteristic formula Θ_f of a finite-state process f has the property that for any (general) process g whose set of actions is contained in the one of f we have that g is bisimilar to f if and only if g satisfies Θ_f. Previous works used the modal μ-calculus to construct characteristic formulae [33]. We show that the much simpler logic *EF* (a fragment of CTL and the modal μ-calculus) suffices. This is significant, because model checking with *EF* is decidable for many more classes of infinite-state systems than with the modal μ-calculus [10,20,24].

Then we apply the designed method to the class of PAD processes (defined in [21]), 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 the logic *EF* [23,20,24,19]. 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 in the case of state-extended BPP processes (which form a natural subclass of Petri nets) the problem of weak bisimilarity with finite-state processes is undecidable.

Decidability of bisimulation-like equivalences has been intensively studied for various process classes (see [28] for a survey). The majority of the results are about the decidability of strong bisimilarity, e.g., [3,9,8,34,7,16,12].

Strong bisimilarity with finite-state processes is known to be decidable for (labeled) Petri nets [15], PA, and pushdown processes [14]. Another positive result of this kind is presented in [22], 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 [13]. 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 the previously established ones.

In Section 2 we define process rewrite systems, the formalism we use to describe infinite-state systems. In Section 3 we describe the general method for deciding bisimilarity between infinite-state systems and finite-state systems. In Section 4 we use this method to construct characteristic formulae and apply them to prove the main positive decidability result. In Section 5 we prove several undecidability results for strong and weak bisimilarity. In the last section we summarize the results and outline possible future work.

2 Definitions

Transition systems are widely accepted as structures 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 $\rightarrow \subseteq S \times A \times S$ is a* transition relation.

We defined Act as a finite set; it is somewhat nonstandard, but we can allow this as all classes of process descriptions we consider generate transition systems of this kind. 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 iff $s \rightarrow^* t$.

Let $Const = \{X, Y, Z, \ldots\}$ be a countably infinite set of *process constants.* The set of (general) *process expressions*, denoted G, is defined by the following abstract syntax equation:

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

Here X ranges over $Const$ and ε is a special constant that denotes the empty expression. Intuitively, the '.' operator corresponds to a sequential composition, while the '$\|$' operator models a simple form of parallelism.

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 '$\|$'
- 'ε' as a unit for '.' and '$\|$'.

A *process rewrite system* [21] 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. The sets of process constants which are used in the rules of Δ is denoted by $Const(\Delta)$, and the set of all process expressions built over $Const(\Delta)$ is denoted by $G(\Delta)$.

Each process rewrite system Δ determines a unique transition system where states are process expressions of $G(\Delta)$, the set of labels is Act, and transitions are determined by Δ and the following inference rules (remember that '$\|$' is commutative):

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

Various subclasses of process rewrite systems can be obtained by imposing certain restrictions on the form of the rules. To specify those restrictions, we first define the classes S and P of *sequential* and *parallel* expressions, composed of all process expressions which do not contain the '$\|$' and the '.' operator, respectively. For short, we also use '1' to denote the set of process constants. A hierarchy of process rewrite systems is presented in Figure 1; the restrictions are specified by a pair (A, B), where A and B are the classes of expressions which can appear on the left-hand and the right-hand side of rules, respectively. The set of states of a system Δ which belongs to the subclass determined by (A, B) is then formed by all process expressions of $B \cap G(\Delta)$. It is important to realize that, e.g., every BPA system Δ can also be seen as a PA system, but the sets of states (processes) of Δ are *different* in the two respective cases.

The hierarchy of Figure 1 contains almost all classes of infinite-state systems which have been studied so far; BPA, BPP, and PA processes are well-known [4], PDA correspond to pushdown processes (as proved by Caucal in [6]), PN correspond to Petri nets (see, e.g., [31]), etc. This hierarchy is strict w.r.t. strong bisimulation, i.e., 'higher' classes are strictly more expressive [21].

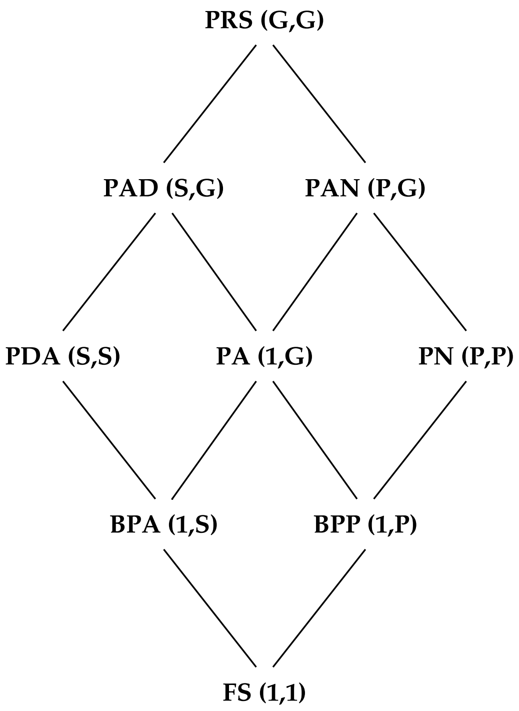

Fig. 1. A hierarchy of process rewrite systems

A convenient way how to extend expressibility of process rewrite systems is to equip them with a finite-state control unit. In order to do that, we first need to introduce the notion of *Step*. Let Δ be a PRS. Observe that each transition $E \xrightarrow{a} F$ is due to some rule $H \xrightarrow{a} K$ of Δ (i.e., H is rewritten to K within E, yielding the expression F). Generally, there can be more than one rule of Δ with this property—if, e.g., $\Delta = \{X \xrightarrow{a} X\|Y, Y \xrightarrow{a} Y\|Y\}$, then the transition $X\|Y \xrightarrow{a} X\|Y\|Y$ can be derived in one step in two different ways. For each transition $E \xrightarrow{a} F$ we denote the set of all rules of Δ which allow to derive the transition in one step by $\mathit{Step}(E \xrightarrow{a} F)$.

A *state-extended PRS* (StExt(PRS)) is a triple (Δ, Q, BT) where Δ is a PRS, Q is a finite set of *control states*, and $BT \subseteq \Delta \times Q \times Q$ is a set of *basic transitions.* The transition system generated by a state-extended PRS (Δ, Q, BT) has $Q \times G(\Delta)$ as the set of states (its elements are called *state-extended PRS processes*, or StExt(PRS) processes for short), *Act* is the set of labels, and the transition relation is determined by the following rule: $(p, E) \xrightarrow{a} (q, F)$ iff $E \xrightarrow{a} F$ and there is $H \xrightarrow{a} K \in \mathit{Step}(E \xrightarrow{a} F)$ such that $(H \xrightarrow{a} K, p, q) \in BT$.

This construction also applies to the aforementioned subclasses of PRS. It can (but does not have to) increase the expressive power of a given subclass. For example, if we add a finite-state control to a FS, PDA, or PN process, we obtain a process which can be equivalently described by another FS, PDA, or PN process, respectively (here the word ‘equivalent’ means ‘the same up to isomorphism’). In the other cases, the mentioned extension brings strictly more power—StExt(BPA) are in fact PDA processes, StExt(BPP) form a proper subclass of PN processes (which is also a proper superclass of BPP), and if we add finite-state control to PA (or to any of its superclasses), we obtain systems with full Turing power. The last fact will be demonstrated in Section 5. Let us note that PRS themselves are not Turing-powerful, because the reachability problem is decidable for them—see [21].

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 $R : Act \to 2^{Act^*}$ 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 $a \in Act$.*
- *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:

- $\mathcal{G} = (G, Act, \to)$ always denotes a (general) transition system.
- $\mathcal{F} = (F, Act, \to)$ 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.

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 $g \xrightarrow{a} g'$ for all $g, g' \in G$ and $a \in Act$, or whether $w \in R(a)$ for a given $w \in Act^*$, etc.)

Definition 3 *The* extended *transition relation $\Rightarrow \subseteq G \times Act \times G$ is defined as follows: $s \overset{a}{\Rightarrow} t$ iff $s \xrightarrow{w} t$ for some $w \in R(a)$.*

Definition 4 *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 $s \xrightarrow{a} s'$, then $t \overset{a}{\Rightarrow} t'$ for some $t' \in G$ such that $(s',t') \in P$.*
- *If $t \xrightarrow{a} t'$, then $s \overset{a}{\Rightarrow} s'$ for some $s' \in G$ such that $(s',t') \in P$.*

States $s,t \in G$ are R-N-bisimilar*, written $s \overset{RN}{\sim} t$, 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 [30,26]). The corresponding versions of R (denoted by S and W, respectively) are defined as follows ($\mathbb{N}_0$ denotes the set of all nonnegative integers):

- $S(a) = \{a\}$ 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 used 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 in the rest of this paper). One can also argue that the N predicate could be omitted from the definition of R-N-bisimilarity, as it is not employed by any known bisimulation-like equivalence. This is not completely true, as, e.g., the version of strong bisimilarity introduced in [28] uses such a predicate to distinguish between 'terminal' and 'final' states of pushdown processes (in this way it is possible to distinguish between a 'successful' termination caused by emptying the stack, and an 'unsuccessful' one (deadlock) caused by entering a state (p,E), where $E \neq \varepsilon$, from which there are no transitions).

Generally, every R-N-bisimilarity is a refinement of R-T-bisimilarity and this fact also suggests the way how to use the predicate N; its basic purpose is to impose some additional conditions on pairs of states which cannot be specified by R, but which should be satisfied by (pairs of) equivalent states. We illustrate this approach by designing a natural refinement of weak bisimilarity.

Example 5 *It is a well-known fact that weak bisimilarity does not distinguish between a state which cannot emit any action (deadlock), and a state which can emit only an infinite number of silent 'τ' actions (livelock). However, these two behaviors are considered to be* different *in many situations; for example, there are very good reasons to distinguish between deadlock and livelock in the context of operating systems. Therefore, it is natural to ask whether there is some refinement of weak bisimilarity which preserves most of its properties but eliminates the mentioned drawback at the same time. A simple solution is to* *define the D predicate in the following way:*

$$D(s,t) \textit{ is true iff } (\mathit{Init}(s) = \emptyset \Longleftrightarrow \mathit{Init}(t) = \emptyset)$$

Here $\mathit{Init}(s)$ *denotes the set of* initial actions, *defined as follows:* $\mathit{Init}(s) = \{a \in \mathit{Act} \mid s \xrightarrow{a} s' \text{ for some } s'\}$. *Now W-D-bisimilarity is a good candidate for the equivalence we are looking for; it is very similar to weak bisimilarity, but it distinguishes between deadlock and livelock. As we shall see, W-D-bisimilarity is also* decidable *between PAD processes and finite-state ones.*

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 $a \in \mathit{Act}$.
- $L(a) = \{w \in \mathit{Act}^* \mid w \text{ begins with } a\}$.
- $M(a) = \begin{cases} \mathit{Act}^* & \text{if } a = \tau; \\ \{w \in \mathit{Act}^* \mid w \text{ contains at least one } a\} & \text{otherwise.} \end{cases}$

The predicate N can also have various forms. We have already mentioned the 'T' (always true) and 'D' (deadlock equivalence). Another natural example is the 'I' predicate: $I(s,t)$ is true iff $\mathit{Init}(s) = \mathit{Init}(t)$. It is easy to see that, e.g., $\overset{ST}{\sim}$ coincides with $\overset{SI}{\sim}$, while $\overset{WI}{\sim}$ refines $\overset{WD}{\sim}$.

An important example of a bisimulation-like equivalence which cannot be seen as R-N-bisimilarity is *branching bisimilarity* (introduced in [35]). This relation places additional 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, labeled by some $a \in \mathit{Act}$.
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 pair of 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

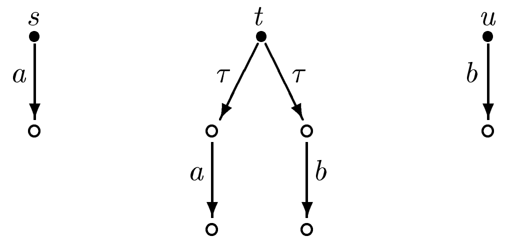

Fig. 2. A transition system considered in Example 6

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 $\overset{RN}{\sim}_i \subseteq G \times G$, $i \in \mathbb{N}_0$, 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 $s \overset{RN}{\sim} t \Longleftrightarrow \forall i \in \mathbb{N}_0 : s \overset{RN}{\sim}_i t$.

Example 6 *It is a well-known fact that in the case of weak bisimilarity (i.e., W-T-bisimilarity) the equivalence*

$$s \overset{WT}{\sim} t \iff \forall i \in \mathbb{N}_0 : s \overset{RN}{\sim}_i t$$

does not hold in general ('$\Longleftarrow$' does not have to be valid). Moreover, $\overset{WT}{\sim}_i$ is not transitive for $i \geq 1$. To see this, consider the states s, t, u in the transition system of Figure 2; we have $s \overset{WT}{\sim}_1 t$ and $t \overset{WT}{\sim}_1 u$, but $s \overset{WT}{\not\sim}_1 u$.

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

Definition 7 *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 the case of R-N-bisimilarity—for each $i \in \mathbb{N}_0$ 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 8 *Two states s, t of $\mathcal{G}$ are R-N-bisimilar iff s and t are extended R-N-bisimilar.*

PROOF. Every extended R-N-bisimulation is also an R-N-bisimulation; here we need that $a \in R(a)$ for each $a \in \mathit{Act}$. Conversely, every R-N-bisimulation is also an extended R-N-bisimulation; each extended transition is a finite sequence of transitions, hence we can concatenate 'responses' to those individual transitions, obtaining a valid response to the original extended transition. Here we need the second requirement of Definition 2, that the relation R is closed under substitution. $\square$

Lemma 9 *The following properties hold:*

(1) $\overset{RN}{\simeq}_i$ *is an equivalence relation for each* $i \in \mathbb{N}_0$.
(2) Let s, t be states of $\mathcal{G}$. Then $\forall i \in \mathbb{N}_0 : s \overset{RN}{\sim}_i t$ *iff* $\forall i \in \mathbb{N}_0 : s \overset{RN}{\simeq}_i t$.

PROOF.

(1) For the first part, reflexivity and symmetry are obvious. Transitivity follows from the condition that the relation R is closed under substitution.
(2) It follows from the definition of $\overset{RN}{\simeq}$ that $s \overset{RN}{\simeq}_i t \implies s \overset{RN}{\sim}_i t$. Hence, it suffices to realize that if $s \overset{RN}{\not\simeq}_i t$, then $s \overset{RN}{\not\sim}_j t$ for some $j \in \mathbb{N}_0$—as Al can force his win using i 'long' moves and each of those moves consists of a finite number of 'short' moves, Al could actually 'decompose' his attacks, playing only (a finite number of) short moves. $\square$

Remark 10 *For all states s, t of $\mathcal{G}$ and $i \in \mathbb{N}_0$ we have that if $s \overset{RN}{\simeq}_i t$ then also $s \overset{RN}{\sim}_i t$. However, there is no 'reverse correspondence'—it can be easily shown that for arbitrarily large j the implication $s \overset{RN}{\sim}_j t \implies s \overset{RN}{\simeq}_1 t$ is generally invalid (the implication is invalid even in the case when t is a state in a one-state TS). See Section 5 for details.*

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

Lemma 11 *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 $\overset{RN}{\simeq}_{k-1} = \overset{RN}{\sim}$. $\square$

Theorem 12 *States $g \in G$ and $f \in F$ are R-N-bisimilar iff the following conditions hold:*

1. $g \overset{RN}{\simeq}_k f$.
2. *For each state g' which is 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 $a \in \mathit{Act}$ (the case when $f' \overset{a}{\Rightarrow} f''$ is handled in the same way). By definition of $\overset{RN}{\simeq}_k$, there is an 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 11). 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 finished. $\square$

Remark 13 *We have already mentioned that the equivalence*

$$s \overset{RN}{\sim} t \iff \forall i \in \mathbb{N}_0 : s \overset{RN}{\simeq}_i t$$

is generally invalid (e.g., in the case of weak bisimilarity). However, as soon as we assume that t is a state in a finite-state transition system, the equivalence holds. This is an immediate consequence of the previous theorem. Moreover, the second part of Lemma 9 says that we could also use the $\overset{RN}{\sim}_i$ approximations on 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 *every* 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. Hence, 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. Therefore, we can effectively replace the transition relation of $\mathcal{F}$ with its corresponding extended transition relation. Al and Ex can now play only 'short' moves consisting of exactly one transition whenever playing within the modified system $\mathcal{F}$—each such move corresponds to some extended transition of the original system $\mathcal{F}$ and vice versa. This observation leads to the notion of *branching tree*, which allows to 'extract' from $\mathcal{F}$ the information which is relevant for the first k moves in the extended R-N-bisimulation game. The aim of the following definition is to describe all such trees up to isomorphism (remember that Act is a *finite* set).

Definition 14 *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 labeled by an element of $F \cup \{\perp\}$.*
- *A Tree with depth at most $i+1$ is any directed tree with root r whose nodes are labeled by elements of $F \cup \{\perp\}$, arcs are labeled 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 $r \xrightarrow{a} s'$ for $s \neq s'$, then the subtrees rooted by s and s' are not isomorphic.*

It is clear that the set $Tree_j$ is finite for every $j \in \mathbb{N}_0$. More precisely, its cardinality (denoted $NT(j)$) is given by:

- $NT(0) = k+1$
- $NT(i+1) = (k+1) \cdot 2^{n \cdot NT(i)}$, where $n = card(Act)$

The set $Tree_j$ is effectively constructible for every $j \in \mathbb{N}_0$. 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, this is a *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 the extended transitions of $\mathcal{F}$).

- The predicate $N(g', p')$, where $g' \in G$ and p' is a state of the Tree process p, is evaluated as follows:

$$N(g',p') = \begin{cases} \text{true} & \text{if } \mathit{label}(p') = \bot \text{ and} \\ & N(g',f) = \text{false for every } f \in F \\ \text{false} & \text{if } \mathit{label}(p') = \bot \text{ and} \\ & N(g',f) = \text{true for some } f \in F \\ N(g', \mathit{label}(p')) & \text{otherwise} \end{cases}$$

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 15 *Let g be a state of $\mathcal{G}$, $j \in \mathbb{N}_0$. Then $g \overset{RN}{\simeq}_j p$ for some $p \in \mathit{Tree}_j$.*

PROOF. We proceed by induction on j:

- $\mathbf{j = 0}$: Then p is a Tree with no arcs and just one node labeled by some $f \in F$ such that $N(g, f)$ is true; if there is no such f, then it is labeled by $\bot$. Clearly $g \overset{RN}{\simeq}_0 p$.
- **Induction step:** We need to construct a Tree p such that $g \overset{RN}{\simeq}_{j+1} p$. The Tree p has a root r whose label is determined in the same way as in the case when $j = 0$. The successors of r are defined by

$$r \overset{a}{\rightarrow} s \quad \text{iff} \quad g \overset{a}{\Rightarrow} g' \text{ and } g' \overset{RN}{\simeq}_j s$$

  Note that for each g' there is $s \in \mathit{Tree}_j$ such that $g' \overset{RN}{\simeq}_j s$ by induction hypothesis. Thus, we have $g \overset{RN}{\simeq}_{j+1} p$ as required. □

Lemma 16 *Let f be a state of $\mathcal{F}$, $j \in \mathbb{N}_0$, and $p \in \mathit{Tree}_j$ such that $f \overset{RN}{\simeq}_j p$. Then for every state g of $\mathcal{G}$ we have that $g \overset{RN}{\simeq}_j f$ iff $g \overset{RN}{\simeq}_j p$.*

PROOF.
'$\Longrightarrow$': By induction on j:

- $\mathbf{j = 0}$: As $f \overset{RN}{\simeq}_0 p$ and $g \overset{RN}{\simeq}_0 f$, we have that $N(g, f)$ is true and (the root of) p is labeled by some f' such that $N(f, f')$ is true. Hence, $N(g, f')$ is true and $g \overset{RN}{\simeq}_0 p$.
- **Induction step:** Let $f \overset{RN}{\simeq}_{j+1} p$ and $g \overset{RN}{\simeq}_{j+1} f$. We prove that $g \overset{RN}{\simeq}_{j+1} p$. Clearly $N(g, \mathit{label}(p))$ is true (see above). Let $g \overset{a}{\Rightarrow} g'$ (the case when $p \overset{a}{\rightarrow} p'$ can be done similarly). We need to show that $p \overset{a}{\rightarrow} p'$ for some p' with

$g' \overset{RN}{\simeq}_j p'$. As $g \overset{RN}{\simeq}_{j+1} f$, there is $f' \in F$ such that $f \overset{a}{\Rightarrow} f'$ and $g' \overset{RN}{\simeq}_j f'$. Furthermore, as $f \overset{RN}{\simeq}_{j+1} p$ and $f \overset{a}{\Rightarrow} f'$, there is p' such that $p \overset{a}{\rightarrow} p'$ and $f' \overset{RN}{\simeq}_j p'$. To sum up, we have $f' \overset{RN}{\simeq}_j p'$ and $g' \overset{RN}{\simeq}_j f'$, hence $g' \overset{RN}{\simeq}_j p'$ by induction hypotheses.

'$\Longleftarrow$': In a similar way. □

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 $g' \overset{RN}{\simeq}_j p$?
A decision algorithm may use an oracle which for any state g'' of $\mathcal{G}$ answers whether $g'' \overset{RN}{\simeq}_j p$.

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 $g' \overset{RN}{\simeq}_k p$?
A decision algorithm may use an oracle which for any state g'' of $\mathcal{G}$ answers whether $g'' \overset{RN}{\simeq}_k p$.

Formally, the transition system $\mathcal{F}$ should also be given in the instances of the aforementioned 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 17 *If the Step-problem is decidable (possibly using the mentioned oracle), then $\overset{RN}{\simeq}_k$ is decidable between all 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 every $0 \leq j \leq k$. 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 need to introduce two auxiliary finite sets:

- The set of **C**ompatible **S**teps, denoted CS_j^f, is composed exactly of all pairs of the form (a, p), where $a \in Act$ and $p \in Tree_j$, such that $f \overset{a}{\Rightarrow} f'$ for some

f' with $f' \overset{RN}{\simeq}_j p$.

- The set of **IN**Compatible **S**teps, denoted $INCS_j^f$, is a complement of CS_j^f w.r.t. $Act \times Tree_j$.

The sets CS_j^f and $INCS_j^f$ 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 every $(a, p) \in CS_j^f$ (see Lemma 15 and 16). 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 $INCS_j^f$ (due to Lemma 15 and 16). 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 18 *Decidability of the Step-problem and the Reach-problem (possibly using the indicated oracles) implies decidability of the problem whether for each g' which is 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 finish the proof, we need to define one auxiliary set:

- The set of **IN**Compatible **T**rees, denoted $INCT$, is composed of all $p \in Tree_k$ such that $f \overset{RN}{\not\simeq}_k p$ for every 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 every 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 15 and 16). □

4 Characteristic Formulae

In this section we show how to apply the previously designed general method to construct characteristic formulae for finite-state systems in the temporal logic $EF_{\mathcal{C}}$ (we show that the Step-problem as well as the Reach-problem can be encoded by $EF_{\mathcal{C}}$ formulae). Consequently, we reduce the problem of R-N-bisimilarity between infinite-state processes and finite-state ones to the model checking problem for $EF_{\mathcal{C}}$. Therefore it is possible to apply decidability results from this area. In this way we 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 $EF_{\mathcal{C}}$ (it is an extended version of the logic *EF* [10] with constraints on sequences of actions). Let $\mathcal{C}$ be a finite set of unary predicates on sequences of atomic actions. The formulae of $EF_{\mathcal{C}}$ have the following syntax (where $a \in Act$ and $C \in \mathcal{C}$):

$$\Phi ::= true \;\mid\; \neg\Phi \;\mid\; \Phi_1 \wedge \Phi_2 \;\mid\; \langle a \rangle \Phi \;\mid\; \Diamond_C \Phi$$

Let $\mathcal{T} = (S, Act, \rightarrow)$ be a transition system. The denotation $[\![\Phi]\!]$ of a formula Φ is a set of states of $\mathcal{T}$ where the formula *holds*; it is defined as follows (sequences of atomic actions are denoted by w):

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

The predicates of $\mathcal{C}$ are used to express constraints on sequences of actions. An instance of the *model checking problem* is given by a state s in S and an $EF_{\mathcal{C}}$ formula Φ. The question is whether $s \in [\![\Phi]\!]$. This property is also denoted by $s \models \Phi$.

A *characteristic formula* Θ_f for a finite-state process f w.r.t. R-N-bisimulation has the property that for every (general) process g whose set of actions is contained in the set of actions of f we have

$$g \overset{RN}{\sim} f \iff g \models \Theta_f$$

For every R-N-bisimulation we define the set of predicates $\mathcal{R}$ as follows:

$$\mathcal{R} = \{ C_a \mid a \in Act, C_a(w) \Longleftrightarrow w \in R(a) \} \cup \{ true, false \}$$

As usual, we write $\Diamond\Phi$ instead of $\Diamond_{true}\Phi$.

Let us fix a general TS $\mathcal{G} = (G, Act, \rightarrow)$ and a finite-state TS $\mathcal{F} = (F, Act, \rightarrow)$ with k states in the same way as in the previous section. We show how to encode the Step and the Reach problems by $EF_{\mathcal{R}}$ formulae. The first difficulty is the N predicate. Although it is decidable, this fact is generally of no use because we cannot make any assumptions on 'strategies' of model checking algorithms. Instead, we restrict our attention to those predicates which can be encoded by $EF_{\mathcal{R}}$ formulae in the following sense: for each $f \in F$ there is an $EF_{\mathcal{R}}$ 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 $EF_{\mathcal{R}}$ formulae is, e.g., the 'I' predicate defined in the previous section: For every $f \in F$ let $A_f := \{a \in Act \mid \exists f'.\, f \xrightarrow{a} f'\}$. Then

$$\Psi_f := \bigwedge_{a \in A_f} \langle a \rangle true \;\wedge \bigwedge_{a \in Act - A_f} \neg\langle a \rangle true$$

The 'D' predicate can be encoded in a similar way.

Now we design the family of $\Phi_{j,p}$ formulae, where $0 \le j \le k$ and $p \in Tree_j$, in such a way that for every $g \in G$ the following equivalence holds:

$$g \overset{RN}{\simeq}_j p \iff g \models \Phi_{j,p}$$

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

  where $f = label(p)$ and $S(p, a) = \{p' \mid p \xrightarrow{a} p'\}$. Empty conjunctions are equivalent to *true*.

Thus, the characteristic formula Θ_f for a process f of a finite-state system $\mathcal{F} = (F, Act, \rightarrow)$ with k states is defined by

$$\Theta_f \equiv \Phi_{k,f} \wedge \neg\Diamond \left(\bigwedge_{f' \in F} \neg\Phi_{k,f'} \right)$$

The decidability of the model checking problem for the logic $EF_{\mathcal{C}}$ depends on properties of the family of constraints $\mathcal{C}$. It has been shown in [23] that the model checking problem for PA processes and the logic $EF_{\mathcal{C}}$ is decidable for the class of *decomposable constraints* (see also [19] where the same result was proved later using a completely different technique). This result has been generalized to PAD processes in [20,24]. These constraints are called decomposable, because they can be decomposed w.r.t. sequential and parallel composition. A 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.

$$\forall w, w_1, w_2.\ w_1 w_2 = w \implies \left(C(w) \iff \bigvee_{i \in I} C_i^1(w_1) \wedge C_i^2(w_2) \right)$$

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.

$$\forall w_1, w_2. \left((\exists w \in \mathit{interleave}(w_1, w_2).\, C(w)) \iff \bigvee_{i \in J} (C_i^1(w_1) \wedge C_i^2(w_2)) \right)$$

   where $\mathit{interleave}(w_1, w_2)$ is the set of all interleavings of w_1 and w_2 defined by

$$\begin{aligned} \mathit{interleave}(\varepsilon, w) &:= \{w\} \\ \mathit{interleave}(w, \varepsilon) &:= \{w\} \\ \mathit{interleave}(a_1 w_1, a_2 w_2) &:= \{a_1 w \mid w \in \mathit{interleave}(w_1, a_2 w_2)\} \cup \\ & \{a_2 w \mid w \in \mathit{interleave}(a_1 w_1, w_2)\} \end{aligned}$$

It is easy to see that the closure of a set of decomposable constraints under disjunction is again a set of decomposable constraints (see [19,32] for more on decomposable constraints and decomposable languages). All the previously mentioned examples of relations R can be expressed by decomposable constraints. Consider the relation W for weak bisimulation. There we have the following constraints:

$$W_\tau(w) := (w = \tau^i \text{ for some } i \in \mathbb{N}_0)$$

$W_a(w) := (w = \tau^i a \tau^j$ for some $i, j \in \mathbb{N}_0)$

These constraints can be decomposed w.r.t. sequential and parallel composition. For W_τ this is trivial. For W_a we have

$$\begin{aligned} W_a(w_1 w_2) &\Longleftrightarrow (W_a(w_1) \wedge W_\tau(w_2)) \vee (W_\tau(w_1) \wedge W_a(w_2)) \\ (\exists w \in \mathit{interleave}(w_1, w_2).\, W_a(w)) &\Longleftrightarrow (W_a(w_1) \wedge W_\tau(w_2)) \vee (W_\tau(w_1) \wedge W_a(w_2)) \end{aligned}$$

Now we show decomposability for some other (nonstandard) relations that were defined in Section 3. For the relation K the decomposition is trivial. For the relation L we have the constraint

$$L_a(w) := w \text{ begins with } a$$

The decomposition is

$$\begin{aligned} L_a(w_1 w_2) &\Longleftrightarrow L_a(w_1) \\ (\exists w \in \mathit{interleave}(w_1, w_2).\, L_a(w)) &\Longleftrightarrow L_a(w_1) \vee L_a(w_2) \end{aligned}$$

For the relation M we have the constraints

$M_\tau(w) := \mathit{true}$
$M_a(w) := w$ contains at least one a

The decomposition of M_τ is trivial. The decomposition of M_a is

$$\begin{aligned} M_a(w_1 w_2) &\Longleftrightarrow M_a(w_1) \vee M_a(w_2) \\ (\exists w \in \mathit{interleave}(w_1, w_2).\, M_a(w)) &\Longleftrightarrow M_a(w_1) \vee M_a(w_2) \end{aligned}$$

However, there are also relations 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. The function R is obviously closed under substitution, but the corresponding set of constraints is not decomposable. On the other hand, there are decomposable constraints that are not closed under substitution like, e.g., $R(a) := \{a^i \mid 1 \leq i \leq 5\}$. Now we can formulate a general decidability theorem:

Theorem 19 *The problem $g \overset{RN}{\sim} f$, where R yields a set of constraints $\mathcal{R}$ contained in a set $\mathcal{DC}$ of decomposable constraints, N is expressible in $EF_{\mathcal{R}}$, g is a PAD processes, and f is a finite-state process, is decidable.*

Corollary 20 *Weak bisimilarity between PAD processes and finite-state ones is decidable.*

Remark 21 (Complexity of the problem)
The complexity of our algorithm for the problem $g \overset{RN}{\sim} f$ depends on the complexity of the model checking problem for $EF_{\mathcal{C}}$ and PAD, which is not known exactly yet. The algorithm for PAD in [20,24] and the different algorithms for PA in [23] and [19] all have non-elementary complexity. For BPP, model checking with $EF_{\mathcal{C}}$ is PSPACE-complete [22,24] (see also Section 6). The $EF_{\mathcal{R}}$ formulae that are constructed for a finite-state system $\mathcal{F}$ with k states have exponential size in k, but a nesting-depth of the operator $\Diamond$ that is only polynomial in k. Model checking can be done 'on-the-fly' while these formulae are constructed and thus polynomial space suffices. Hence, the problem $g \overset{RN}{\sim} f$ is in PSPACE for BPP.

For BPA and PDA, model checking with $EF_{\mathcal{C}}$ is known to be in EXPTIME [36,5]. It was claimed in [5] that it is even in PSPACE, but the given proof contains an error (it assumed that an accepting polynomial space-bounded Turing machine always has an accepting computation of polynomial length; however, there are cases where the shortest accepting computation has an exponential length). Thus the question about the complexity of model checking pushdown systems with $EF_{\mathcal{C}}$ is open again. Still we conjecture PSPACE-completeness to be most likely, because the number of alternations between conjunction and disjunction in the model checking problem is bounded by the size of the formula and thus polynomial. So far, our construction yields an EXPTIME algorithm for the problem $g \overset{RN}{\sim} f$ for BPA and PDA.

The known lower bounds for the model checking problem are PSPACE-hardness for BPP [10] and BPA [25] (and thus also for for PDA, PA and PAD). However, unlike the upper bounds, the lower bounds for the model checking problem do not carry over to the bisimulation problem $g \overset{RN}{\sim} f$. For example, it has recently been shown that weak bisimilarity between BPA and finite-state systems is decidable in polynomial time [18], while model checking BPA with $EF_{\mathcal{C}}$ is PSPACE-hard [25].

Decidability of the model checking problem for the $EF_{\mathcal{R}}$ logic in a certain class of transition systems $\mathcal{K}$ is a sufficient but not necessary condition for decidability of R-N-bisimilarity between processes of $\mathcal{K}$ and finite-state ones. For example, model checking the 'pure' *EF* (without any constraints) is undecidable for Petri nets, but the Step and the Reach problems are decidable for S-T-bisimilarity [15]. In fact, strong bisimilarity is the simplest form of R-N-bisimilarity and the *EF* formulae which encode the two problems are therefore very simple as well. An exact formulation of this observation is given in the following theorem:

Theorem 22 *An EF formula is* simple *iff it is of the form $\Diamond\Phi$ where the subformula Φ does not contain any $\Diamond$-operator (i.e., Φ is a formula of Hennessy-Milner logic [26]). If the model checking problem for simple EF formulae is* *decidable in a class $\mathcal{K}$ of transition systems, then strong bisimilarity is decidable between processes of $\mathcal{K}$ and finite-state ones.*

PROOF. The $\overset{ST}{\simeq}_j$ equivalence with a given Tree process p can be encoded by a formula of Hennessy-Milner logic for every $j \in \mathbb{N}_0$. Consequently, the Step problem can also be encoded by a formula of Hennessy-Milner logic, and the Reach problem is encoded by a formula of the form $\Diamond\Phi$ where Φ is a formula of Hennessy-Milner logic. □

The model checking problem for simple *EF* formulae is essentially a kind of generalized reachability problem (one checks whether there is a reachable state that satisfies a given formula of Hennessy-Milner logic). Of course, it is much easier than the general model checking problem for *EF*. Thus, decidability issues can be different—we have already mentioned that model checking *EF* logic is undecidable for Petri nets; however, model checking simple *EF* is decidable (due to the decidability of the Reach problem—see below). For example, in the case of Petri nets we can observe that the markings which satisfy some formula of H.M. logic can be characterized by boolean combinations of constraints of the form $p \geq k$ or $p \leq k$, meaning that there are at least/at most k tokens in place p. This leads to a generalized reachability problem which is decidable [11].

Now we show that the model checking problem for simple *EF* formulae can be seen as a reformulation of the Step and the Reach problems in the case of strong bisimilarity (the Step problem is trivially decidable, and the Reach problem is 'equivalently hard' to the model checking problem for the simple *EF* logic). This shows the essence of the whole problem in a new light.

Theorem 23 *The model checking problem for simple EF formulae and the special variant of the Reach problem for strong bisimilarity are inter-reducible in the Turing sense (i.e., decidability of one of the two problems implies decidability of the other one).*

PROOF. Decidability of the model checking problem for simple *EF* formulae implies decidability of the Reach problem, as shown in Theorem 22. We prove the other direction; let $\Diamond\Phi$ be a simple *EF* formula. First, let us realize that the sub-formula Φ cannot distinguish between states related by $\overset{ST}{\simeq}_n$, where $n = \mathit{length}(\Phi)$. Due to Lemma 15 we know that for every state g of the transition system $\mathcal{G}$ there is a $p \in \mathit{Tree}_n$ such that $g \overset{ST}{\simeq}_n p$ (as the predicate T is trivial, we do not have to label the nodes of Trees; hence the construction of Tree_n does not depend on the transition system $\mathcal{F}$—see Definition 14). For each $p \in \mathit{Tree}_n$ we check whether $p \models \Phi$. Now it is easy to see that $g' \models \Diamond\Phi$ iff $\mathit{Reach}(g', p) = \mathit{true}$ for some $p \in \mathit{Tree}_n$ such that $p \models \Phi$. □

5 Undecidability Results

In this section we provide several negative (undecidability) results which help to clarify the decidability/undecidability border in the area of comparing infinite-state processes with finite-state ones.

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.

Definition 24 A counter machine $\mathcal{M}$ *with nonnegative counters* $c_1, c_2, ..., c_m$ *is a sequence of instructions*

$$\begin{array}{ll} 1: & \mathit{INS}_1 \\ 2: & \mathit{INS}_2 \\ \vdots & \\ n-1: & \mathit{INS}_{n-1} \\ n: & \texttt{halt} \end{array}$$

where each INS_i *(*$i = 1, 2, ..., n-1$*) is in one of the following two forms (assuming* $1 \le k, k_1, k_2 \le n$, $1 \le j \le m$*)*

- $c_j := c_j + 1;\ \ \texttt{goto}\ k$
- $\texttt{if}\ c_j = 0\ \texttt{then goto}\ k_1\ \texttt{else}\ (c_j := c_j - 1;\ \ \texttt{goto}\ k_2)$

The halting problem is undecidable even for Minsky machines with two counters initialized to zero values [27]. Any such machine $\mathcal{M}$ can be easily 'mimicked' by a StExt(PA) process $P(\mathcal{M}) = (\Delta, Q, BT)$ where

- Δ contains the following rules:
  - · $Z_j \xrightarrow{a} I_j.Z_j$, $Z_j \xrightarrow{a} Z_j$
  - · $I_j \xrightarrow{a} I_j.I_j$, $I_j \xrightarrow{a} \varepsilon$

  where $j \in \{1, 2\}$.
- $Q = \{q_1, \ldots, q_n\}$, where n is the number of instructions of $\mathcal{M}$.
- BT is determined by the following rules:

(1) If the program of $\mathcal{M}$ contains an instruction of the form

$$l:\ c_j := c_j + 1;\quad \texttt{goto}\ k$$

then BT contains the elements $(Z_j \xrightarrow{a} I_j.Z_j, q_l, q_k)$ and $(I_j \xrightarrow{a} I_j.I_j, q_l, q_k)$.

(2) If the program of $\mathcal{M}$ contains an instruction of the form

$$l: \texttt{ if } c_j = 0 \texttt{ then goto } k_1 \texttt{ else } (c_j := c_j - 1;\ \texttt{goto } k_2)$$

then BT contains the elements $(Z_j \xrightarrow{a} Z_j, q_l, q_{k_1})$ and $(I_j \xrightarrow{a} \varepsilon, q_l, q_{k_2})$.

(3) Each element of BT can be derived using the rule 1 or 2.

Intuitively, the (two) counters of the machine $\mathcal{M}$ are modeled by a simple PA process $(I_1.I_1 \ldots I_1.Z_1)\|(I_2.I_2 \ldots I_2.Z_2)$ where the number of I_j's means the current value of the counter c_j, $j \in \{1,2\}$ (the starting zero point being modeled by $Z_1\|Z_2$). The control states $q_1, \ldots, q_n$ correspond to the instructions of $\mathcal{M}$ (more precisely, to their labels). Each state determines the unique transition to be performed next with the exception of q_n which is the 'halting state'. The process $(q_1, Z_1\|Z_2)$ is able either to perform the action a boundedly many times and to stop (its behavior can be defined as a^m for some $m \in \mathbb{N}_0$) or to do a forever (its behavior being a^ω); this depends on whether the machine $\mathcal{M}$ halts or not. Notice that a^ω is the behavior of the one-state process f where $\{f \xrightarrow{a} f\}$ is the underlying PRS. When we declare as *reasonable* any equivalence which distinguishes between (processes with) behaviors a^ω and a^m, we can conclude:

Theorem 25 *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. It suffices to realize that we can label all transitions in $P(\mathcal{M})$ by τ and add a special a-transition enabled in the (halting) state q_n. Now $q_1(Z_1\|Z_2) \overset{WT}{\simeq}_1 \tau^\omega$ iff the machine $\mathcal{M}$ does not halt, and similarly $q_1(Z_1\|Z_2) \overset{WT}{\simeq}_1 f$ where $\{f \xrightarrow{\tau} f, f \xrightarrow{a} g\}$ iff the machine $\mathcal{M}$ halts.

Now, the claim of Remark 10 is also easy to see; if we take the modified process $P(\mathcal{M})$ of the previous paragraph, we can observe that $q_1(Z_1\|Z_2) \overset{WT}{\sim}_j \tau^\omega$ for every j which is less than the number of computational steps of $\mathcal{M}$. On the other hand, if $\mathcal{M}$ halts then $q_1(Z_1\|Z_2) \overset{WT}{\not\simeq}_1 \tau^\omega$. Therefore, the implication $q_1(Z_1\|Z_2) \overset{WT}{\sim}_j \tau^\omega \implies q_1(Z_1\|Z_2) \overset{WT}{\simeq}_1 \tau^\omega$ is invalid for any $j \in \mathbb{N}$, because for each such j there is a machine with more then j computational steps which halts.

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.

We have already shown that every R-N-bisimilarity such that R yields de-

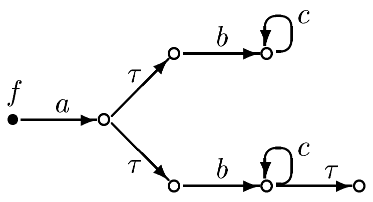

Fig. 3. A finite-state system used in the proof of Theorem 26

composable constraints and N is expressible within $EF_{\mathcal{R}}$ is decidable between StExt(BPA) (i.e., PDA) processes and finite-state ones. In the case of StExt(BPP), strong bisimilarity with finite-state processes is decidable [15]. Here we demonstrate that the problem for weak bisimilarity is undecidable. Our proof is obtained by modifying the construction which has been used in [13] to show the undecidability of weak bisimilarity between Petri nets and finite-state systems. To make this paper self-contained, we now give a concise description of this modified construction.

Theorem 26 *Weak bisimilarity is undecidable between StExt(BPP) processes and finite-state ones.*

PROOF. Consider a Minsky machine $\mathcal{M}$ as in Definition 24 with just two counters ($m = 2$). In a stepwise manner, we show how to construct a StExt(BPP) process $P(\mathcal{M})$ such that $P(\mathcal{M})$ is weakly bisimilar to the process f of Figure 3 iff $\mathcal{M}$ does not halt.

We begin with using just the action τ, the process constants Z, I_1, I_2, D, S and the control states $p_1, \ldots, p_n, q_1, \ldots, q_n$. The states (p_1, Z), (q_1, Z) are considered as two possible starting ones. Basic transitions of $P(\mathcal{M})$ are determined as follows: for every machine instruction

$$l: \; c_j := c_j + 1; \quad \texttt{goto } k$$

we have $(Z \xrightarrow{\tau} I_j \| S \| Z, p_l, p_k)$ and $(Z \xrightarrow{\tau} I_j \| S \| Z, q_l, q_k)$. For every machine instruction

$$l: \; \texttt{if } c_j = 0 \texttt{ then goto } k_1 \texttt{ else } (c_j := c_j - 1; \; \texttt{goto } k_2)$$

we have

$$(I_j \xrightarrow{\tau} D \| S, p_l, p_{k_2}), (Z \xrightarrow{\tau} S \| Z, p_l, p_{k_1}), (I_j \xrightarrow{\tau} I_j \| S, p_l, q_{k_1}),$$

and

$$(I_j \xrightarrow{\tau} D \| S, q_l, q_{k_2}), (Z \xrightarrow{\tau} S \| Z, q_l, q_{k_1}).$$

We observe that, for every reachable state (r, E), exactly one occurrence of Z appears in the expression E; the constant Z only serves as an auxiliary symbol which is used to model the 'empty left-hand side' of rules. The number of I_1 (I_2) in E is meant to correspond to the current value of counter c_1 (c_2). By (the occurrences of) S we count the number of steps, and by D the number of 'decreasing steps'.

Thus both (p_1, Z) and (q_1, Z) can simulate the computation of $\mathcal{M}$ (with counters initialized to zero). Nevertheless, also 'cheating steps' (performing a 'zero step' instead of a decreasing one) are possible; it reflects the inability of StExt(BPP) (more generally, Petri nets) to test for zero. Note that by (and only by) a cheating step we can go from the 'p-domain' to the 'q-domain'.

Now we shall refine the transitions mentioned so far. The idea is to view the sequence of steps as a string of 0's (non-decreasing steps) and 1's (decreasing steps), and to enable D to 'count' the respective binary number. We introduce an additional auxiliary constant C, and replace every transition $(E_1 \xrightarrow{\tau} E_2, r_1, r_2)$ by the set

$$\begin{aligned}&(E_1 \xrightarrow{\tau} E_2, r_1, r'),\ (D \xrightarrow{\tau} C \| C, r', r'),\ (Z \xrightarrow{\tau} Z, r', r''),\\ &(C \xrightarrow{\tau} D, r'', r''),\ (Z \xrightarrow{\tau} Z, r'', r_2)\end{aligned}$$

where r', r'' are newly added control states. It allows (though does not force) to double the number of D's in each step (after adding 1 in the case of a decreasing step). Now we add a control state h and the basic transition $(Z \xrightarrow{\tau} D \| Z, q_n, h)$.

Defining $vec(E)$ as the 5-dimensional vector giving the numbers of (occurrences of) I_1, I_2, C, D, S in E, we can easily derive (similarly as in [12]) that the set $\{\, vec(E) \mid \exists r$ such that (r, E) is reachable from $(p_1, Z)\,\}$ is a subset of $\{\, vec(E) \mid \exists r$ such that (r, E) is reachable from $(q_1, Z)\,\}$; moreover, the two sets are equal iff $\mathcal{M}$ does not halt.

To proceed with the construction of our desired $P(\mathcal{M})$, we now take a disjoint union of the so far constructed StExt(BPP) system with its isomorphic duplicate. For a control state r (or a process constant X), we denote the respective duplicate by $\overline{r}$ (or $\overline{X}$).

We now introduce new control states s_1, s_2, s_3; moreover, the pairs (s, r), $(s, \overline{r})$—where $s \in \{s_1, s_2, s_3\}$ and r is 'old'—will also serve as control states. The process $P(\mathcal{M})$ is defined as $((s_1, q_1), Z \| \overline{Z})$ when we also include the following basic transitions (adding actions a, b, c): for every $(E \xrightarrow{\tau} E', r, r')$ we add

$$(E \xrightarrow{\tau} E', (s_1, r), (s_1, r')), (E \xrightarrow{\tau} E', (s_2, \overline{r}), (s_2, \overline{r'})).$$

For every r, we add

$$(Z \stackrel{\tau}{\rightarrow} Z, (s_1, r), s_1), (Z \stackrel{\tau}{\rightarrow} Z, (s_2, \overline{r}), s_2).$$

We also add $(Z \stackrel{a}{\rightarrow} Z, s_1, (s_2, \overline{p_1}))$ and $(Z \stackrel{b}{\rightarrow} Z, s_2, s_3)$. Finally, for every $X \in \{I_1, I_2, C, D, S\}$ we add a new control state s_X and

$$(X \stackrel{c}{\rightarrow} X, s_3, s_3),\ (\overline{X} \stackrel{c}{\rightarrow} \overline{X}, s_3, s_3),\ (X \stackrel{\tau}{\rightarrow} \varepsilon, s_3, s_X),$$
$$(\overline{X} \stackrel{\tau}{\rightarrow} \varepsilon, s_X, s_3),\ (Z \stackrel{c}{\rightarrow} Z, s_X, s_X)$$

Checking that $P(\mathcal{M})$ is weakly bisimilar to f iff $\mathcal{M}$ does not halt can be done analogously to [13]. □

6 Conclusions, Future Work

We designed a general method for proving decidability of R-N-bisimilarity between infinite-state processes and finite-state ones (Theorem 12) by reducing this problem to two other problems—the Step and the Reach problem (Theorem 17 and 18). We also showed how to encode these special problems by formulae of $EF_{\mathcal{R}}$ logic. In this way we constructed characteristic formulae for finite-state systems up to bisimulation in the logic $EF_{\mathcal{C}}$. As this logic is decidable for PAD (and hence also PA and PDA) processes, we obtained a general decidability theorem (Theorem 19), which says that every R-N-bisimilarity such that R yields decomposable constrains on sequences of actions and N can be expressed by $EF_{\mathcal{R}}$ formulae is decidable between PAD and finite-state processes. This class of R-N-bisimilarities includes all versions of R-N-bisimulation equivalences mentioned in this paper. Examples are the relations $\overset{KI}{\sim}$, $\overset{LT}{\sim}$, $\overset{MI}{\sim}$, or $\overset{WD}{\sim}$, but most importantly $\overset{ST}{\sim}$ and $\overset{WT}{\sim}$ (i.e., strong and weak bisimilarity).

Then we demonstrated that each 'reasonable' R-N-bisimilarity is undecidable between StExt(PA) processes and finite-state ones (Theorem 25); this is caused by the fact that StExt(PA) processes have full Turing power. Moreover, even if we restrict our attention to StExt(BPP), we get an undecidability result for weak bisimilarity (Theorem 26). This proof is obtained by a modification of the one which has been used for Petri nets.

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 clarify what results have been previously obtained by other researchers, our table contains more rows than it is necessary (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).

We also add a special column which indicates decidability of the model-checking problem for the logic *EF*. The decidability of *EF* for pushdown processes (PDA) and BPA follows from a much stronger result by Muller and Schupp [29] who showed the decidability of monadic second order logic for pushdown automata. Later, model checking PDA with *EF* was shown to be in *EXPTIME*[36,5] (see also Remark 21). Model checking BPP with *EF* was shown to be decidable by Esparza [10] and *PSPACE*-complete by Mayr [22,24]. Decidability of *EF* for PA was shown by Mayr [23] and later by Lugiez and Schnoebelen [19], who used a completely different method. The decidability for PAD was shown in [20,24]. The undecidability of *EF* for Petri nets was shown by Esparza in [10]. The undecidability of *EF* for StExt(BPP) and StExt(PA) follows directly from the undecidability results on bisimilarity in this paper.

	$\overset{ST}{\sim}$	$\overset{WT}{\sim}$	$\overset{RN}{\sim}$	*EF*
BPA	Yes [9]	**YES**	**YES**	Yes [29,5]
BPP	Yes [8]	Yes [22]	**YES**	Yes [10,22,24]
PA	Yes [14]	**YES**	**YES**	Yes [23,19]
StExt(BPA), i.e., PDA	Yes [14]	**YES**	**YES**	Yes [29,5]
StExt(BPP), i.e., PPDA	Yes [15]	**NO**	**NO**	**NO**
StExt(PA)	No [14]	No [14]	No [14]	**NO**
PAD	**YES**	**YES**	**YES**	Yes [20,24]
Petri nets	Yes [15]	No [13]	No [13]	No [10]

The results obtained in this paper are in boldface. 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 [15] in fact demonstrates decidability of the Reach problem (the Step problem is trivially decidable), hence our general strategy applies also in this case.

A unifying concept similar to R-N-bisimulation can also be used for simulation-like equivalences—we can define the R-N-simulation relation in the very same way as R-N-bisimulation (which can be then seen as a special case of R-N-simulation with the property that its inverse is also an R-N-simulation). The predicate N becomes more important in this context, as it allows to define some of the known and studied simulation-like equivalences (e.g., the ready simulation equivalence). An interesting open problem is whether it is possible to design a general strategy for deciding R-N-simulation equivalence between infinite-state and finite-state processes in a similar way as for R-N-bisimilarity (recently, the decidability/tractability border for strong simulation (i.e., S-T-simulation) with finite-state systems has been established in [17]). Another set of open problems is the decidability of branching bisimilarity with finite-state processes.

References

[1] *Proceedings of CONCUR'96*, volume 1119 of *Lecture Notes in Computer Science*. Springer, 1996.

[2] *Proceedings of CONCUR'97*, volume 1243 of *Lecture Notes in Computer Science*. Springer, 1997.

[3] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. *Journal of the Association for Computing Machinery*, 40:653–682, 1993.

[4] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

[5] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: application to model checking. In *Proceedings of CONCUR'97* [2], pages 135–150.

[6] D. Caucal. On the regular structure of prefix rewriting. *Theoretical Computer Science*, 106:61–86, 1992.

[7] I. Černá, M. Křetínský, and A. Kučera. Comparing expressibility of normed BPA and normed BPP processes. *Acta Informatica*, 36(3):233–256, 1999.

[8] S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation is decidable for all basic parallel processes. In *Proceedings of CONCUR'93*, volume 715 of *Lecture Notes in Computer Science*, pages 143–157. Springer, 1993.

[9] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, 121:143–148, 1995.

[10] J. Esparza. Decidability of model checking for infinite-state concurrent systems. *Acta Informatica*, 34:85–107, 1997.

[11] P. Jančar. Decidability of a temporal logic problem for Petri nets. *Theoretical Computer Science*, 74:71–93, 1990.

[12] P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301, 1995.

[13] P. Jančar and J. Esparza. Deciding finiteness of Petri nets up to bisimilarity. In *Proceedings of ICALP'96*, volume 1099 of *Lecture Notes in Computer Science*, pages 478–489. Springer, 1996.

[14] P. Jančar and A. Kučera. Bisimilarity of processes with finite-state systems. *Electronic Notes in Theoretical Computer Science*, 9, 1997.

[15] P. Jančar and F. Moller. Checking regular properties of Petri nets. In *Proceedings of CONCUR'95*, volume 962 of *Lecture Notes in Computer Science*, pages 348–362. Springer, 1995.

[16] A. Kučera. On effective decomposability of sequential behaviours. *Theoretical Computer Science*. To appear.

[17] A. Kučera and R. Mayr. Simulation preorder on simple process algebras. In *Proceedings of ICALP'99*, volume 1644 of *Lecture Notes in Computer Science*, pages 503–512. Springer, 1999.

[18] A. Kučera and R. Mayr. Weak bisimilarity with infinite-state systems can be decided in polynomial time. In *Proceedings of CONCUR'99*, Lecture Notes in Computer Science. Springer, 1999. To appear.

[19] D. Lugiez and Ph. Schnoebelen. The regular viewpoint on PA-processes. In *Proceedings of CONCUR'98*, volume 1466 of *Lecture Notes in Computer Science*, pages 50–66. Springer, 1998.

[20] R. Mayr. Decidability of model checking with the temporal logic EF. *Theoretical Computer Science*. To appear.

[21] R. Mayr. Process rewrite systems. *Information and Computation*. To appear.

[22] R. Mayr. Weak bisimulation and model checking for basic parallel processes. In *Proceedings of FST&TCS'96*, volume 1180 of *Lecture Notes in Computer Science*, pages 88–99. Springer, 1996.

[23] R. Mayr. Model checking PA-processes. In *Proceedings of CONCUR'97* [2], pages 332–346.

[24] R. Mayr. *Decidability and Complexity of Model Checking Problems for Infinite-State Systems*. PhD thesis, TU-München, 1998.

[25] R. Mayr. Strict lower bounds for model checking BPA. *Electronic Notes in Theoretical Computer Science*, 18, 1998.

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

[27] M.L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

[28] F. Moller. Infinite results. In *Proceedings of CONCUR'96* [1], pages 195–216.

[29] D.E. Muller and P.E. Schupp. The theory of ends, pushdown automata, and second order logic. *Theoretical Computer Science*, 37(1):51–75, 1985.

[30] D.M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings* 5th *GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

[31] W. Reisig. *Petri Nets—An Introduction*. Springer, 1985.

[32] Ph. Schnoebelen. Decomposable regular languages and the shuffle operator. *EATCS Bulletin*, 67:283–289, February 1999.

[33] B. Steffen and A. Ingólfsdóttir. Characteristic formulae for processes with divergence. *Information and Computation*, 110(1):149–163, 1994.

[34] C. Stirling. Decidability of bisimulation equivalence for normed pushdown processes. *Theoretical Computer Science*, 195:113–131, 1998.

[35] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the Association for Computing Machinery*, 43(3):555–600, 1996.

[36] I. Walukiewicz. Pushdown processes: Games and model checking. In *Proceedings of CAV'96*, volume 1102 of *Lecture Notes in Computer Science*. Springer, 1996.