Equivalence-Checking on Infinite-State Systems: Techniques and Results

ANTONÍN KUČERA∗

Faculty of Informatics, Masaryk University, Botanická 68a, CZ-602 00 Brno, Czech Republic.
(*e-mail:* tony@fi.muni.cz)

PETR JANČAR†

Dept. of Computer Science, FEI, Technical University of Ostrava, 17. listopadu 15, CZ-708 33 Ostrava, Czech Republic.
(*e-mail:* Petr.Jancar@vsb.cz)

submitted 28 November 2003; revised 20 September 2004; accepted 10 May 2005

Abstract

The paper presents a selection of recently developed and/or used techniques for equivalence-checking on infinite-state systems, and an up-to-date overview of existing results (as of September 2004).

1 Introduction

A *reactive system* is a system which continuously interacts with its environment and whose behavior is strongly influenced by this interaction. Reactive systems usually consist of several asynchronous (but communicating) processes which run in parallel. This asynchrony, together with unpredictable actions of the environment, contribute to a high degree of non-determinism. Another characteristic feature is divergence; a reactive system is often supposed to run forever, though its processes can be dynamically created and terminated. Since reactive systems control potentially dangerous devices like power plants, airports, weapon systems, etc., there is a strong need for rigorous methods which allow to *prove* correctness (or at least safety) of such systems.

Two popular approaches to formal verification of reactive systems are *model-checking* and *equivalence-checking*. In the model-checking approach, desired properties of the verified implementation are defined as a formula of a suitable modal logic, and then it is shown that (a formal model of) the implementation satisfies the formula. In the equivalence-checking approach, one constructs a formal model of the intended behavior of the verified system (called *specification*) and then it is shown that the implementation is equivalent to the specification.

∗ Supported by the research center Institute for Theoretical Computer Science (ITI), project No. 1M0021620808.
† Supported by the grant No. 1ET101940420 of the Czech national program "Information society".

A principal difficulty of automated formal verification is that reactive systems tend to have a very large state space. There are various strategies for tackling this problem. For example, the technique of *symbolic model-checking* introduced in (Burch et al. 1992) uses a symbolic state-space representation based on OBDD's (ordered binary decision diagrams). This method was successfully used for formal verification of hardware circuits. *Partial-order reduction* (as described, e.g., in (Clark et al. 1999)) enables a practical verification of concurrent software based on model-checking with the logic LTL. Though these methods handle systems with large state spaces, they are still limited to finite-state systems. However, many systems are (or should be seen as) unbounded, i.e., having a potentially infinite state space. For example, unbounded data types such as counters, stacks, channels, or queues, require an infinite number of states. Parametrized systems (e.g., N philosophers, N/M readers/writers, etc.) should also be seen as infinite-state if we want to show their correctness for every choice of parameters. Another example are systems with a dynamically evolving structure (e.g., mobile networks).

Model-checking and equivalence-checking on infinite-state systems is a popular research field which has been attracting attention for almost two decades. Consequently, the collection of achieved results is large and diverse today. There have been several surveys presenting various subfields of this research area, like (Moller 1996; Esparza 1997; Jančar and Moller 1999; Bouajjani 2001; Srba 2002a), including a major Handbook chapter (Burkart et al. 1999). This paper is intended as a contribution to the collection of surveys, and its aim is twofold. First, it presents a selection of some recently developed techniques for equivalence-checking on infinite-state systems which have not yet been fully covered in the existing surveys. The emphasis is on explaining the core of underlying principles rather than presenting full proofs of particular results. Second, the paper gives an up-to-date overview of existing results for equivalence-checking on infinite-state systems (as of September 2004).

The style of presentation adopted in this paper reflects the authors' intention to explain "proof techniques" rather than particular proofs. Ideally, this would be achieved by first formulating a given technique "abstractly", and then showing how it applies in concrete situations. In most cases, we provide a detailed explanation just for the "abstract" part, and then indicate how and where the principle can be applied without going much into details (just pointing to the relevant literature). When we feel that the abstract formulation is too vague, the functionality is demonstrated on concrete examples.

The paper is organized as follows. Section 2 contains basic definitions. Section 3 is devoted to the presentation of selected proof techniques. In particular, Section 3.1 presents general results about the relationship between simulation preorder/equivalence and bisimulation equivalence. Subsection 3.1.1 starts by a simple observation about a specific power of the defender in simulation games. This observation is then used in a general reduction scheme which allows to (efficiently) reduce bisimilarity problems to their simulation counterparts. In Subsection 3.1.2 it is shown that there is also a generic "reduction" of the simulation equivalence problem to the bisimilarity problem. Although this "reduction" is rarely effective (due to fundamental reasons), it reveals a simple and generic relationship between simulation equivalence and bisimilarity.

Section 3.2 is devoted to selected techniques which have recently been used to establish new decidability results and upper complexity bounds for equivalence-checking problems. In Subsection 3.2.1, the technique of *bisimulation bases* is recalled (in a somewhat "abstracted" form) and then it is shown how this technique applies to checking weak bisimilarity between infinite and finite-state systems. In Subsection 3.2.2, the problem of effective constructibility of *characteristic formulae* which express the equivalence with a given finite-state system is examined. First, well-known results about the constructibility of characteristic formulae in the modal μ-calculus are recalled. Then, it is shown how to construct characteristic formulae w.r.t. (strong and weak) bisimilarity in the simpler logic EF. In Subsection 3.2.3, the so-called *DD-functions* are presented. This is a recently discovered "tool" used for several decidability and complexity results.

In Section 3.3 we discuss techniques for undecidability and lower complexity bounds. A common principle which is used in almost all undecidability and hardness proofs for bisimilarity- and simulation-checking problems is the ability of the defender to "force" the attacker to perform a specific transition. The variant for simulation-checking is, in fact, discussed already in Subsection 3.1.1; a similar principle exists also for bisimilarity. Since the abstract formulation of the two techniques does not say much about their applicability, we demonstrate them on selected examples.

Section 4 contains an up-to-date overview of existing results.

2 Basic Definitions

The set of all non-negative integers $0, 1, 2, \ldots$ is denoted by $\mathbb{N}$. The symbol ω is used to denote an infinite amount.

The first step of formal verification is to create a *formal model* of the verified system. The low-level semantics of such a model is given by its associated *transition system*; in our framework we assume that transitions (between states) are labelled by actions taken from a finite set.

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

Processes are formally understood as states in transition systems; from now on we do not distinguish between "states" and "processes". The dynamics of processes, i.e., possible computational steps, are defined by the transition relation. We write $s \xrightarrow{a} t$ instead of $(s, a, t) \in \rightarrow$, and say that t is an a-successor of s. This notation is extended to finite strings over Act in the natural way. A state t is *reachable* from a state s, written $s \rightarrow^* t$, if there is $w \in Act^*$ such that $s \xrightarrow{w} t$. A transition system is *image-finite* if each state has only finitely many a-successors for every $a \in Act$. The *branching degree* of a transition system $\mathcal{T}$, denoted $d(\mathcal{T})$, is the least $k \in \mathbb{N}$

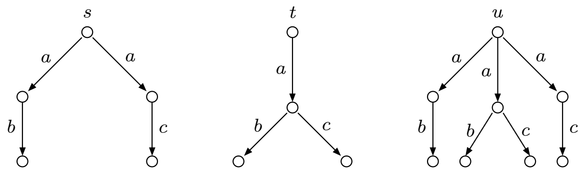


Fig. 1. Processes s, t, and u.

such that every state of $\mathcal{T}$ has at most k successors (if there is no such k then $d(\mathcal{T}) = \infty$).

2.1 Behavioral Equivalences

The notion of process equivalence can be formalized in many different ways (van Glabbeek 1999; van Glabbeek 1993). A straightforward idea is to employ the classical notion of language equivalence from automata theory (here we consider all states as accepting):

Definition 2
Let $\mathcal{T} = (S, Act, \rightarrow)$ be a transition system, $s \in S$. We say that $w \in Act^*$ is a *trace* of s iff $s \xrightarrow{w} s'$ for some s'. Let $tr(s)$ be the set of all traces of s. We write $s \sqsubseteq_{tr} t$ iff $tr(s) \subseteq tr(t)$. Moreover, we say that s and t are *trace equivalent*, written $s =_{tr} t$, iff $tr(s) = tr(t)$.

In concurrency theory, trace equivalence is usually considered as being too coarse. For example, the processes s and t of Fig. 1 are trace equivalent but their behavior is different—s can do either b or c (but not both) after performing a, while t can always choose between b and c after a. A finer level of "semantical sameness" of two processes can be defined by formalizing the ability of one process to "mimic" (or *simulate*) computational steps of another process.

Definition 3
Let $\mathcal{T} = (S, Act, \rightarrow)$ be a transition system, $s, t \in S$. A binary relation R over S is a *simulation* iff whenever $(s, t) \in R$ then for every $a \in Act$

if $s \xrightarrow{a} s'$ then $t \xrightarrow{a} t'$ for some t' such that $(s', t') \in R$.

A process s is *simulated* by a process t, written $s \sqsubseteq_{sm} t$, iff there is a simulation R such that $(s, t) \in R$. Note that the relation $\sqsubseteq_{sm}$ is a preorder. We say that s and t are *simulation equivalent*, written $s =_{sm} t$, iff $s \sqsubseteq_{sm} t$ and $t \sqsubseteq_{sm} s$.

For example, for processes of Fig. 1 we have that $s \sqsubseteq_{sm} t$, $t \not\sqsubseteq_{sm} s$, and $t =_{sm} u$.

Simulation preorder and equivalence can also be defined in terms of games (Stirling 2001; Thomas 1993). Imagine there are two tokens put on states s and t. Two players, the *attacker* and the *defender*, start to play a *simulation game* which consists of (possibly infinite) sequence of *rounds*, where each round is performed as follows:

1. the attacker takes the first token (the one which was put on s originally) and moves it along an arbitrary transition labeled by some $a \in Act$;

2. the defender has to respond by moving the other token along some transition with the same label a.

One player *wins* if the other player cannot move. Moreover, the defender wins every infinite play. It is easy to see that $s \sqsubseteq_{sm} t$ iff the defender has a universal winning strategy. Simulation equivalence can be understood similarly; we simply allow the attacker to choose his token at the beginning of the first round.

The finest (and probably the most important) behavioral equivalence we consider is *bisimulation equivalence* (Park 1981; Milner 1989).

Definition 4
Let $\mathcal{T} = (S, Act, \rightarrow)$ be a transition system, $s, t \in S$. A binary relation R over S is a *bisimulation* iff whenever $(s,t) \in R$ then for every $a \in Act$

- if $s \xrightarrow{a} s'$ then $t \xrightarrow{a} t'$ for some t' such that $(s', t') \in R$,
- if $t \xrightarrow{a} t'$ then $s \xrightarrow{a} s'$ for some s' such that $(s', t') \in R$.

Processes s, t are *bisimulation equivalent* (or *bisimilar*), written $s \sim t$, iff there is a bisimulation R such that $(s,t) \in R$.

A *bisimulation game* is defined in the same way as the simulation game. The only difference is that the attacker can *choose* his token at the beginning of *every* round (the defender has to respond with the other token). Again we have that $s \sim t$ iff the defender has a universal winning strategy in the bisimulation game initiated in s, t. For example, one can check that the processes s, t, u of Fig. 1 are pairwise non-bisimilar.

Internal computational steps which are not directly observable are by convention denoted by a special action τ. The notion of *weak bisimilarity* (Milner 1989) allows to "ignore" the internal steps to some extent.

Definition 5
Let $\mathcal{T} = (S, Act, \rightarrow)$ be a transition system. The *extended transition relation* $\Rightarrow \subseteq S \times Act \times S$ is defined as follows: $s \stackrel{a}{\Rightarrow} t$ iff one of the two conditions holds:

- $a \neq \tau$ and there are $s', s'' \in S$, $i, j \in \mathbb{N}$ such that $s \xrightarrow{\tau^i} s' \xrightarrow{a} s'' \xrightarrow{\tau^j} t$.
- $a = \tau$ and there is $i \in \mathbb{N}$ such that $s \xrightarrow{\tau^i} t$.

Here $s \xrightarrow{\tau^0} s'$ iff $s = s'$. In particular, this means that $s \stackrel{\tau}{\Rightarrow} s$ for every $s \in S$. A binary relation R over S is a *weak bisimulation* iff whenever $(s,t) \in R$ then for every $a \in Act$

- if $s \stackrel{a}{\Rightarrow} s'$ then $t \stackrel{a}{\Rightarrow} t'$ for some t' such that $(s', t') \in R$,
- if $t \stackrel{a}{\Rightarrow} t'$ then $s \stackrel{a}{\Rightarrow} s'$ for some s' such that $(s', t') \in R$.

Processes s, t are *weakly bisimulation equivalent* (or *weakly bisimilar*), written $s \approx t$, iff there is a weak bisimulation R such that $(s,t) \in R$.

A *weak bisimulation game* is defined in the same way as the bisimulation game, but both players now use the extended transitions.

We say that processes s and t are *bisimilar up to* $i \in \mathbb{N}$, written $s \sim_i t$, if the defender has a winning strategy for the first i rounds of the bisimulation game initiated in s and t. It is easy to see that $\sim_i$ is an equivalence relation and that $\sim_{i+1}$ refines $\sim_i$ for every $i \in \mathbb{N}$. Also note that $s \sim_0 t$ for all processes s, t. An important observation, taken from (Baeten et al. 1987), is

Theorem 1
Let $\mathcal{T} = (S, Act, \rightarrow)$ be a transition system and let s, t be processes of $\mathcal{T}$ such that each state t' reachable from t has only finitely many a-successors for every $a \in Act$ (note that there is no assumption about the process s). Then $s \sim t$ iff $s \sim_i t$ for every $i \in \mathbb{N}$.

Proof
The "$\Longrightarrow$" is obvious. For the other direction, one can check that the relation $R = \{(s', t') \mid (\forall i \in \mathbb{N} : s' \sim_i t') \wedge t \rightarrow^* t'\}$ is a bisimulation: Since t' has finitely many a-successors, for each $s' \xrightarrow{a} s''$ there must be some $t' \xrightarrow{a} t''$ such that $\forall i \in \mathbb{N} : s'' \sim_i t''$. Now consider a move $t' \xrightarrow{a} t''$. Obviously, for each $i \in \mathbb{N}$ there is $s' \xrightarrow{a} s_i$ such that $s_i \sim_i t''$. Each of the $s' \xrightarrow{a} s_i$ moves must be matched by some transition of t'. Since t' has only finitely many a-successors, there is a transition $t' \xrightarrow{a} t'''$ which was used infinitely many times. That is, there is an infinite sequence $s_{i_1}, s_{i_2}, \ldots$ such that for each s_{i_j} we have $\forall i \in \mathbb{N} : s_{i_j} \sim_i t'''$. This means $\forall i \in \mathbb{N} : t''' \sim_i t''$, and hence for every s_{i_j} we have $\forall i \in \mathbb{N} : s_{i_j} \sim_i t''$. $\square$

Weak bisimilarity up to $i \in \mathbb{N}$, denoted $\approx_i$, is defined in the same way (we use the weak bisimulation game). The aforementioned observations about $\sim_i$ are valid also for $\approx_i$ (incl. Theorem 1 where the a-successors are considered w.r.t. $\stackrel{a}{\Rightarrow}$).

Behavioral equivalences can also be used to relate processes of *different* transition systems. Formally, we can consider two transition systems to be a *single* one by taking their disjoint union (the labeling of transitions is preserved).

The relationship among the introduced equivalences is given by $=_{tr} \supset =_{sm} \supset \sim$. Weak bisimilarity properly subsumes $\sim$ and is incomparable with $=_{tr}$ and $=_{sm}$. (We do not consider weak versions of trace equivalence and simulation equivalence in this paper.) There are also other behavioral preorders and equivalences studied within the framework of concurrency theory. It seems, however, that trace, simulation, and especially (weak) bisimulation equivalence are of special importance as their accompanying theories are developed very intensively. Moreover, each equivalence in the linear/branching time spectrum of (van Glabbeek 1999) can be classified either as trace-like or as simulation-like. This means that $=_{tr}$, $=_{sm}$, and $\sim$ are good representatives for the whole spectrum; techniques and results achieved for these equivalences usually extend to others.

2.2 Formal Models of Infinite-State Systems

In this section we formally introduce some of the studied models of infinite-state systems. At a certain level of abstraction, most of them can be seen as various types of term rewriting systems. The structure of terms represents both control and data of the system, and the individual rewriting steps model atomic computational steps.

We start with the definition of a general process rewrite system (PRS) (Mayr

2000c). Then, we define various subclasses of PRS by imposing certain restrictions on the introduced formalism.

We assume a countable infinite set $\mathcal{C}$ of (process) *constants*. The abstract syntax of general *process expressions* is given by

$$E \quad ::= \quad X \quad | \quad \varepsilon \quad | \quad E \cdot E \quad | \quad E \| E$$

where the (meta)variable X ranges over $\mathcal{C}$ and ε denotes the empty expression. Intuitively, "$\cdot$" corresponds to sequencing, while "$\|$" models a simple form of parallelism. From now on we do not distinguish between expressions related by the *structural congruence*, which is the smallest congruence over $\mathcal{E}$ satisfying the following laws: "$\cdot$" and "$\|$" are associative, ε is the unit for both operators, and "$\|$" is also commutative.

The set of all process expressions is denoted by $\mathcal{E}$. The sets of *sequential* and *parallel* expressions, denoted $\mathcal{S}$ and $\mathcal{P}$, are formed by all process expressions which do not contain any "$\|$" and "$\cdot$", respectively. Observe that parallel expressions can also be seen as *multisets* of constants. Given $\mathcal{C}' \subseteq \mathcal{C}$, we use $\mathcal{S}(\mathcal{C}')$, $\mathcal{P}(\mathcal{C}')$, and $\mathcal{E}(\mathcal{C}')$ to denote the set of all sequential expressions, parallel expressions, and general expressions, respectively, which contain only the constants from $\mathcal{C}'$.

We also assume a countable infinite set $\mathcal{A}$ of *actions*, ranged over by $a, b, c, \ldots$. A *process rewrite system (PRS)* is a *finite* subset Δ of $\mathcal{E} \times \mathcal{A} \times \mathcal{E}$. Elements of Δ are called *rules* (a rule (α, a, β) is usually written $\alpha \xrightarrow{a} \beta$). Given a PRS Δ, we use $\mathcal{C}(\Delta)$ to denote the set of all constants appearing in the rules of Δ. We also use $\mathcal{S}(\Delta)$, $\mathcal{P}(\Delta)$, and $\mathcal{E}(\Delta)$ to denote $\mathcal{S}(\mathcal{C}(\Delta))$, $\mathcal{P}(\mathcal{C}(\Delta))$, and $\mathcal{E}(\mathcal{C}(\Delta))$ respectively. Moreover, $\mathcal{A}(\Delta)$ denotes the set of actions which are used in the rules of Δ.

Each PRS Δ determines a unique transition system $\mathcal{T}_\Delta$ where $\mathcal{E}(\Delta)$ is the set of states, $\mathcal{A}(\Delta)$ is the set of actions, and the transition relation is determined by the following inference rules (which should be understood modulo the structural congruence over expressions introduced above):

$$\frac{(E \xrightarrow{a} F) \in \Delta}{E \xrightarrow{a} F} \qquad \frac{E \xrightarrow{a} F}{E \cdot G \xrightarrow{a} F \cdot G} \qquad \frac{E \xrightarrow{a} F}{E \| G \xrightarrow{a} F \| G}$$

Various subclasses of PRS can be obtained by imposing certain restrictions on the form of the rules. Such a restriction is formally specified by a pair (A, B), where A and B are the subsets of expressions which can appear at the left-hand side and the right-hand side of rules, respectively. It has been argued in (Mayr 2000c) that "reasonable" restrictions should satisfy $A \subseteq B$. Moreover, if Δ is an (A, B)-restricted PRS, then the set of states of $\mathcal{T}_\Delta$ is restricted to $B \cap \mathcal{E}(\Delta)$. Some of the most important subclasses of PRS are listed below.

- *Finite state (FS) systems.* These are $(\mathcal{C}, \mathcal{C})$-restricted PRS which correspond to "ordinary" nondeterministic finite automata; the only difference is that there are no initial/final states.
- *BPA systems.* The restriction is $(\mathcal{C}, \mathcal{S})$. This model corresponds to the BPA (Basic Process Algebra) fragment of ACP (Baeten and Weijland 1990).
- *BPP systems.* The restriction is $(\mathcal{C}, \mathcal{P})$. BPP (Basic Parallel Processes) first appeared in the work (Christensen 1993).

- *PA systems*. The restriction is $(\mathcal{C}, \mathcal{E})$. PA (Process Algebra) systems subsume both BPA and BPP systems and correspond to another natural fragment of ACP (Baeten and Weijland 1990).
- *PDA systems*. The restriction is $(\mathcal{S}, \mathcal{S})$. It has been shown in (Caucal 1992) that every PDA system Δ can be efficiently transformed to a "normal form" Δ' where

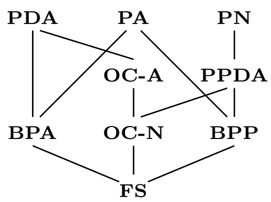
  — the set $\mathcal{C}(\Delta')$ can be partitioned into two disjoint subsets $\mathit{Control}(\Delta')$ and $\mathit{Stack}(\Delta')$;
  — the rules are of the form $p \cdot X \xrightarrow{a} q \cdot \beta$ where $p, q \in \mathit{Control}(\Delta')$, $X \in \mathit{Stack}(\Delta')$, and $\beta \in \mathcal{S}(\mathit{Stack}(\Delta'))$;
  — the set of states of $\mathcal{T}_{\Delta'}$ is restricted to those elements of $\mathcal{S}(\Delta')$ which are of the form $p \cdot \alpha$ where $p \in \mathit{Control}(\Delta')$ and $\alpha \in \mathcal{S}(\mathit{Stack}(\Delta'))$.

  Hence, PDA systems correspond to pushdown automata (Hopcroft and Ullman 1979). Consistently with the standard notation, we write $p\alpha$ instead of $p \cdot \alpha$. Observe that BPA can be also seen as PDA with just one control state.
- *PN systems*. The restriction is $(\mathcal{P}, \mathcal{P})$. PN systems correspond to the well-known model of Petri nets. Here the elements of $\mathcal{C}(\Delta)$ are referred to as *places* and the states of $\mathcal{T}_\Delta$ (i.e., multisets of places) as *markings*. In the rest of this paper we use the standard graphical representation of Petri nets to define PN systems—places are depicted as circles, and for every rule $X_1 \| \ldots \| X_n \xrightarrow{a} Y_1 \| \ldots \| Y_n$ we draw a new square labeled by "a". The square is connected to every X_i by an arrow pointing to the square, and to every Y_j by an arrow pointing to Y_j. For example, the middle part of Fig. 6 represents the rule $Q_i \| C_j \xrightarrow{dec} Q_l$, the right-hand part represents the rules $Q_i \xrightarrow{zer} Q_k$, $Q_i \| C_j \xrightarrow{zer} Q'_k \| C_j$ etc.
- *PPDA systems*. This is a subclass of PN known as "Parallel PushDown Automata" (Moller 1996). A system Δ is PPDA if the set $\mathcal{C}(\Delta)$ can be partitioned into two disjoint subsets $\mathit{Control}(\Delta)$ and $\mathit{Stack}(\Delta)$ so that every rule of Δ is of the form $p \| X \xrightarrow{a} q \| \beta$ where $p, q \in \mathit{Control}(\Delta)$, $X \in \mathit{Stack}(\Delta)$, and $\beta \in \mathcal{P}(\mathit{Stack}(\Delta))$.
  For a PPDA system Δ, the set of states of $\mathcal{T}_\Delta$ is restricted to those elements of $\mathcal{P}(\Delta)$ which are of the form $p \| \alpha$ where $p \in \mathit{Control}(\Delta)$ and $\alpha \in \mathcal{P}(\mathit{Stack}(\Delta))$. Usually we write $p\alpha$ instead of $p \| \alpha$.
- *OC-A systems*. These are PDA systems in normal form such that $\mathit{Stack}(\Delta) = \{I, Z\}$ and all transitions are of the form $pZ \xrightarrow{a} qI^iZ$ or $rI \xrightarrow{a} sI^j$, where $i, j \geq 0$. Here I^i denotes the sequential composition of i copies of the symbol I. The set of states of $\mathcal{T}_\Delta$ is restricted to $Q \times \{I^iZ \mid i \geq 0\}$. Hence, OC-A systems are one-counter automata where the counter ranges over nonnegative values. The counter can be incremented, decremented (if positive), and tested for zero.
- *OC-N systems*. These are OC-A systems which in addition satisfy the following condition: if $pZ \xrightarrow{a} qI^iZ$ is a rule of Δ, then also $pI \xrightarrow{a} qI^iI$ is a rule of Δ. In other words, there are no "zero-specific" transitions which could be used

to test the counter for zero. OC-N systems are equivalent to Petri nets with at most one unbounded place.

Let C be one of the just defined subclasses of PRS. A *C-process* is a state in $\mathcal{T}_\Delta$ where Δ is a member of C. The class of all C-processes is denoted $\mathbf{C}$. Important subclasses of BPA, BPP, and PA systems can be obtained by an extra condition of *normedness*. A BPA, BPP, or PA system Δ is *normed* if for every $X \in \mathcal{C}(\Delta)$ we have $X \rightarrow^* \varepsilon$. Hence, a system is normed if each of its processes can terminate via a finite number of transitions. The normed subclasses of BPA, BPP, and PA are denoted by **nBPA**, **nBPP**, and **nPA**, respectively.

Let $\leq$ be an ordering over process classes defined by $\mathbf{C}_1 \leq \mathbf{C}_2$ iff for every C_1-process there is a bisimilar C_2-process. The relationship among the introduced subclasses of processes (w.r.t. $\leq$) is shown in the following figure (we refer to (Moller 1996) for results about expressiveness).



Let $\simeq$ be a relation over processes. The problem of deciding $\simeq$ between processes of process classes $\mathbf{A}$ and $\mathbf{B}$ is denoted $\mathbf{A} \simeq \mathbf{B}$. For example, the problem of deciding bisimilarity between BPA and BPP processes is denoted $\mathbf{BPA} \sim \mathbf{BPP}$, and the problem of deciding simulation preorder between PA a FS processes is denoted $\mathbf{PA} \sqsubseteq_{sm} \mathbf{FS}$.

3 Some Recent Techniques and Results

In this section we explain some techniques which have recently been used to establish new decidability/complexity results for equivalence-checking on infinite-state systems. The material is divided into three (sub)sections. In Section 3.1 we explore the relationship between bisimilarity and simulation equivalence. Section 3.2 sketches some techniques for decidability and upper complexity bounds. Section 3.3 deals with techniques for undecidability and lower complexity bounds.

The generality and versatility of proof techniques is of course hard to measure. In the context of equivalence-checking on infinite-state systems, one good indication of a wider applicability of a given technique is a possibility to formulate its underlying principle in terms of transition systems (then we can say that the technique is "implemented" in a given syntax). However, such a formulation is not always possible despite a clear feeling that many proofs are just "instances" of the same idea. Here, we have to rely on an informal explanation and present an example which uses the technique in its simple and "clean" form.

3.1 The Relationship Between Simulation and Bisimulation

Since formal definitions of simulation and bisimulation are quite similar, a natural question is whether the decidability/complexity results achieved for one of the equivalences carry over to the other one. In this section we examine the question in greater detail.

3.1.1 Reducing Bisimilarity to Simulation Preorder/Equivalence.

According to the known decidability/complexity results for simulation and bisimilarity (which will be presented in Section 4), the problems $\mathbf{A} \sqsubseteq_{sm} \mathbf{B}$ and $\mathbf{A} =_{sm} \mathbf{B}$ are computationally harder than the problem $\mathbf{A} \sim \mathbf{B}$ for all major process classes $\mathbf{A}$ and $\mathbf{B}$. The aim of this section is to show that this is not a pure coincidence—there are general techniques which allow to (polynomially) reduce bisimilarity to simulation preorder/equivalence over many classes of infinite-state systems. The material presented in this section is based mainly on (Kučera and Mayr 2002d).

We start with a simple observation about a specific power of the defender in simulation games. Although the defender moves only his token during a play, his choice of a defending move can indirectly "force" the attacker to do a specific transition (with the attacker's token) in the next round. To illustrate this, we consider the first two rounds of the simulation game for the states s and t in the transition system of Fig. 2 (left and middle). After the attacker plays his only a-move, the

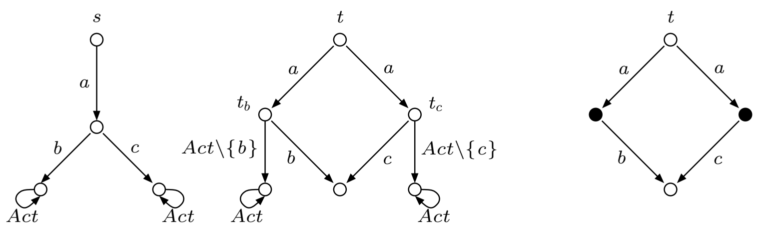


Fig. 2. The defender can enforce b or c in the second round.

defender can choose between moving to t_b or t_c. When he moves to t_b, he forces the attacker to use a b-move in the next round—if the attacker plays any other action, the defender moves to a state which enables all actions forever and therefore wins. Similarly, when the defender moves to t_c, he forces the attacker to use a c-move. We say that the b- and $c-$ transitions are *enforced* by t_b and t_c, respectively. To simplify our figures, we indicate the states which enforce the actions of their out-going transitions by black-filled circles. So, the middle part of Fig. 2 can be simplified to the right-hand part of Fig. 2.

The defender's ability to enforce the next attacker's transition is a crucial ingredient of several "hardness proofs" for simulation preorder/equivalence. (We address this issue in greater detail in Section 3.3 where we also deal with a similar technique for bisimilarity). Moreover, this was used in (Kučera and Mayr 2002d) to show that there are general "reduction schemes" allowing for efficient reductions of the $\mathbf{A} \sim \mathbf{B}$ problem to the $\mathbf{A} \sqsubseteq_{sm} \mathbf{B}$ problem for certain process classes $\mathbf{A}$ and $\mathbf{B}$.

More specifically, such a "reduction scheme" defines for every pair of processes s, t a new pair of processes s', t' so that $s \sim t$ iff $s' \sqsubseteq_{sm} t'$. The scheme is "applicable" to process classes $\mathbf{A}$ and $\mathbf{B}$ if for all processes $s \in \mathbf{A}$ and $t \in \mathbf{B}$ we have that the s' and t' are efficiently definable in the syntax of $\mathbf{A}$ and $\mathbf{B}$, respectively.

The existing reduction schemes are based on a possibility to emulate one round of the bisimulation game by one or two rounds of the simulation game. Here, the above discussed enforcing of transitions is used to emulate the "exchange of tokens" which can take place in the bisimulation game. To get a better idea on how this can be done, consider two states s, t of transition systems $\mathcal{S}$ and $\mathcal{T}$ which have the same set of actions Act and $\max\{d(\mathcal{S}), d(\mathcal{T})\} \leq 3$ (i.e., the branching degrees are at most 3). Further, let us suppose that s and t have just two successors s_1, s_2 and t_1, t_2, respectively (see top of Fig. 3). We show how to emulate one round of the bisimulation game initiated in s and t by at most two rounds of the simulation game initiated in (other) states s' and t' of transition systems $\mathcal{S}'$ and $\mathcal{T}'$ so that $s \sim t$ iff $s' \sqsubseteq_{sm} t'$.

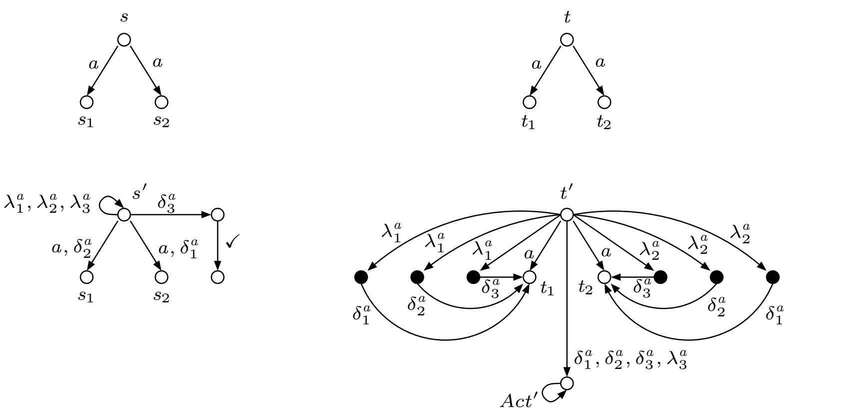


Fig. 3. The reduction of bisimilarity to simulation preorder. The systems $\mathcal{S}$ and $\mathcal{T}$ are in the first row (left and right, resp.), and the systems $\mathcal{S}'$ and $\mathcal{T}'$ are in the second row (left and right, resp.).

Here the systems $\mathcal{S}'$ and $\mathcal{T}'$ (see Fig. 3) are obtained just by extending $\mathcal{S}$ and $\mathcal{T}$ by other states and transitions labeled by fresh actions (the set of actions of $\mathcal{S}'$ and $\mathcal{T}'$ is denoted by Act'). The definition of $\mathcal{S}'$ (or $\mathcal{T}'$) depends just on $\mathcal{S}$ (or $\mathcal{T}$), Act, and $\max\{d(\mathcal{S}), d(\mathcal{T})\}$. The rules of the bisimulation game allow the attacker to choose his token at the beginning of every round. If he plays with the token put on s (e.g., by performing $s \xrightarrow{a} s_1$), the emulation is trivial and takes just one round of the simulation game initiated in s' and t' (in our case, the attacker would play $s' \xrightarrow{a} s_1'$ and the defender could also just mimic the response from the bisimulation game between s and t). Now suppose that the attacker takes the other token and plays, e.g., $t \xrightarrow{a} t_2$. In this case, the emulation is slightly more complicated and takes two rounds. First, the attacker performs the λ_2^a-loop on s'. By doing so, he in fact says that he wants to emulate the second a-transition of t in $\mathcal{T}$ (hence, the λ has a and 2 as its upper and lower index, respectively). To enable that the attacker can emulate moves from any state (not just t), we provide $\max\{d(\mathcal{S}), d(\mathcal{T})\}$ distinct λ_i^x-loops for each action $x \in Act$. In Fig. 3 we indicated just those successors of s' and t' which handle the action a; if there was another $b \in Act$, there would be a family of analogously constructed λ_i^b and δ_i^b transitions of s' and t' even if s and t have no outgoing b-transitions. As a response to the λ_2^a-loop played by the attacker, the defender can choose a state which enforces either δ_1^a, δ_2^a, or δ_3^a. Intuitively, he says that he wants to emulate the move to the first/second/third a-successor of s in $\mathcal{S}$. The δ_3^a is needed because the defender must be able to act accordingly for any position of the attacker's token. This finishes the first round, i.e., the first emulation phase where each of the two players makes his choice. The purpose of the second round is to ensure that the resulting position of tokens (after performing the second round) really corresponds to the choice which has been made. In our scenario, the attacker is forced to play the chosen δ_i^a action; and the only possibility available to the defender is to go to the state which was previously selected by the λ_2^a action, i.e., to t_2'.

If one of the two players cheats in the first round by trying to emulate a transition which does not really exist in s or t, the other player wins. For example, if the attacker performs the λ_3^a-loop on s' (i.e., he chooses the third a-successor of t which does not exist), the defender can respond by going to a state which can simulate everything. Similarly, if the attacker plays λ_1^a and the defender enforces δ_3^a, the attacker wins in two rounds by performing δ_3^a and then $\checkmark$. It follows that $s \sim t$ iff $s' \sqsubseteq_{sm} t'$.

The above scheme is applicable to process classes **A** and **B** if the syntax of **A** and **B** allows to "test for non-enabledness" of transitions. Examples include PDA, BPA, OC-A, 1-safe Petri nets, finite-state automata, etc. This means that, e.g., the problem **PDA** $\sim$ **FS** is polynomially reducible to **PDA** $\sqsubseteq_{sm}$ **FS** and **FS** $\sqsubseteq_{sm}$ **PDA**. Moreover, simulation preorder is easily reducible to simulation equivalence as follows: given processes s and t, we define other processes s' and t' which have (exactly) the transitions $s' \xrightarrow{a} s$, $s' \xrightarrow{a} t$, and $t' \xrightarrow{a} t$. We see that $s \sqsubseteq_{sm} t$ iff $s' =_{sm} t'$. This reduction is easily applicable to almost all process classes (thus, e.g., **PDA** $\sim$ **FS** is polynomially reducible to **PDA** $=_{sm}$ **FS**). However, there are also process classes to which the above scheme is not applicable. For example, general Petri nets cannot test a place for non-emptiness and therefore we cannot implement the families of λ and δ transitions in the syntax of Petri nets. However, the bisimilarity problem for Petri nets is still polynomially reducible to the problem of simulation preorder/equivalence by employing a *different* reduction scheme (also presented in (Kučera and Mayr 2002d)). There are also models (like, e.g., BPP or PA) where none of the known schemes works. An interesting question is if the existing schemes can be further generalized so that they cover all "reasonable" classes of infinite-state systems. A more detailed discussion can be found in (Kučera and Mayr 2002d).

3.1.2 Reducing Simulation Equivalence to Bisimilarity.

The results which will be presented in Section 4 indicate that there cannot be any general scheme for an efficient reduction of simulation equivalence to bisimilarity. Nevertheless, there *is* a general principle which can, in some sense, be seen as such a "reduction". Of course, this "reduction" is not effective in general. It can be effectively applied only in some restricted cases. Nevertheless, it also reveals an interesting relationship between simulation equivalence and bisimilarity and therefore we present it shortly. This subsection is based on (Kučera and Mayr 2002b).

Let $\mathcal{T} = (S, Act, \rightarrow)$ be an image-finite transition system. A transition $s \xrightarrow{a} t$ is *maximal* iff for every transition of the form $s \xrightarrow{a} t'$ we have that if $t \sqsubseteq_{sm} t'$ then also $t' \sqsubseteq_{sm} t$. In other words, $s \xrightarrow{a} t$ is maximal if t is maximal w.r.t. simulation preorder among all a-successors of s. Note that if the set of all a-successors of s is nonempty, there must be at least one maximal a-transition from s because $\mathcal{T}$ is image-finite. For example, the only maximal transition of the process u of Fig. 1 is the middle one.

Definition 6
Let $\mathcal{T} = (S, Act, \rightarrow)$ be an image-finite transition system. We define the system $\bar{\mathcal{T}} = (\bar{S}, Act, \mapsto)$ where $\bar{S} = \{\bar{s} \mid s \in S\}$ and $\bar{s} \overset{a}{\mapsto} \bar{t}$ iff $s \xrightarrow{a} t$ is a maximal transition of $\mathcal{T}$.

Hence, $\bar{\mathcal{T}}$ is obtained from $\mathcal{T}$ by renaming its states and deleting all non-maximal transitions. Now consider a simulation game between states s and $\bar{s}$. Intuitively, none of the two players can gain anything by using the non-maximal transitions because they are surely not the most optimal attacks/defenses. Thus, we obtain that $s =_{sm} \bar{s}$ for every $s \in S$. From this we immediately get that $s =_{sm} t$ iff $\bar{s} =_{sm} \bar{t}$ for all $s, t \in S$. Finally, note that if $\bar{s} =_{sm} \bar{t}$ then also $\bar{s} \sim \bar{t}$. To see this, one can readily check that the relation $R = \{(\bar{s}, \bar{t}) \mid \bar{s} =_{sm} \bar{t}\}$ is a bisimulation. As a simple consequence of presented observations, we obtain

Theorem 2
Let $\mathcal{T}$ be an image-finite transition system. For all $s, t \in S$ we have that $s =_{sm} t$ iff $\bar{s} \sim \bar{t}$, where $\bar{s}$ and $\bar{t}$ are the "twins" of s and t in $\bar{\mathcal{T}}$, respectively.

Using the previous theorem one can "reduce" certain simulation problems to their bisimulation counterparts. For example, instead of deciding simulation equivalence between s and t, we can (in principle) decide bisimilarity between $\bar{s}$ and $\bar{t}$. However, this "reduction" is rarely effective. If $\mathcal{T}$ is generated by a PRS Δ, one cannot compute another PRS $\bar{\Delta}$ which generates the system $\bar{\mathcal{T}}$ in general. It is not even clear if such a $\bar{\Delta}$ exists. Nevertheless, the effective construction is possible in some restricted cases. For example, if Δ is deterministic, then trivially $\bar{\Delta} = \Delta$. If Δ is a FS system, then $\bar{\Delta}$ is constructible in polynomial time because simulation preorder between the states of $\mathcal{T}_\Delta$ is computable in polynomial time. A less trivial example are OC-N systems—if Δ is an OC-N system, then $\bar{\Delta}$ is an effectively definable OC-A system (Jančar et al. 2000). Hence, certain simulation problems for OC-N processes are effectively reducible to the corresponding bisimulation problems over

OC-A processes, and the decidability of some of them has indeed been established in this way (Jančar et al. 2000).

3.2 Decidability and Upper Complexity Bounds

3.2.1 Bisimulation Bases.

The technique of *bisimulation bases* was pioneered by Caucal in (Caucal 1990). We start by explaining the underlying principle which is to some extent model-independent. The introduced notions are then illustrated on a concrete example. Finally, we show how the method applies to weak bisimilarity.

Since the "classical" results about bisimulation bases are carefully presented in (Burkart et al. 1999), we mention them just shortly. The main point of this section is the part about weak bisimilarity which is based on recent results (Kučera and Mayr 2002c).

Definition 7
Let $\mathcal{T}_1 = (S_1, Act, \to_1)$ and $\mathcal{T}_2 = (S_2, Act, \to_2)$ be two transition systems; we will write just $\to$ instead of $\to_1$, $\to_2$. Let $R \subseteq S_1 \times S_2$. We say that a *pair* $(s,t) \in S_1 \times S_2$ *expands in* R if

- for every $s \xrightarrow{a} s'$ there is some $t \xrightarrow{a} t'$ such that $(s',t') \in R$;
- for every $t \xrightarrow{a} t'$ there is some $s \xrightarrow{a} s'$ such that $(s',t') \in R$.

Now let $P, R \subseteq S_1 \times S_2$. We say that P *expands in* R if all pairs of P expand in R.

Let $\mathbf{C_1}$ and $\mathbf{C_2}$ be subclasses of process rewrite systems (not necessarily different), and let $\Delta_1 \in \mathbf{C_1}$ and $\Delta_2 \in \mathbf{C_2}$. Further, let

$$Bis = \{(\alpha, \beta) \mid \alpha \in \mathcal{T}_{\Delta_1}, \beta \in \mathcal{T}_{\Delta_2}, \alpha \sim \beta\}$$

be the bisimilarity relation between the processes of Δ_1 and Δ_2. A *bisimulation base* $\mathcal{B}$ (for Δ_1 and Δ_2) is a finite subset of Bis consisting only of "crucial" bisimilar pairs from which the whole relation Bis can be generated in some "syntactic" way. More precisely, one defines an operator Gen which for each relation $R \subseteq \mathcal{T}_{\Delta_1} \times \mathcal{T}_{\Delta_2}$ returns another relation $Gen(R) \subseteq \mathcal{T}_{\Delta_1} \times \mathcal{T}_{\Delta_2}$ so that the following conditions are satisfied:

(1) $Gen(\mathcal{B}) = Bis$.
(2) Gen is monotonic, i.e., if $R \subseteq R'$ then $Gen(R) \subseteq Gen(R')$.
(3) If R is a relation which expands in $Gen(R)$, then also $Gen(R)$ expands in $Gen(R)$. (In other words, if R expands in $Gen(R)$ then $Gen(R)$ is a bisimulation.)

Of course, finite bisimulation bases, and the associated Gen operators, exist only for *some* subclasses $\mathbf{C_1}$ and $\mathbf{C_2}$ of PRS. If the question whether $(\alpha, \beta) \in Gen(R)$ is semidecidable (R being finite), then the question whether R expands in $Gen(R)$ is also semidecidable. Therefore, the problem $\mathbf{C_1} \sim \mathbf{C_2}$ is semidecidable—to verify that $\alpha \sim \beta$, we can run a semidecision procedure which is guaranteed to find a finite relation R which expands in $Gen(R)$ and for which $(\alpha, \beta) \in Gen(R)$ (on condition that such a relation R exists). If $\alpha \sim \beta$, then this procedure halts because the finite base $\mathcal{B}$ must eventually be found (observe that $\mathcal{B}$ has all the required properties). And if the procedure halts because some relation R satisfying all of the required properties is found, we can conclude that $Gen(R)$ is a bisimulation (due to (3) above), hence $\alpha \sim \beta$.

Since the negative subcase $\mathbf{C_1} \not\sim \mathbf{C_2}$ is semidecidable due to generic reasons (see Theorem 1), we in fact obtain the decidability of the $\mathbf{C_1} \sim \mathbf{C_2}$ problem.

Now assume that the membership in $Gen(R)$ is even *decidable* for every R, and that for all Δ_1 and Δ_2 there is an effectively computable relation $\mathcal{G}$ which is guaranteed to subsume the base. Then the base is computable by the algorithm of Fig. 4. Note that if $\mathcal{B} \subseteq R$, then $\mathcal{B}$ expands in $Gen(R)$, because $\mathcal{B}$ expands in $Gen(\mathcal{B})$ and Gen is monotonic (see (2) above). This means that $\mathcal{B} \subseteq B$ is an invariant of the **repeat-until** loop of the algorithm of Fig. 4. Moreover, if $\mathcal{G}$ is computable in polynomial time (in the size of Δ_1 and Δ_2), and the membership in $Gen(R)$ is decidable in polynomial time, then the base is computable in polynomial time.

Input: Process Rewrite Systems $\Delta_1 \in \mathbf{C_1}$, $\Delta_2 \in \mathbf{C_2}$.
Output: The base $\mathcal{B}$.

$B := \mathcal{G}$;
repeat
  $R := B$; $B := \emptyset$
  for all $(\alpha, \beta) \in R$ **do**
    if (α, β) expands in $Gen(R)$ **then** $B := B \cup \{(\alpha, \beta)\}$ **fi**
  od;
until $B = R$
$\mathcal{B} := B$;

Fig. 4. An algorithm for computing $\mathcal{B}$

Example 1
If $\mathbf{C_1} = \mathbf{C_2} = \mathbf{nBPA}$ and $\Delta_1 = \Delta_2 = \Delta$, one can put

$$\mathcal{B} = \{(X, \alpha) \mid X \in \mathcal{C}(\Delta), \alpha \in \mathcal{S}(\Delta), X \sim \alpha\}$$

and $Gen(R) = Congr(R)$, where $Congr(R)$ is the least congruence over $\mathcal{S}(\Delta)$ w.r.t. "$\cdot$" subsuming R. The $\mathcal{B}$ can be over-approximated by a finite relation

$$\mathcal{G} = \{(X, \alpha) \mid X \in \mathcal{C}(\Delta), \alpha \in \mathcal{S}(\Delta), norm(X) = norm(\alpha)\}$$

where $norm(\alpha)$ is the length of the shortest sequence $w \in Act^*$ such that $\alpha \xrightarrow{w} \varepsilon$. Realize that $\mathcal{B}$ and $\mathcal{G}$ are finite relations because bisimilar processes must have the same norm and there are only finitely many processes with a given finite norm.

To get some idea on how all this works, let us prove that $Gen(\mathcal{B}) = Bis$. Clearly $Gen(\mathcal{B}) \subseteq Bis$, because bisimilarity is a congruence over $\mathcal{S}(\Delta)$ w.r.t. "$\cdot$". To prove $Bis \subseteq Gen(\mathcal{B})$, consider some $\alpha \sim \beta$; by induction on $norm(\alpha) = norm(\beta)$ we prove that $(\alpha, \beta) \in Gen(\mathcal{B})$. If $norm(\alpha) = 1$, then $\alpha = X$ for some X and hence $(\alpha, \beta) \in \mathcal{B}$. Now let $norm(\alpha) > 1$. Then $\alpha = X \cdot \gamma$ and $\beta = Y \cdot \delta$; let us assume that $norm(X) \leq norm(Y)$ (the other case is symmetric). Let $X \cdot \gamma \xrightarrow{w} \gamma$ where $length(w) = norm(X)$. The bisimilar process $Y \cdot \delta$ must be able to match this sequence of transitions by some $Y \cdot \delta \xrightarrow{w} \xi \cdot \delta$ so that $\gamma \sim \xi \cdot \delta$. Observe that $(\gamma, \xi \cdot \delta) \in Gen(\mathcal{B})$ by induction hypothesis. As $X \cdot \gamma \sim Y \cdot \delta$ and $\gamma \sim \xi \cdot \delta$, we also have $X \cdot \xi \cdot \delta \sim Y \cdot \delta$ and thus $X \cdot \xi \sim Y$ by applying the right cancellation law which is admitted by normed BPA processes. This means that $(Y, X \cdot \xi) \in \mathcal{B}$. To sum up, $(\gamma, \xi \cdot \delta) \in Gen(\mathcal{B})$ and $(Y, X \cdot \xi) \in \mathcal{B}$, which means that also $(X \cdot \gamma, Y \cdot \delta) \in Gen(\mathcal{B})$.

The operator *Gen* is clearly monotonic, and one can show that the condition (3) above is also satisfied.

From the previous example, it follows that the problem **nBPA** $\sim$ **nBPA** is decidable. This proof is essentially due to Caucal (Caucal 1990). Later, the structure of $\mathcal{B}$ was further simplified so that its size (and the size of $\mathcal{G}$) became *polynomial* in the size of Δ, and a suitable *Gen* was designed so that the algorithm of Fig. 4 terminates in *polynomial time* (Hirshfeld et al. 1996a). Hence, **nBPA** $\sim$ **nBPA** is in **P**. In (Christensen et al. 1995), it has been shown that a finite bisimulation base exists also for general (not necessarily normed) BPA processes. This implies the semidecidability (and hence also the decidability) of the **BPA** $\sim$ **BPA** problem. An algorithm for computing the bisimulation base for general BPA processes appeared in (Burkart et al. 1995), and this result led to an elementary upper complexity bound for the **BPA** $\sim$ **BPA** problem (a later result due to Srba (Srba 2002c) shows that the problem is **PSPACE**-hard).

Finite bisimulation bases exist also for BPP processes (Christensen et al. 1993). In the case of normed BPP processes, the base is small and can be computed in polynomial time (Hirshfeld et al. 1996b). The general problem **BPP** $\sim$ **BPP** is **PSPACE**-hard (Srba 2002b), and in fact **PSPACE**-complete (Jančar 2003) (see also Section 3.2.3).

The technique of bisimulation bases works also for weak bisimilarity, if the notion of expansion is modified as follows:

Definition 8
Let $\mathcal{T}_1 = (S_1, Act, \rightarrow)$ and $\mathcal{T}_2 = (S_2, Act, \rightarrow)$ be transition systems, and let $R \subseteq S_1 \times S_2$ be relations. A pair $(s, t) \in S_1 \times S_2$ *weakly expands* in R if

- for every $s \xrightarrow{a} s'$ there is some $t \stackrel{a}{\Rightarrow} t'$ such that $(s', t') \in R$;
- for every $t \xrightarrow{a} t'$ there is some $s \stackrel{a}{\Rightarrow} s'$ such that $(s', t') \in R$.

Let $P, R \subseteq S_1 \times S_2$. We say that P *weakly expands in* R if all pairs of P weakly expand in R.

The "asymmetry" which appears in the definition of weak expansion matches the original definition of weak bisimilarity used in (Milner 1989). The principle would work also for the "symmetric version" of weak expansion, but the introduced asymmetry leads to important algorithmic simplifications.

Example 2
Let $\mathbf{C_1} = \mathbf{BPA}$, $\mathbf{C_2} = \mathbf{FS}$, Δ be a BPA system and Δ_2 a FS system such that

$\mathcal{C}(\Delta) \cap \mathcal{C}(\Delta_2) = \emptyset$. For technical convenience, we put $\Delta_1 = \Delta \cup \Delta_2$. Note that Δ_1 is a BPA system. Now let

$$\begin{aligned} \mathcal{B} &= \{(AX, Y) \mid A \in \mathcal{C}(\Delta),\ X, Y \in \mathcal{C}(\Delta_2),\ AX \approx Y\} \\ &\cup \{(A, Y) \mid A \in \mathcal{C}(\Delta),\ Y \in \mathcal{C}(\Delta_2),\ A \approx Y\} \\ &\cup \{(\varepsilon, Y) \mid Y \in \mathcal{C}(\Delta_2),\ \varepsilon \approx Y\} \end{aligned}$$

Note that $\mathcal{B}$ can be over-approximated by a relation $\mathcal{G}$ of size $\mathcal{O}(|\Delta_1| \cdot |\Delta_2|^2)$ which consists of all syntactically conformable pairs.

For every relation $R \subseteq \mathcal{G}$ we define $Gen(R)$ to be the least relation K (between states of $\mathcal{T}_{\Delta_1}$ and states of $\mathcal{T}_{\Delta_2}$) subsuming R such that

- whenever $(\alpha X, Y) \in K$ and $(\beta, X) \in K$, then also $(\alpha\beta, Y) \in K$;
- whenever $(\beta, X) \in K$ where $norm(\beta) = \infty$, then also $(\beta\gamma, X) \in K$ for all $\gamma \in \mathcal{S}(\Delta_1)$.

One can readily check that $Gen(\mathcal{B}) = Bis$ and that Gen is monotonic. The proof that the condition (3) is also satisfied is more involved and can be found in (Kučera and Mayr 2002c).

Since the membership in $Gen(R)$ is easily decidable in polynomial time, one is tempted to conclude that the algorithm of Fig. 4 computes the base in polynomial time. This is indeed the case, but an additional problem has to be solved first. Let us consider, e.g., a pair of the form (A, Y) where $A \in \mathcal{C}(\Delta)$ and $Y \in \mathcal{C}(\Delta_2)$. According to Definition 8, (A, Y) weakly expands in $Gen(R)$ if for every "$\xrightarrow{a}$" move of one of the two processes there is a "$\overset{a}{\Rightarrow}$" move of the other process such that the resulting pair belongs to $Gen(R)$. The problem is that A can have *infinitely many* $\overset{a}{\Rightarrow}$ successors and hence we cannot simply try them one by one. If we denote $Reach^a_A = \{\alpha \mid A \overset{a}{\Rightarrow} \alpha\}$ and $Gen_X(R) = \{\alpha \mid (\alpha, X) \in Gen(R)\}$, the question whether for a given $Y \xrightarrow{a} X$ there is some $A \overset{a}{\Rightarrow} \alpha$ such that $(\alpha, X) \in Gen(R)$ reduces to the problem of checking whether $Reach^a_A \cap Gen_X(R) = \emptyset$. Since both sets can be infinite, the key is to find a suitable finite representation for them. In this case, it suffices to employ finite-state automata—both sets are regular and the associated finite-state automata are small and efficiently computable. Now the emptiness of $Reach^a_A \cap Gen_X(R)$ can be decided in polynomial time by standard methods of automata theory (Hopcroft and Ullman 1979).

The details can be found in (Kučera and Mayr 2002c), where a similar method is used to show that also the problem **nBPP** $\sim$ **FS** is decidable in polynomial time. In this case, the set of states which are reachable from a given BPP process in one "$\overset{a}{\Rightarrow}$" move is represented by a context-free grammar. Since the structure of the base is still regular, one can rely on the standard result saying that the emptiness of the intersection of a given CF-language and a given regular language can be decided in polynomial time. Recently, the method for BPA and FS processes described in Example 2 was generalized to PDA and FS systems and other behavioral equivalences (Kučera and Mayr 2004). In (Brázdil et al. 2004), it is shown that the technique of bisimulation bases is applicable also to probabilistic bisimilarity and probabilistic extensions of BPA, BPP, and PDA processes.

3.2.2 Characteristic Formulae for Finite-State Processes.

The problem of checking a given behavioral equivalence between an infinite-state process g and a finite-state specification f has recently been identified as an important subcase of the general equivalence-checking problem. There are two main reasons why this question attracts a special attention. First, in equivalence-based verification, one usually compares a "real-life" system with an abstract behavioral specification. A faithful model of the real-life system often requires features like counters, or subprocess creation, or unbounded buffers, that make the model infinite-state. On the other hand, the behavioral specification is usually abstract, hence naturally finite-state. Moreover, infinite-state systems are often abstracted to finite-state systems even before applying further analytical methods. This approach naturally subsumes the question if the constructed abstraction is correct (i.e., equivalent to the original system). The second reason is that checking equivalence between an infinite and a finite-state process is computationally easier than comparing two infinite-state processes (as also demonstrated by results of Section 4).

In this section we first recall the notion of a characteristic formula and show how to construct characteristic formulae in the modal μ-calculus (Steffen and Ingólfsdóttir 1994). Then, we concentrate on bisimulation-like equivalences. We present a simple theorem which reformulates the problem of bisimilarity between an infinite and a finite-state process to some kind of "reachability question". This approach originated in (Jančar and Moller 1995; Abdulla and Kindahl 1995; Jančar and Kučera 1997). A more abstract formulation which applies also to weak bisimilarity is due to (Jančar et al. 2001). Using this result, we show that characteristic formulae for finite-state systems w.r.t. bisimulation-like equivalences can also be constructed in the branching-time logic EF. This logic is much simpler than the modal μ-calculus, and consequently the model-checking problem with the logic EF is decidable for many classes of infinite-state systems. Thus, a number of decidability/complexity results about checking bisimilarity between infinite and finite-state processes have been obtained (Jančar et al. 2001).

Definition 9
Let $\mathcal{F} = (F, \mathit{Act}, \rightarrow)$ be a finite-state system, $f \in F$, and $\leftrightarrow$ an equivalence over the class of all processes. Let $\mathbf{C}_f$ be the class of all processes s such that the set of actions of s (in its underlying transition system) is included in Act. A formula φ is *characteristic* for f w.r.t. $\leftrightarrow$ if for every $s \in \mathbf{C}_f$ we have that $s \leftrightarrow f$ iff s satisfies φ.

Characteristic formulae w.r.t. $\sim_i$ (for given $i \in \mathbb{N}$ and Act) are easily definable in *Hennessy-Milner (H.M.) logic* (Milner 1989). The syntax of H.M. logic is given by

$$\varphi \quad ::= \quad \mathtt{tt} \quad | \quad \varphi \wedge \varphi \quad | \quad \neg\varphi \quad | \quad \langle a \rangle \varphi$$

where a ranges over actions. Formulae are interpreted over processes; the propositional connectives have the standard meaning and $s \models \langle a \rangle \varphi$ iff there is some $s \xrightarrow{a} t$ such that $t \models \varphi$. A formula $\neg\langle a \rangle \neg\varphi$ is usually abbreviated to $[a]\varphi$.

Now consider the transition system of Fig. 5. The behavior of f and h is described (up to bisimilarity) by the following recursively defined properties φ_f and

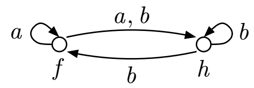


Fig. 5. Processes f and h.

φ_h, respectively.

$$\begin{array}{lcl}\varphi_f & \equiv & \langle a\rangle\varphi_f \ \wedge \ \langle a\rangle\varphi_h \ \wedge \ \langle b\rangle\varphi_h \ \wedge \ [a](\varphi_f \vee \varphi_h) \ \wedge \ [b]\varphi_h \\ \varphi_h & \equiv & \langle b\rangle\varphi_f \ \wedge \ \langle b\rangle\varphi_h \ \wedge \ [a]\mathtt{ff} \ \wedge \ [b](\varphi_f \vee \varphi_h)\end{array}$$

These equations can be used to construct characteristic formulae for f and h w.r.t. $\sim_i$; we inductively define the family of ξ_i^f and ξ_i^h formulae as follows:

$$\begin{array}{lcllcl}\xi_0^f & = & \mathtt{tt} & \qquad \xi_0^h & = & \mathtt{tt} \\ \xi_{i+1}^f & = & \varphi_f[\xi_i^f/\varphi_f, \xi_i^h/\varphi_h] & \qquad \xi_{i+1}^h & = & \varphi_h[\xi_i^f/\varphi_f, \xi_i^h/\varphi_h]\end{array}$$

Here $\varphi[\xi/\psi]$ denotes the formula obtained from φ by replacing each occurrence of subformula ψ with formula ξ. A straightforward proof confirms that for every process $s \in \mathbf{C}_f$ and $i \in \mathbb{N}$ we have that $s \sim_i f$ iff $s \models \xi_i^f$, and $s \sim_i h$ iff $s \models \xi_i^h$. By Theorem 1, this means that $\bigwedge_{i=0}^{\infty} \xi_i^f$ and $\bigwedge_{i=0}^{\infty} \xi_i^h$ are characteristic formulae for f and h w.r.t. $\sim$, respectively. These infinite conjunctions can be encoded in the modal μ-calculus (Kozen 1983) by translating the recursive dependence between φ_f and φ_h into an explicit greatest fixed-point definition; thus, we obtain the formula Φ^f.

$$\begin{array}{lcll}\Phi^f & \equiv & \nu S\cdot & \langle a\rangle S \ \wedge \ \langle a\rangle\varphi_h \ \wedge \ \langle b\rangle\varphi_h \ \wedge \ [a](S \vee \varphi_h) \ \wedge \ [b]\varphi_h \ \text{ where} \\ \varphi_h & \equiv & \nu T\cdot & \langle b\rangle S \ \wedge \ \langle b\rangle T \ \wedge \ [a]\mathtt{ff} \ \wedge \ [b](S \vee T)\end{array}$$

An analogous construction works also for weak bisimilarity. Instead of the "$\langle a\rangle$" modality of H.M. logic we employ its "weak form" $\langle\!\langle a\rangle\!\rangle$ defined by $\langle\!\langle a\rangle\!\rangle\varphi \equiv \Diamond_\tau\langle a\rangle\Diamond_\tau\varphi$ where $s \models \Diamond_\tau\varphi$ iff there is $s \overset{\tau}{\Rightarrow} t$ such that $t \models \varphi$. Since the "$\Diamond_\tau$" is expressible in the modal μ-calculus, one can construct characteristic formulae w.r.t. $\approx$ in this logic.

Characteristic formulae w.r.t. simulation equivalence are also easily definable in the modal μ-calculus. To see this, examine the recursively defined properties ψ_f, ψ_h and ϱ_f, ϱ_h:

$$\begin{array}{lcllcl}\psi_f & \equiv & \langle a\rangle\psi_f \ \wedge \ \langle a\rangle\psi_h \ \wedge \ \langle b\rangle\psi_h & \qquad \varrho_f & \equiv & [a](\varrho_f \vee \varrho_h) \ \wedge \ [b]\varrho_h \\ \psi_h & \equiv & \langle b\rangle\psi_f \ \wedge \ \langle b\rangle\psi_h & \qquad \varrho_h & \equiv & [a]\mathtt{ff} \ \wedge \ [b](\varrho_f \vee \varrho_h)\end{array}$$

A closer look reveals that for every $s \in \mathbf{C}_f$ we have $s \models \psi_f$ iff $f \sqsubseteq_{sm} s$, and $s \models \varrho_f$ iff $s \sqsubseteq_{sm} f$. Hence, $s =_{sm} f$ iff $s \models \psi_f \wedge \varrho_f$. The formulae ψ_f and ϱ_f can be encoded in the modal μ-calculus similarly as the formula φ_f above.

To sum up, the modal μ-calculus is sufficiently powerful to express characteristic formulae w.r.t. bisimilarity and simulation equivalence, and the size of these formulae is essentially the same as the size of the underlying transition system of f. Thus, the problem of checking bisimilarity and simulation equivalence with a finite-state process is polynomially reducible to the model-checking problem with the modal μ-calculus. This is applicable to PDA and BPA processes where model-checking the modal μ-calculus is known to be **EXPTIME**-complete (Walukiewicz 2001); hence, the problems **PDA** $\sim$ **FS**, **PDA** $\approx$ **FS**, **PDA** $\sqsubseteq_{sm}$ **FS**, **FS** $\sqsubseteq_{sm}$ **PDA**, and **PDA** $=_{sm}$ **FS** are in **EXPTIME**. The bounds for simulation are already tight, because these problems are also **EXPTIME**-hard (Kučera and Mayr 2002a). Actually, this holds even for BPA. However, we can do better for bisimilarity; the problems **PDA** $\sim$ **FS** and **PDA** $\approx$ **FS** are **PSPACE**-complete (Mayr 2000b; Kučera and Mayr 2002a). This requires an application of a different method which is described below.

If **C** is a class of processes such that $\sim_{i-1} = \sim_i$ over $\mathbf{C} \times \mathbf{C}$, then $\sim_i$ is a bisimulation relation and hence $\sim_{i-1} = \sim_i = \sim$ over $\mathbf{C} \times \mathbf{C}$. For example, if **C** is the set of processes of a finite-state transition system with k states, then surely $\sim_{k-1} = \sim_k$ because any equivalence over **C** has at most k equivalence classes and $\sim_{i+1} \subseteq \sim_i$ for every $i \in \mathbb{N}$. The same holds for $\approx_i$. The following theorem (Jančar et al. 2001) presents a simple (but important) observation about the problem of bisimilarity-checking with finite-state processes.

Theorem 3

Let $\mathcal{G} = (G, Act, \rightarrow)$ be a (general) transition system and $\mathcal{F} = (F, Act, \rightarrow)$ a finite-state transition system with k states. States $g \in G$ and $f \in F$ are bisimilar iff the following conditions hold:

- $g \sim_k f$;
- for each state g' such that $g \rightarrow^* g'$ there is a state $f' \in F$ such that $g' \sim_k f'$.

Proof

"$\Longrightarrow$" is obvious. To prove the "$\Longleftarrow$" direction, we show that the relation $R \subseteq G \times F$ given by

$$R = \{(g', f') \mid g \rightarrow^* g' \text{ and } g' \sim_k f'\}$$

is a bisimulation. Let $(g', f') \in R$ and let $g' \xrightarrow{a} g''$ for some $a \in Act$ (the case when $f' \xrightarrow{a} f''$ is handled in the same way). By definition of $\sim_k$, there is an f'' such that $f' \xrightarrow{a} f''$ and $g'' \sim_{k-1} f''$. It suffices to show that $g'' \sim_k f''$; as $g \rightarrow^* g''$, there is a state $\bar{f}$ of $\mathcal{F}$ such that $g'' \sim_k \bar{f}$. By transitivity of $\sim_{k-1}$ we have $\bar{f} \sim_{k-1} f''$, hence $\bar{f} \sim_k f''$ (remember that $\sim_{k-1} = \sim_k$ over $F \times F$). Now $g'' \sim_k \bar{f} \sim_k f''$ and thus $g'' \sim_k f''$ as required. Clearly $(g, f) \in R$ and the proof is finished. $\square$

The previous theorem holds also for weak bisimilarity (we use $\approx_k$ instead of $\sim_k$, and $\stackrel{a}{\Rightarrow}$ instead of $\xrightarrow{a}$).

Theorem 3 is applicable to a variety of models. Since $\sim_k$ is decidable for all "reasonably defined" classes of processes, the problem of bisimilarity-checking between infinite-state processes of a class **C** and finite-state processes reduces to a kind of reachability problem for **C**—all we need is an algorithm which, for a given process s of **C**, decides if s can reach a state s' which is not related by $\sim_k$ to any state of the considered finite-state system. In some cases, this is quite easy.

Example 3
Let $p\alpha$ be a PDA process. The behavior of PDA processes up to $\sim_k$ is determined by the current control state and the top k symbols of the stack. Hence, for all processes $q\beta$ where the length of β is bounded by k we do the following (re-using the computational space for each of the exponentially many $q\beta$'s): first we decide if there is some state f of the given finite-state system such that $q\beta \sim_k f$ (note that this can be done in polynomial space). If not, we either decide if $p\alpha \to^* q\beta$ (when $|\beta| < k$), or if $p\alpha \to^* q\beta\gamma$ for some γ (when $|\beta| = k$). This can be done in polynomial time by employing standard techniques for pushdown automata (Hopcroft and Ullman 1979). Thus, we obtain a polynomial-space algorithm for the problem **PDA** $\sim$ **FS** (the **PSPACE**-hardness is due to (Mayr 2000b)).

Similarly, one can handle other models like BPP, PA, or Petri nets; proofs are still simple but not completely immediate (Jančar and Moller 1995; Jančar and Kučera 1997).

With help of Theorem 3 one can also construct characteristic formulae w.r.t. strong and weak bisimilarity in the logic EF. This logic is obtained by extending the H.M. logic with the "$\Diamond$" (reachability) operator; $s \models \Diamond\varphi$ iff there is $s \to^* s'$ such that $s' \models \varphi$. For the construction of characteristic formulae w.r.t. $\approx$, we also need the aforementioned "$\Diamond_\tau$" operator to express the "$\langle\!\langle a \rangle\!\rangle$" modality. The dual operators are $\Box\varphi \equiv \neg\Diamond\neg\varphi$ and $\Box_\tau\varphi \equiv \neg\Diamond_\tau\neg\varphi$. A characteristic formula Φ^f for the process f of Fig. 5 w.r.t. $\sim$ (or $\approx$) in the logic EF looks as follows:

$$\Phi^f \quad \equiv \quad \xi_k^f \;\wedge\; \Box(\xi_k^f \vee \xi_k^h) \tag{1}$$

Here ξ_k^f and ξ_k^h are characteristic formulae for f and h w.r.t. $\sim_k$ (or $\approx_k$). Note that, in general, the size of the formula (1) is *exponential* in the size of the underlying transition system of f. However, the size of the DAG[1] representing this formula is only *polynomial*. This is important because the complexity of many model-checking algorithms depends on the size of the DAG rather then on the size of the formula itself. Moreover, the DAG representing Φ^f is computable in polynomial time. Thus, results about model-checking with the logic EF carry over to the problem of strong/weak bisimilarity with a finite-state process. For example, model-checking the logic EF is decidable for PA processes (Mayr 2001) (while model-checking the modal μ-calculus is undecidable already for BPP), and thus we obtain the decidability of **PA** $\sim$ **FS** and even **PA** $\approx$ **FS**. Since model-checking the logic EF for PDA is **PSPACE**-complete (Walukiewicz 2000), we obtain that the **PDA** $\sim$ **FS** and **PDA** $\approx$ **FS** problems are in **PSPACE** and hence **PSPACE**-complete (Kučera and Mayr 2002a).

Recently, Theorem 3 and the corresponding results about characteristic formulae have been generalized also to other behavioural equivalences (Kučera and Schnoebelen 2004).

[1] A DAG (directed acyclic graph or "circuit") representing a formula φ is obtained from the syntax tree of φ by identifying the nodes corresponding to the same subformula.

3.2.3 DD-functions

The technique of DD-functions was introduced in (Jančar 2003) in order to show that the problem **BPP** $\sim$ **BPP** is in **PSPACE**. Combined with Srba's result (Srba 2002b), **PSPACE**-completeness has thus been established. The technique of DD-functions was then also used in demonstrating the decidability of **BPA** $\sim$ **BPP** (Jančar et al. 2003).

Let $\mathcal{T} = (S, Act, \rightarrow)$ be a transition system. Stipulating that $\min \emptyset = \omega$, for all $s, t \in S$ we define the *distance from s to t* by

$$dist(s,t) = \min \{ \, length(w) \mid s \xrightarrow{w} t \, \}.$$

Here ω denotes an infinite amount. The set $\mathbb{N} \cup \{\omega\}$ is denoted $\mathbb{N}_\omega$, and we put $\omega - n = \omega$ for each $n \in \mathbb{N}_\omega$.

DD-functions are defined inductively. First, for every action a we define a function dd_a which, for every process s, gives the *"distance to disabling"* the action a. Formally,

$$dd_a(s) = \min \{ \, dist(s,t) \mid t \text{ has no } a\text{-successor} \, \}.$$

Given a tuple of (so far defined) DD-functions $\mathcal{F} = (d_1, \ldots, d_k)$, we observe that each transition $s \xrightarrow{a} t$ determines a *change* of $\mathcal{F}$, denoted $\mathcal{F}(t) - \mathcal{F}(s)$, which is a k-tuple of values from $\{-1\} \cup \mathbb{N}_\omega$ given by

$$\mathcal{F}(t) - \mathcal{F}(s) = \big(d_1(t) - d_1(s), \ldots, d_k(t) - d_k(s)\big).$$

Note that $d_i(s) = \omega$ implies $d_i(t) = \omega$. For technical reasons, we can then view $d_i(t) - d_i(s)$ as undefined, being interested only in changes of (so far) finite DD-functions.

The notion of change is used in the inductive step of the definition of DD-functions. For each triple $(a, \mathcal{F}, \delta)$, where a is an action, $\mathcal{F}$ is a k-tuple of DD-functions, and δ is a k-tuple of values from $\{-1\} \cup \mathbb{N}_\omega$, the function $dd_{(a,\mathcal{F},\delta)}$ (distance to disabling the action a causing the change δ of $\mathcal{F}$) is also a DD-function, defined by

$$dd_{(a,\mathcal{F},\delta)}(s) = \min \{ \, dist(s,t) \;\mid\; \forall r : \text{ if } t \xrightarrow{a} r \text{ then } \mathcal{F}(r) - \mathcal{F}(t) \neq \delta \, \}.$$

Here we (implicitly) assume that all functions from $\mathcal{F}$ are finite on t, which means that $\mathcal{F}(r) - \mathcal{F}(t)$ is defined. Note that the dd_a functions can be viewed as $dd_{(a,\mathcal{F},\delta)}$ where $\mathcal{F}$ and δ are the empty tuples (i.e., 0-tuples).

It is easy to show that all DD-functions are *bisimulation invariant*, i.e., $s \sim t$ implies $d(s) = d(t)$ for all DD-functions d. So, equality of the values of all DD-functions is a necessary condition for two states being bisimilar. For image-finite transition systems, this condition is also sufficient.

Let Δ be a BPP system. A key observation in (Jančar 2003) reveals that DD-functions on states of Δ coincide with *"norms"* w.r.t. effectively constructible subsets of $\mathcal{C}(\Delta)$. For all $Q \subseteq \mathcal{C}(\Delta)$ and $\alpha \in \mathcal{P}(\Delta)$ we define

$$\text{NORM}_Q(\alpha) = \min \{ \, dist(\alpha, \beta) \;\mid\; \beta \text{ does not contain any constant from } Q \, \}.$$

The result of (Jančar 2003) says that for every DD-function d there is some $Q \subseteq$

$\mathcal{C}(\Delta)$ such that $d(\alpha) = \textsc{norm}_Q(\alpha)$ for every $\alpha \in \mathcal{P}(\Delta)$. Since there are only finitely many subsets of $\mathcal{C}(\Delta)$, there are only finitely many DD-functions which are pairwise different on the states of Δ.

So, to find out if $\alpha \sim \beta$, it suffices to construct the relevant Q's and check whether $\textsc{norm}_Q(\alpha) = \textsc{norm}_Q(\beta)$ for each of them. Although there can be exponentially many relevant Q's, there is an algorithm performing the mentioned checking in polynomial space (Jančar 2003).

DD-functions were also used in (Jančar et al. 2003) to demonstrate the decidability of **BPA** $\sim$ **BPP**. A key point was to prove that DD-functions are *prefix-encoded* over BPA processes, which, roughly speaking, means that large finite values of DD-functions on BPA processes are tightly related to (i.e., represented by) large prefixes of these processes. More precisely, given a BPA system Δ, for each DD-function d there is a constant c such that if $c < d(X\alpha) < \omega$ and $X \rightarrow \gamma$ then $d(\gamma\alpha) - d(X\alpha) = \|\gamma\| - \|X\|$ (where $\|\cdot\|$ denotes the norm, i.e., $\|\beta\| = dist(\beta, \varepsilon)$). Hence, a BPA process cannot perform a (short) sequence of moves causing a different change of two large finite DD-values. We say that DD-functions are *dependent* over BPA processes, i.e., for every two DD-functions d_1, d_2 there is c such that if $c < d_1(\alpha) < \omega$, $c < d_2(\alpha) < \omega$ and $\alpha \rightarrow \beta$ then $d_1(\beta) - d_1(\alpha) = d_2(\beta) - d_2(\alpha)$.

If we are to find out whether $\alpha \sim \beta$ for a BPA process α and a BPP process β, we can proceed as follows. By using the above mentioned results from (Jančar 2003), one can use standard methods from Petri net theory to show that we can effectively check whether there are two DD-functions which are not dependent over the states reachable from β. If there are two such (independent) DD-functions then β is not bisimilar to any BPA process. If all DD-functions are (pairwise) dependent then we can show that there is a constant C such that for every γ reachable from β all finite DD-values which are larger than C coincide (i.e., if $c < d_1(\gamma) < \omega$ and $c < d_2(\gamma) < \omega$, then $d_1(\gamma) = d_2(\gamma)$). Hence, all "large" DD-values can be represented by a single number. One can even effectively construct a one-counter process β' which is bisimilar to β—the counter is used to represent the "large" DD-values, while "small" DD-values are remembered in the finite control unit. The process β' is generally not definable in the OC-A syntax, because there can be a need to reset the counter back to zero in a single transition (when the "large" DD-values change to ω). However, the reset can be easily modeled in PDA syntax by pushing a new bottom-of-stack symbol. Hence, β' can be seen as an (effectively definable) PDA process. In (Jančar et al. 2003), the decidability proof was finished by resorting to the involved result by Sénizergues (Sénizergues 1998) enabling to verify if $\alpha \sim \beta'$. (This "heavy machinery" is certainly not necessary for establishing the decidability of **BPA** $\sim$ **BPP**; the reduction was used just for technical convenience.)

3.3 Undecidability Results and Lower Complexity Bounds

Almost all existing undecidability and hardness proofs for simulation- and bisimilarity-checking take advantage of the defender's ability to (indirectly) *force* the attacker to do a specific transition. In a simulation game, the defender can "threaten" the attacker by a possibility to go to a universal state in the way indicated in Fig. 2 (see

Section 3.1.1 for further comments). A similar principle can be used also in bisimulation games. Here, the "threat" is based on a possibility to enter a *bisimilar* state. Consider processes s, t with transitions $s \xrightarrow{a} s'$, $t \xrightarrow{a} t'$, and $t \xrightarrow{a} t''$ where $s' \sim t'$. Under these assumptions, the move $t \xrightarrow{a} t''$ can be seen as the only (hopeful) option available to the attacker; the other options clearly lead to the defender's winning. This simple idea was used implicitly, e.g., in (Jančar 1995a). An explicit formulation is due to Srba (Srba 2003) who used this technique to establish **PSPACE**-hardness of the **BPP** $\sim$ **BPP** and **BPA** $\sim$ **BPA** problems (Srba 2002b; Srba 2002c).

To demonstrate the use (and power) of the above principles, we present selected undecidability and hardness proofs for concrete models. In Section 3.3.1 we show that the problem **PN** $\approx$ **PN** is *highly* undecidable (more concretely, Σ_1^1-complete), and that the problem **PA** $\sqsubseteq_{sm}$ **FS** is undecidable.

3.3.1 Encodings of Minsky Machines.

As can be expected, the undecidability results in the surveyed area have been obtained by reductions from the halting problem. As an example, we will recall the result for bisimilarity over Petri nets from (Jančar 1995b). This example is not really recent but we will expand it to show how the *high* undecidability result for *weak* bisimilarity from (Jančar 1995a) can be strengthened and made much more elegant using a recent technique of Srba (Srba 2004).

Minsky counter machines (with their halting problem) are a universal model which is technically convenient for our reduction. *A counter machine* $\mathcal{M}$ with non-negative counters $c_1, \cdots, c_m$ is a sequence of instructions

$$1 : \mathrm{INS}_1; \quad 2 : \mathrm{INS}_2; \quad \cdots \quad n{-}1 : \mathrm{INS}_{n-1}; \quad n : \mathtt{halt}$$

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

- $c_j := c_j + 1;$ `goto` k
- `if` $c_j = 0$ `then goto` k `else` $(c_j := c_j - 1;$ `goto` $l)$

Example 4
PN $\sim$ **PN** is undecidable.

Proof
Given a counter machine $\mathcal{M}$ with m counters and n instructions, we construct a Petri net $\mathcal{N}_\mathcal{M}$ with places $C_1, \ldots, C_m, Q_1, \ldots, Q_n, Q'_1, \ldots, Q'_n$. Intuitively, $C_1, \ldots, C_m$ correspond to the counters (the number of tokens in C_j represents the value of c_j) and $Q_1, \ldots, Q_n$ correspond to the control places (i.e., to the instructions)—the presence of the "control token" in Q_i means that INS_i is now to be performed. The places $Q'_1, \ldots, Q'_n$ are "copies" of the control places $Q_1, \ldots, Q_n$; their purpose becomes clear later. The (labelled) transitions of $\mathcal{N}_\mathcal{M}$ are constructed as follows.

- For each instruction $i : c_j := c_j + 1;$ `goto` k we add a transition depicted in Fig. 6 (left); an analogous transition will be also added for the "copy" places Q'_i, Q'_k.

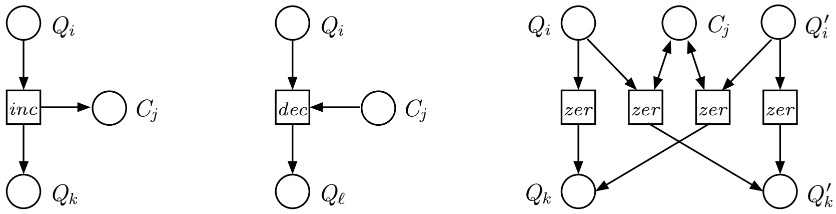
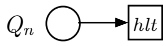


Fig. 6. Transitions of the Petri net $\mathcal{N}_\mathcal{M}$ of Example 4

- For each instruction i : `if` $c_j = 0$ `then goto` k `else` ($c_j := c_j - 1$; `goto` l) we add a transition depicted in Fig. 6 (middle), together with an analogous transition for Q'_i, Q'_ℓ. We also add four transitions with label *zer* as depicted in Fig. 6 (right). Note that the two "middle" *zer*-transitions can be performed only when C_j is positive but leave C_j unchanged.
- Finally, we add a transition



  which has no counterpart for Q'_n.

Having the constructed net $\mathcal{N}_\mathcal{M}$, it is a simple exercise to verify that the marking with one token in Q_1 and zero elsewhere is bisimilar to the marking with one token in Q'_1 and zero elsewhere iff the counter machine $\mathcal{M}$ halts for the zero initial values in the counters (which is an undecidable problem). In particular, observe the role of the previously mentioned forcing—if the attacker performs a move which does not correspond to a faithful simulation of $\mathcal{M}$ (i.e., uses a *zer*-transition when the respective c_j is nonzero), the defender can "punish" him by reaching an identical pair of markings (which is clearly a winning position for the defender). So, the only reasonable option for the attacker is to simulate the computation of the counter machine. The defender must mimic, and thus the attacker wins exactly when the machine halts. $\square$

The "level of undecidability" of $\mathbf{PN} \sim \mathbf{PN}$ is low; this is just a Π^0_1-complete problem in the arithmetical hierarchy (the negative subcase, i.e., the existence of a winning strategy for the attacker, is easily seen to be semidecidable). Perhaps somewhat surprisingly, the problem $\mathbf{PN} \approx \mathbf{PN}$ turns out to be highly undecidable. In (Jančar 1995a), it was shown that the problem is beyond the arithmetical hierarchy, though clearly in the class Σ^1_1 of the analytical hierarchy. Now we show that $\mathbf{PN} \approx \mathbf{PN}$ is in fact a Σ^1_1-complete problem. This is achieved by modifying the construction recently presented by Srba (Srba 2004).

A well-known Σ^1_1-complete problem is the question whether a given *nondeterministic* counter machine allows an infinite computation performing the first instruction infinitely often (the "recurrence problem"). Now we formulate another Σ^1_1-complete problem which better suits our purposes.

Consider "extended" Minsky machines which are defined in the same way as

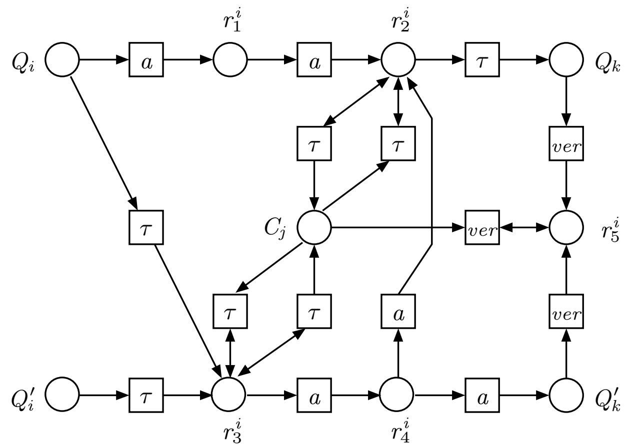


Fig. 7. Modelling the instruction i : `set` c_j ; `goto` k in Example 5

"ordinary" (deterministic) Minsky machines, but the instruction set is extended by allowing instructions of the form

i : `set` c_j ; `goto` k

The instruction `set` c_j sets the counter c_j to a nondeterministically chosen value (which can be an arbitrary nonnegative integer). Hence, we have unbounded nondeterminism. It is a routine programming exercise to show that the recurrence problem can be reduced to the problem if there is an infinite computation of our extended counter machine: The (bounded) nondeterminism can be easily simulated; and we can add a special counter *step* which is (programmed to be) set to an arbitrary value before each performing of the (original) first instruction, and is decremented before each other (original) instruction—if this is not possible (since *step* is 0), a jump to the halting state is performed.

Example 5
$\mathbf{PN} \approx \mathbf{PN}$ is Σ_1^1-complete.

Proof
Let $\mathcal{M}$ be an extended Minsky machine. We construct a Petri net $\mathcal{N}_{\mathcal{M}}$ by taking the same sets of places and transitions as in Example 4, and adding further auxiliary places and transitions to handle instructions of the form i : `set` c_j ; `goto` k. The places $(r_1^i, r_2^i, r_3^i, r_4^i, r_5^i)$ and transitions which are added for a given instruction i : `set` c_j ; `goto` k are shown in Fig. 7 (their role is explained in the following paragraphs).

Let us take two copies $\mathcal{N}, \mathcal{N}'$ of the constructed net $\mathcal{N}_{\mathcal{M}}$, and assume that the control token is in Q_i in $\mathcal{N}$ and in Q_i' in $\mathcal{N}'$, and the values of counters are the same in both nets. If the attacker wants to avoid reaching an identical pair of markings, he is forced to start by the a-move from Q_i in $\mathcal{N}$ (he moves the control token to r_1^i). The defender then has to move the control token in $\mathcal{N}'$ from Q_i' to r_4^i, via the place r_3^i. Observe that while having the control token in r_3^i, the defender could perform a sequence of the respective two τ-transitions and thus set any chosen value to C_j (in $\mathcal{N}'$). Now, when the control tokens are in r_1^i (in $\mathcal{N}$) and in r_4^i (in $\mathcal{N}'$), the attacker is forced to make the a-move in $\mathcal{N}'$, shifting the token from r_4^i to Q'_k (otherwise the defender could immediately reach an identical pair of markings). The defender answers by moving the token from r_1^i to Q_k (in $\mathcal{N}$) via r_2^i, where he can set C_j (in $\mathcal{N}$) to any chosen value. (We can safely assume that the instruction k is not another set-instruction and thus no τ-moves are possible from Q_k, Q'_k. The defender does not gain anything by leaving the token in r_2^i, because the attacker could move the token to Q_k in the next round anyway.) Now, the control tokens are in Q_k, Q'_k and it was the defender who set values to C_j in both $\mathcal{N}$, $\mathcal{N}'$. If the defender has set two different values, the attacker can obviously win by performing a sequence of actions *ver*. Otherwise, the correct simulation of a computation of $\mathcal{M}$ continues.

Hence, starting with markings M of $\mathcal{N}$ and M' of $\mathcal{N}'$, where M and M' has just a token in Q_1 and Q'_1, respectively, it is clear that $M \approx M'$ iff $\mathcal{M}$ has an infinite computation. $\square$

Reductions of the halting problem to simulation problems are usually simpler, because the constructed processes do not have to be "coupled" so tightly as in the case of bisimilarity. This is demonstrated in the last example of this subsection.

Example 6
PA $\sqsubseteq_{sm}$ **FS** is undecidable.

Proof
Let $\mathcal{M}$ be a counter machine with two counters initialized to zero. We construct a (deterministic) PA process $Z_1 \| Z_2$ and a deterministic FS process f_1 such that $Z_1 \| Z_2 \sqsubseteq_{sm} f_1$ iff $\mathcal{M}$ does not halt.

The rules of the underlying system of $Z_1 \| Z_2$ look as follows:

$$\begin{array}{llll} Z_1 \xrightarrow{z_1} Z_1, & Z_1 \xrightarrow{i_1} C_1 \cdot Z_1, & C_1 \xrightarrow{i_1} C_1 \cdot C_1, & C_1 \xrightarrow{d_1} \varepsilon, \\ Z_2 \xrightarrow{z_2} Z_2, & Z_2 \xrightarrow{i_2} C_2 \cdot Z_2, & C_2 \xrightarrow{i_2} C_2 \cdot C_2, & C_2 \xrightarrow{d_2} \varepsilon \end{array}$$

Hence, $Z_1 \| Z_2$ is a parallel composition of two counters initialized to zero. The underlying FS system Δ of f_1 corresponds to the finite control of $\mathcal{M}$. For every instruction of the form $i : c_j := c_j{+}1;$ `goto` k we have a rule $f_i \xrightarrow{i_j} f_k$. For every instruction of the form $i :$ `if` $c_j = 0$ `then goto` k `else` $c_j := c_j{-}1;$ `goto` l we have the rules $f_i \xrightarrow{z_j} f_k$ and $f_i \xrightarrow{d_j} f_l$. Then we "enforce" these transitions. That is,

- we add a new constant u together with rules $u \xrightarrow{a} u$ for every action a;
- for every f_i, where $i < n$, and every action a: If there is no rule $f_i \xrightarrow{a} f_j$ for any f_j, then we add a rule $f_i \xrightarrow{a} u$.

The attacker (who plays with $Z_1 \| Z_2$) can choose a counter and perform one of the available operations on it. Since the defender "enforces" the right choice, the only attacker's chance is to faithfully emulate the machine $\mathcal{M}$; if $\mathcal{M}$ halts, then the defender is eventually forced to enter the state f_n where he loses the game. Hence, $Z_1 \| Z_2 \sqsubseteq_{sm} f_1$ iff $\mathcal{M}$ does not halt. $\square$

3.3.2 Hardness Results.

The use of the "enforced" transitions in hardness proofs will be demonstrated on two examples. We show that the problems **PDA** $\sim$ **FS** and **PDA** $\sqsubseteq_{sm}$ **FS** are **PSPACE**-hard by reducing the QBF (Quantified Boolean Formula) problem to each of them. Our objective is to show what has to be done differently in the two respective cases, i.e., how the two "enforcing" techniques are implemented for the same models. (Note that the problems **PDA** $\sim$ **FS** and **PDA** $\sqsubseteq_{sm}$ **FS** are in fact **PSPACE**-complete and **EXPTIME**-complete, respectively (Kučera and Mayr 2002a)).

For the rest of this section, let us fix a quantified Boolean formula

$$\varphi \;\equiv\; \forall x_1 \exists x_2 \cdots \forall x_{n-1} \exists x_n : C_1 \wedge \cdots \wedge C_m$$

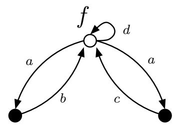
where every C_i is a clause, i.e., a disjunction of possibly negated propositions from $\{x_1, \ldots, x_n\}$. We can safely assume that n is even. The problem whether a given quantified Boolean formula holds is known to be **PSPACE**-complete; see, e.g., (Papadimitriou 1994).

Example 7
PDA $\sqsubseteq_{sm}$ **FS** is **PSPACE**-hard.

Proof
Let us consider a process gL_1Z of a PDA system with rules

- $gL_i \xrightarrow{a} gL_{i+1}X_i$, $gL_i \xrightarrow{a} gL_{i+1}\bar{X}_i$ for all odd i such that $1 \le i < n$;
- $gL_i \xrightarrow{b} gL_{i+1}X_i$, $gL_i \xrightarrow{c} gL_{i+1}\bar{X}_i$ for all even i such that $1 \le i \le n$;
- $gL_{n+1} \xrightarrow{d} c_j\varepsilon$ for every $1 \le j \le m$;
- $c_jX_i \xrightarrow{d} c_jX_i$, $c_j\bar{X}_i \xrightarrow{d} c_j\varepsilon$ for all $1 \le i \le n$ and $1 \le j \le m$ such that x_i appears in the clause C_j;
- $c_jX_i \xrightarrow{d} c_j\varepsilon$, $c_j\bar{X}_i \xrightarrow{d} c_j\bar{X}_i$ for all $1 \le i \le n$ and $1 \le j \le m$ such that $\neg x_i$ appears in the clause C_j;
- $c_jZ \xrightarrow{e} c_jZ$ for all $1 \le j \le m$.

We claim that the fixed quantified Boolean formula φ holds iff $gL_1Z \sqsubseteq_{sm} f$, where f is a finite-state process of the following system:



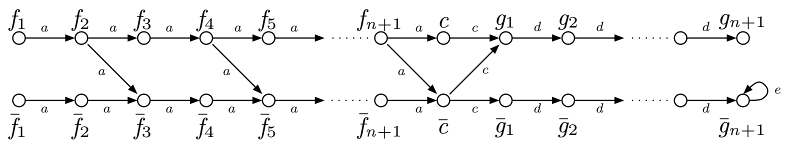
Here, the black-filled circles denote the states which enforce the actions of their outgoing transitions (see Section 3.1). Intuitively, the attacker (who plays with gL_1Z) is responsible for choosing the assignment for variables with odd index, while the defender (who plays with f) chooses the assignment for variables with even index by forcing the attacker to do b or c in the next round. After the guessing phase, the attacker chooses a clause by performing one of the $gL_{n+1} \xrightarrow{d} c_j\varepsilon$ transitions and starts to pop symbols from the stack, trying to find a symbol which witnesses the validity of the chosen clause. If no such symbol is found, the attacker eventually emits the action e and thus wins the game. Otherwise, he just performs an infinite number of d's and hence the defender wins. $\square$

Example 8
The problem **PDA** $\sim$ **FS** is **PSPACE**-hard.

Proof
For purposes of this proof, let us assume (wlog) that φ contains a clause which is true for every assignment. Let gL_1Z be a PDA process defined by

- $gL_i \xrightarrow{a} gL_{i+1}X_i$, $gL_i \xrightarrow{a} gL_{i+1}\bar{X}_i$ for all $1 \leq i \leq n$;
- $gL_{n+1} \xrightarrow{c} c_j\varepsilon$ for every $1 \leq j \leq m$;
- $c_jX_i \xrightarrow{d} p\varepsilon$, $c_j\bar{X}_i \xrightarrow{d} c_j\varepsilon$ for all $1 \leq i \leq n$ and $1 \leq j \leq m$ such that x_i appears in the clause C_j;
- $c_jX_i \xrightarrow{d} c_j\varepsilon$, $c_j\bar{X}_i \xrightarrow{d} p\varepsilon$ for all $1 \leq i \leq n$ and $1 \leq j \leq m$ such that $\neg x_i$ appears in the clause C_j;
- $pX_i \xrightarrow{d} p\varepsilon$, $p\bar{X}_i \xrightarrow{d} p\varepsilon$ for all $1 \leq i \leq n$;
- $c_jZ \xrightarrow{e} c_jZ$ for all $1 \leq j \leq m$.

Moreover, we also add transitions $gL_i \xrightarrow{a} \bar{f}_{i+1}L_i$ for every even i where $1 \leq i \leq n$, and another family of transitions which ensure that every process of the form $\bar{f}_{i+1}L_i\alpha$, where $1 \leq i \leq n$, is bisimilar to the state $\bar{f}_{i+1}$ in the following finite-state system:



We argue that φ holds iff $gL_1Z \sim f_1$. The "ideal" scenario for bisimulation game between the two processes looks as follows: the assignment for variables with odd index is chosen by the attacker who performs an appropriate a-move in the PDA process; the defender has to reply by the only available a-move in the finite-state system. If a variable x_i with an even index is to be assigned a value, the attacker performs the move $f_i \xrightarrow{a} f_{i+1}$ in the finite-state system. Now we distinguish two possibilities.

- the formula $\exists x_i \forall x_{i+1} \cdots \exists x_n : C_1 \wedge \cdots \wedge C_m$ is false after substituting each occurrence of x_j (for all $j < i$) with its previously assigned value. Then, the defender chooses some assignment for x_i by performing an a-move in the PDA process, but it does not really matter which one—from this point on, the attacker can always choose such an assignment for variables with odd index so that the above given formula is false for every even i. Hence, the attacker can enforce the game situation when one token is on c and the chosen assignment falsifies some clause C_j. Then, the attacker performs the transition $gL_{n+1} \xrightarrow{c} c_j\varepsilon$ and the defender has to respond by $c \xrightarrow{c} g_1$. Now, the attacker pops symbols from the stack, and since there is no symbol witnessing the validity of C_j, he eventually emits e and thus he wins.

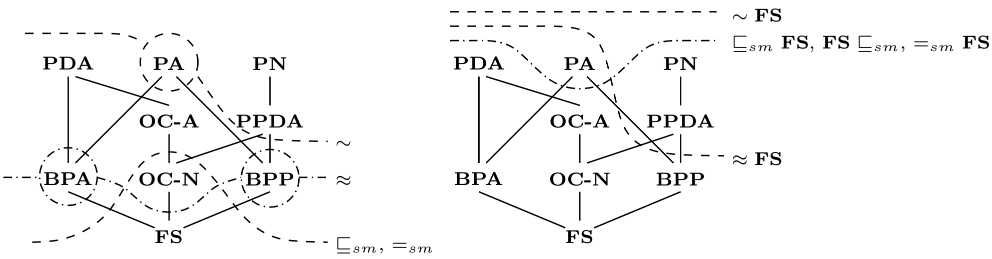


Fig. 8. The Decidability Border for Equivalence-Checking Problems

- otherwise, the defender chooses the "right" value for x_i, keeping a chance that the final assignment will satisfy all clauses. If the formula φ holds, he can thus enforce the game situation when one token is on c and the assignment stored in the PDA processes satisfies every clause C_j; it is easy to check that the defender wins the game from this configuration.

The construction ensures that the two players do not gain anything by violating the just specified scenario (a full justification requires a detailed analysis). For example, the attacker cannot use the transitions $f_i \xrightarrow{a} \bar{f}_{i+1}$ in the finite-state system because the defender could go to a bisimilar PDA state. $\square$

4 An Overview of Existing Results

In this section we give a brief overview of existing decidability and complexity results from the area of equivalence-checking on infinite-state processes. Results about the related *regularity* problem are also presented (given a process s and a behavioral equivalence $\leftrightarrow$, we ask if s is "regular", i.e., equivalent to some unspecified finite-state process).

The decidability border for equivalence-checking on infinite-state processes has already been determined for some behavioral equivalences. The left-hand part of Fig. 8 shows the decidability border for the problem $\mathbf{C} \leftrightarrow \mathbf{C}$, where $\mathbf{C}$ is a subclass of PRS and $\leftrightarrow$ one of the $\sim$, $\approx$, and $=_{sm}$ equivalences (the decidability of $\mathbf{PA} \sim \mathbf{PA}$, $\mathbf{BPA} \approx \mathbf{BPA}$, and $\mathbf{BPP} \approx \mathbf{BPP}$ is still open; this is indicated by dashed circles because it is not known whether the bordering line goes above or below the considered class). The right-hand side of Fig. 8 shows the decidability border for the $\mathbf{C} \leftrightarrow \mathbf{FS}$ problem. Detailed comments are split into several subsections.

4.1 Results for (Weak) Bisimilarity

4.1.1 Bisimilarity-Checking between Infinite-State Systems

The first result indicating that bisimilarity is "more decidable" than trace/language equivalence is due to Baeten, Bergstra, and Klop (Baeten et al. 1993) who established the decidability of bisimilarity for normed BPA processes. The proof is based on isolating a complex periodicity hidden in the structure of transition systems generated by normed BPA processes. A simpler proof of this result was later given by

Caucal in (Caucal 1990), where the technique of bisimulation bases was introduced. Another short proof is (Groote 1992). In (Hüttel and Stirling 1998), a sound and complete tableau-based deductive system for bisimilarity on normed BPA processes has been designed. The complexity of the problem was first addressed by Huynh and Tian (Huynh and Tian 1994) who gave a $\Sigma_2^P = \mathbf{NP}^{\mathbf{NP}}$ upper bound. Later, Hirshfeld, Jerrum, and Moller demonstrated that the problem is decidable in polynomial time (Hirshfeld et al. 1996a). The decidability result has been extended to all (not necessarily normed) BPA processes by Christensen, Hüttel, and Stirling in (Christensen et al. 1995). Again, it is shown that bisimilarity over all states of a given BPA system can be represented by a finite bisimulation base. As the decidability result is obtained by a combination of two semidecision procedures, it does not allow for any complexity estimations. An algorithm with elementary complexity was given in (Burkart et al. 1995) (the authors mention that some straightforward optimizations would lead to a doubly exponential algorithm). A technical core of the result is a procedure which computes a finite bisimulation base for general BPA processes. Recently, a **PSPACE** lower bound for the problem **BPA** $\sim$ **BPA** has been established by Srba in (Srba 2002c). The exact complexity classification is still missing.

The observation that bisimilarity over processes of a given BPP system is finitely generated by a bisimulation base is due to Christensen, Hirshfeld, and Moller (Christensen et al. 1993) who proved the decidability of bisimilarity for BPP processes. A polynomial-time algorithm for normed BPP processes has been given in (Hirshfeld et al. 1996b). The complexity of the general case was addressed by Mayr in (Mayr 2000a) who gave a **coNP**-lower bound for the problem, which has been improved to **PSPACE** by Srba (Srba 2002b). This result has recently been complemented by Jančar who gave a matching **PSPACE** upper complexity bound (Jančar 2003), which means that the **BPP** $\sim$ **BPP** problem is **PSPACE**-complete. When Jančar's algorithm is carefully implemented for normed BPP processes, it runs in time $O(n^3)$, as shown in (Jančar and Kot 2004).

The decidability of bisimilarity between normed BPA and normed BPP processes was proved by Blanco (Blanco 1995) and independently in (Černá et al. 1999). Later, the result was extended to parallel compositions of normed BPA and normed BPP processes in (Kučera 2000a). Recently, the decidability of **BPA** $\sim$ **BPP** has been established in (Jančar et al. 2003). A deep result (Hirshfeld and Jerrum 1999) due to Hirshfeld and Jerrum says that bisimilarity is decidable for normed PA processes. The proof is based on the unique decomposition property of normed processes w.r.t. "$\cdot$" and "$\|$", and hence the method is not applicable to general PA processes.

The semilinear structure of bisimilarity over one-counter processes has been identified in (Jančar 2000); it allows to conclude that bisimilarity is semidecidable (and thus decidable) for one-counter processes. However, the problem is computationally intractable even for one-counter nets—**DP**-hardness of **OC-N** $\sim$ **OC-N** was demonstrated in (Kučera 2003) (the class **DP** is expected to be somewhat larger than the union of **NP** and **coNP**). In (Sénizergues 1998), Sénizergues proved that bisimilarity is decidable for general PDA processes. This also extends a previous result due to Stirling (Stirling 1998) which says that bisimilarity is decidable for a subclass of PDA processes which can always empty their stack. Sénizergues's proof is obtained by adapting the method which previously led to the decidability of language equivalence for deterministic pushdown automata (Sénizergues 2001). Recently, Stirling presented a primitive recursive algorithm for the same problem (Stirling 2002). As for lower bounds, the **PDA** $\sim$ **PDA** problem is known to be **EXPTIME**-hard (Kučera and Mayr 2002a).

The undecidability of bisimilarity for Petri nets is due to Jančar (Jančar 1995b). In fact, the proof (see Example 4) also works for PPDA processes. A related undecidability result is (Schnoebelen 2001) where Schnoebelen proved that bisimilarity as well as other process equivalences are undecidable for lossy channel systems.

As for weak bisimilarity, many problems are still open. Weak bisimilarity is known to be semilinear, and thus semidecidable for BPP processes (Esparza 1995). Although the general case is still open, there is a decidability result for the subclass of *totally normed* BPP processes (Hirshfeld 1996) (a process is totally normed if it can reach ε in a finite sequence of transitions, but each such sequence must contain at least one action different from τ). The best known lower bound for the **BPP** $\approx$ **BPP** problem is **PSPACE** (Srba 2003), which is valid also for the normed subcase (previously, there was an **NP** (Stříbrná 1998) and $\Pi_2^P = \mathbf{coNP}^{\mathbf{NP}}$ lower bound (Mayr 2000a)). Weak bisimilarity between totally normed BPA processes is also decidable (Hirshfeld 1996). The problem **BPA** $\approx$ **BPA** is known to be **PSPACE**-hard (Stříbrná 1998), even in the normed subcase (Srba 2003). Recently, the lower complexity bound for weak bisimilarity on normed BPA has been improved to **EXPTIME** in (Mayr 2004). The problem **PDA** $\approx$ **PDA** is already undecidable (Srba 2002e). This result has been generalized in (Mayr 2003) where it is shown that even the problem **OC-N** $\approx$ **OC-N** is undecidable. An incomparable result of (Srba 2002d) shows that **PA** $\approx$ **PA** is also undecidable (Srba 2002d). Weak bisimilarity between Petri nets is even *highly* undecidable (i.e., beyond arithmetical hierarchy) (Jančar 1995a); this result has been strengthened to Σ_1^1-completeness and achieved also for PDA and PA in (Jančar and Srba 2004).

4.1.2 Bisimilarity-Checking between an Infinite and a Finite-State System

The problem has been considered in (Jančar and Moller 1995) where it is shown that **PN** $\sim$ **FS** is decidable. However, **PN** $\approx$ **FS** is already undecidable (Jančar and Esparza 1996). The decidability of **BPP** $\approx$ **FS** was shown in (Mayr 1996). Theorem 3 has been explicitly formulated in (Jančar and Kučera 1997) and (in a more abstract form) in (Jančar et al. 2001) where it is also shown that weak bisimilarity is decidable between so-called PAD processes and finite-state ones (the PAD class subsumes both PA and PDA processes). Complexity results followed—in (Kučera and Mayr 2002c) it was shown that the problems **BPA** $\approx$ **FS** and **nBPP** $\approx$ **FS** are solvable in polynomial time. The problem **BPP** $\approx$ **FS** is in **PSPACE** (Jančar et al. 2001), and the problem **BPP** $\sim$ **FS** is in **P**(Kot and Sawa 2004). The problem **PDA** $\sim$ **FS** is **PSPACE**-hard (Mayr 2000b), and the matching upper bound for **PDA** $\approx$ **FS** was given in (Kučera and Mayr 2002a), which means that the problems **PDA** $\sim$ **FS** and **PDA** $\approx$ **FS** are **PSPACE**-complete. Bisimilarity between one-counter processes and finite-state processes was studied in (Kučera 2003). It is shown that **OC-N** $\approx$ **FS** is **DP**-hard, while **OC-A** $\sim$ **FS** is solvable in polynomial time. The decidability of bisimilarity between lossy channel systems and finite-state systems is due to (Abdulla and Kindahl 1995). However, this problem (and in fact all non-trivial problems related to formal verification of lossy channel systems) are of nonprimitive recursive complexity (Schnoebelen 2002).

4.1.3 Regularity-Checking

The decidability of regularity w.r.t. $\sim$ for Petri nets is due to (Jančar and Esparza 1996). The regularity problem is also decidable for BPA processes (Burkart et al. 1996) and OC-A processes (Jančar 2000). For normed processes, regularity w.r.t. $\sim$ usually coincides with "syntactical boundedness", i.e., the question if a given process can reach infinitely many syntactically distinct states. This condition can be in some cases checked in polynomial time; it applies, e.g., to normed PA (Kučera 1999) and normed PDA processes. There are also some lower complexity bounds—regularity-checking w.r.t. $\sim$ is known to be **PSPACE**-hard for BPA (Srba 2002c) and BPP (Srba 2002b) (previously, there was **coNP**-lower bound for BPP (Mayr 2000a) and **PSPACE**-lower bound for PDA (Mayr 2000b)). For Petri nets, one can easily establish the **EXPSPACE**-lower bound by employing the simulation of a deterministic exponentially bounded machine due to Lipton (Lipton 1976). The problem is still open for general PA and PDA processes, though it is clearly semidecidable because bisimilarity with a (given) finite-state process is decidable for these models. Regularity w.r.t. $\approx$ is undecidable for Petri nets (Jančar and Esparza 1996) and **EXPTIME**-hard for PDA (Mayr 2004); for other major models of infinite-state systems, the problem remains open (it is again at least semidecidable by applying the same argument as above).

4.2 Results for Simulation and Trace Preorder/Equivalence

4.2.1 Simulation Preorder/Equivalence

As opposed to bisimilarity, simulation preorder/equivalence between infinite-state processes tends to be undecidable. Since trace preorder and simulation preorder coincide over deterministic processes, the undecidability of simulation preorder/equivalence for BPA processes follows immediately from Friedman's result (Friedman 1976) which says that the language inclusion problem for simple grammars is undecidable. As for BPP, simulation preorder/equivalence is also undecidable as shown by Hirshfeld (Hirshfeld 1994). The only known class of infinite-state processes where simulation preorder/equivalence remains decidable are one-counter nets. The result has been achieved by Abdulla and Čerāns (Abdulla and Čerāns 1998). A simpler proof was later given in (Jančar et al. 1999), where it is also shown that simulation preorder/equivalence for one-counter processes is already undecidable. A **DP** lower bound for the **OC-N** $\sqsubseteq_{sm}$ **OC-N** and **OC-N** $=_{sm}$ **OC-N** problems is given in (Jančar et al. 2004).

Deciding simulation between an infinite and a finite-state system is computationally easier. The decidability of **PN** $\sqsubseteq_{sm}$ **FS**, **FS** $\sqsubseteq_{sm}$ **PN** (and thus also **PN** $=_{sm}$ **FS**) is due to (Jančar and Moller 1995). Simulation between lossy channel systems and finite systems is also decidable (in both directions) (Abdulla and Kindahl 1995). The result of (Schnoebelen 2002) implies that this problem is of non-primitive recursive complexity A more general argument showing the decidability of simulation between processes of the so-called well-structured transition systems and finite-state processes has been presented in (Abdulla et al. 1996).

The decidability/tractability border for the problem has been established in (Kučera and Mayr 2002b). It is shown that **PDA** $\sqsubseteq_{sm}$ **FS** and **FS** $\sqsubseteq_{sm}$ **PDA** are in **EXPTIME**, and that **PA** $\sqsubseteq_{sm}$ **FS** and **FS** $\sqsubseteq_{sm}$ **PA** are already undecidable. Moreover, the following lower bounds are given: **FS** $\sqsubseteq_{sm}$ **BPA** and **FS** $\sqsubseteq_{sm}$ **BPP** are **PSPACE**-hard, and **BPA** $\sqsubseteq_{sm}$ **FS** and **BPP** $\sqsubseteq_{sm}$ **FS** (thus also for **BPA** $=_{sm}$ **FS** and **BPP** $=_{sm}$ **FS**) are **coNP**-hard. Recently (Kučera and Mayr 2002a), the simulation preorder/equivalence problem between a BPA/PDA process and a finite-state process was shown to be **EXPTIME**-complete (for both directions of simulation preorder). In this case, the only difference between PDA and BPA (from the complexity point of view) is that simulation preorder/equivalence between PDA and FS is **EXPTIME**-complete even for a *fixed* finite-state process, while simulation between a BPA and any fixed finite-state process f is decidable in polynomial time (Kučera and Mayr 2002a). Other tractable problems are **OC-N** $\sqsubseteq_{sm}$ **FS**, **FS** $\sqsubseteq_{sm}$ **OC-N**, and **OC-N** $=_{sm}$ **FS**, which are all decidable in polynomial time (Kučera 2000b). However, **OC-A** $\sqsubseteq_{sm}$ **FS**, **FS** $\sqsubseteq_{sm}$ **OC-A**, and **OC-A** $=_{sm}$ **FS** are already **DP**-hard (Kučera 2000b; Jančar et al. 2004). As for regularity-checking w.r.t. $=_{sm}$, the problem is known to be decidable for OC-N processes (Jančar et al. 2000), and undecidable for Petri nets (Jančar and Moller 1995) and PA processes (Kučera and Mayr 2002b).

4.2.2 Trace Preorder/Equivalence

Since trace preorder/equivalence are closely related to language inclusion/equivalence of automata theory (Hopcroft and Ullman 1979), all (un)decidability results about BPA and PDA processes follow easily from the "classical" ones. It means that almost all problems are undecidable; the only notable exception is the **PDA** $\sqsubseteq_{tr}$ **FS** problem which is decidable. The undecidability of trace preorder/equivalence between BPP processes is due to (Hirshfeld 1994).

Trace preorder/equivalence with a finite-state system is undecidable for BPA and PDA, but decidable for Petri nets; **PN** $\sqsubseteq_{tr}$ **FS** and **FS** $\sqsubseteq_{tr}$ **PN** are decidable as shown in (Jančar and Moller 1995). In the same paper it is shown that regularity w.r.t. $=_{tr}$ is undecidable for Petri nets.

4.2.3 Acknowledgment

We thank Jiří Srba for his many useful comments and suggestions.

References

Abdulla, P. and Čerāns, K. 1998. Simulation is decidable for one-counter nets. In *Proceedings of CONCUR'98*. Lecture Notes in Computer Science, vol. 1466. Springer, 253–268.

Abdulla, P., Čerāns, K., Jonsson, B., and Tsay, Y.-K. 1996. General decidability theorems for infinite-state systems. In *Proceedings of LICS'96*. IEEE Computer Society Press, 160–170.

Abdulla, P. and Kindahl, M. 1995. Decidability of simulation and bisimulation between lossy channel systems and finite state systems. In *Proceedings of CONCUR'95*. Lecture Notes in Computer Science, vol. 962. Springer, 333–347.

Baeten, J., Bergstra, J., and Klop, J. 1987. On the consistency of Koomen's fair abstraction rule. *Theoretical Computer Science 51,* 1, 129–176.

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

Baeten, J. and Weijland, W. 1990. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Blanco, J. 1995. Normed BPP and BPA. In *Proceedings of ACP'94*. Workshops in Computing. Springer, 242–251.

Bouajjani, A. 2001. Languages, rewriting systems, and verification of infinite-state systems. In *Proceedings of ICALP'2001*. Lecture Notes in Computer Science, vol. 2076. Springer, 24–39.

Brázdil, T., Kučera, A., and Stražovský, O. 2004. Deciding probabilistic bisimilarity over infinite-state probabilistic systems. In *Proceedings of CONCUR 2004*. Lecture Notes in Computer Science, vol. 3170. Springer, 193–208.

Burch, J., Clarke, E., McMillan, K., Dill, D., and Hwang, L. 1992. Symbolic model checking: 10^{20} states and beyond. *Information and Computation 98,* 2, 142–170.

Burkart, O., Caucal, D., Moller, F., and Steffen, B. 1999. Verification on infinite structures. *Handbook of Process Algebra*, 545–623.

Burkart, O., Caucal, D., and Steffen, B. 1995. An elementary decision procedure for arbitrary context-free processes. In *Proceedings of MFCS'95*. Lecture Notes in Computer Science, vol. 969. Springer, 423–433.

Burkart, O., Caucal, D., and Steffen, B. 1996. Bisimulation collapse and the process taxonomy. In *Proceedings of CONCUR'96*. Lecture Notes in Computer Science, vol. 1119. Springer, 247–262.

Caucal, D. 1990. Graphes canoniques des graphes algébriques. *Informatique Théorique et Applications (RAIRO) 24,* 4, 339–352.

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

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

Christensen, S. 1993. Decidability and decomposition in process algebras. Ph.D. thesis, The University of Edinburgh.

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

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

Clark, E., Grumberg, O., and Peled, D. 1999. *Model Checking*. The MIT Press.

Esparza, J. 1995. Petri nets, commutative context-free grammars, and basic parallel processes. In *Proceedings of FCT'95*. Lecture Notes in Computer Science, vol. 965. Springer, 221–232.

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

Friedman, E. 1976. The inclusion problem for simple languages. *Theoretical Computer Science 1,* 4, 297–316.

Groote, J. 1992. A short proof of the decidability of bisimulation for normed BPA processes. *Information Processing Letters 42*, 167–171.

Hirshfeld, Y. 1994. Petri nets and the equivalence problem. In *Proceedings of CSL'93*. Lecture Notes in Computer Science, vol. 832. Springer, 165–174.

Hirshfeld, Y. 1996. Bisimulation trees and the decidability of weak bisimulations. *Electronic Notes in Theoretical Computer Science 5*.

Hirshfeld, Y. and Jerrum, M. 1999. Bisimulation equivalence is decidable for normed process algebra. In *Proceedings of ICALP'99*. Lecture Notes in Computer Science, vol. 1644. Springer, 412–421.

Hirshfeld, Y., Jerrum, M., and Moller, F. 1996a. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theoretical Computer Science 158,* 1–2, 143–159.

Hirshfeld, Y., Jerrum, M., and Moller, F. 1996b. A polynomial algorithm for deciding bisimulation equivalence of normed basic parallel processes. *Mathematical Structures in Computer Science 6,* 3, 251–259.

Hopcroft, J. and Ullman, J. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Hüttel, H. and Stirling, C. 1998. Actions speak louder than words: Proving bisimilarity for context-free processes. *Journal of Logic and Computation 8,* 4, 485–509.

Huynh, D. and Tian, L. 1994. Deciding bisimilarity of normed context-free processes is in Σ_2^P. *Theoretical Computer Science 123*, 183–197.

Jančar, P. 1995a. High undecidability of weak bisimilarity for Petri nets. In *Proceedings of CAAP'95*. Lecture Notes in Computer Science, vol. 915. Springer, 349–363.

Jančar, P. 1995b. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science 148,* 2, 281–301.

Jančar, P. 2000. Decidability of bisimilarity for one-counter processes. *Information and Computation 158,* 1, 1–17.

Jančar, P. 2003. Strong bisimilarity on basic parallel processes is PSPACE-complete. In *Proceedings of LICS 2003*. IEEE Computer Society Press, 218–227.

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

Jančar, P. and Kot, M. 2004. Bisimilarity on normed Basic Parallel Processes can be decided in time $O(n^3)$. In *Proceedings of Third International Workshop on Automated Verification of Infinite-State Systems (AVIS'04)*.

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

Jančar, P., Kučera, A., and Mayr, R. 2001. Deciding bisimulation-like equivalences with finite-state processes. *Theoretical Computer Science 258,* 1–2, 409–433.

Jančar, P., Kučera, A., and Moller, F. 2000. Simulation and bisimulation over one-counter processes. In *Proceedings of STACS'2000*. Lecture Notes in Computer Science, vol. 1770. Springer, 334–345.

Jančar, P., Kučera, A., and Moller, F. 2003. Deciding bisimilarity between BPA and BPP processes. In *Proceedings of CONCUR 2003*. Lecture Notes in Computer Science, vol. 2761. Springer, 159–173.

Jančar, P., Kučera, A., Moller, F., and Sawa, Z. 2004. DP lower bounds for equivalence-checking and model-checking of one-counter automata. *Information and Computation 188,* 1, 1–19.

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

Jančar, P. and Moller, F. 1999. Techniques for decidability and undecidability of bisimilarity. In *Proceedings of CONCUR'99*. Lecture Notes in Computer Science, vol. 1664. Springer, 30–45.

Jančar, P., Moller, F., and Sawa, Z. 1999. Simulation problems for one-counter machines. In *Proceedings of SOFSEM'99*. Lecture Notes in Computer Science, vol. 1725. Springer, 404–413.

Jančar, P. and Srba, J. 2004. Highly undecidable questions for process algebras. In *Proceedings of IFIP TCS'2004*. Kluwer, 507–520.

Kot, M. and Sawa, Z. 2004. Bisimulation equivalence of a BPP and a finite state system can be decided in polynomial time. In *Proceedings of INFINITY'2004*, J. Bradfield and F. Moller, Eds. 73–82.

Kozen, D. 1983. Results on the propositional μ-calculus. *Theoretical Computer Science 27*, 333–354.

Kučera, A. 1999. Regularity of normed PA processes. *Information Processing Letters 72,* 1–2, 9–17.

Kučera, A. 2000a. Effective decomposability of sequential behaviours. *Theoretical Computer Science 242,* 1–2, 71–89.

Kučera, A. 2000b. On simulation-checking with sequential systems. In *Proceedings of ASIAN 2000*. Lecture Notes in Computer Science, vol. 1961. Springer, 133–148.

Kučera, A. 2003. The complexity of bisimilarity-checking for one-counter processes. *Theoretical Computer Science 304,* 1–3, 157–183.

Kučera, A. and Mayr, R. 2002a. On the complexity of semantic equivalences for pushdown automata and BPA. In *Proceedings of MFCS 2002*. Lecture Notes in Computer Science, vol. 2420. Springer, 433–445.

Kučera, A. and Mayr, R. 2002b. Simulation preorder over simple process algebras. *Information and Computation 173,* 2, 184–198.

Kučera, A. and Mayr, R. 2002c. Weak bisimilarity between finite-state systems and BPA or normed BPP is decidable in polynomial time. *Theoretical Computer Science 270,* 1–2, 677–700.

Kučera, A. and Mayr, R. 2002d. Why is simulation harder than bisimulation? In *Proceedings of CONCUR 2002*. Lecture Notes in Computer Science, vol. 2421. Springer, 594–609.

Kučera, A. and Mayr, R. 2004. A generic framework for checking semantic equivalences between pushdown automata and finite-state automata. In *Proceedings of IFIP TCS'2004*. Kluwer, 395–408.

Kučera, A. and Schnoebelen, P. 2004. A general approach to comparing infinite-state systems with their finite-state specifications. In *Proceedings of CONCUR 2004*. Lecture Notes in Computer Science, vol. 3170. Springer, 371–386.

Lipton, R. 1976. The reachability problem requires exponential space. Technical report 62, Yale University.

Mayr, R. 1996. Weak bisimulation and model checking for basic parallel processes. In

Proceedings of FST&TCS'96. Lecture Notes in Computer Science, vol. 1180. Springer, 88–99.

Mayr, R. 2000a. On the complexity of bisimulation problems for basic parallel processes. In *Proceedings of ICALP 2000*. Lecture Notes in Computer Science, vol. 1853. Springer, 329–341.

Mayr, R. 2000b. On the complexity of bisimulation problems for pushdown automata. In *Proceedings of IFIP TCS'2000*. Lecture Notes in Computer Science, vol. 1872. Springer, 474–488.

Mayr, R. 2000c. Process rewrite systems. *Information and Computation 156,* 1, 264–286.

Mayr, R. 2001. Decidability of model checking with the temporal logic EF. *Theoretical Computer Science 256,* 1–2, 31–62.

Mayr, R. 2003. Undecidability of weak bisimulation equivalence for 1-counter processes. In *Proceedings of ICALP 2003*. Lecture Notes in Computer Science, vol. 2719. Springer, 570–583.

Mayr, R. 2004. Weak bisimilarity and regularity of context-free processes is EXPTIME-hard. *Electronic Notes in Theoretical Computer Science 96*, 153–172.

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

Moller, F. 1996. Infinite results. In *Proceedings of CONCUR'96*. Lecture Notes in Computer Science, vol. 1119. Springer, 195–216.

Papadimitriou, C. 1994. *Computational Complexity*. Addison-Wesley.

Park, D. 1981. Concurrency and automata on infinite sequences. In *Proceedings* 5^{th} *GI Conference*. Lecture Notes in Computer Science, vol. 104. Springer, 167–183.

Schnoebelen, P. 2001. Bisimulation and other undecidable equivalences for lossy channel systems. In *Proceedings of TACS'2001*. Lecture Notes in Computer Science, vol. 2215. Springer, 385–399.

Schnoebelen, P. 2002. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters 83,* 5, 251–261.

Sénizergues, G. 1998. Decidability of bisimulation equivalence for equational graphs of finite out-degree. In *Proceedings of FOCS'98*. IEEE Computer Society Press, 120–129.

Sénizergues, G. 2001. L(A)=L(B)? Decidability results from complete formal systems. *Theoretical Computer Science 251,* 1–2, 1–166.

Srba, J. 2002a. Roadmap of infinite results. *EATCS Bulletin 78*, 163–175.

Srba, J. 2002b. Strong bisimilarity and regularity of basic parallel processes is PSPACE-hard. In *Proceedings of STACS'2002*. Lecture Notes in Computer Science, vol. 2285. Springer, 535–546.

Srba, J. 2002c. Strong bisimilarity and regularity of basic process algebra is PSPACE-hard. In *Proceedings of ICALP 2002*. Lecture Notes in Computer Science, vol. 2380. Springer, 716–727.

Srba, J. 2002d. Undecidability of weak bisimilarity for PA-processes. In *Proceedings of DLT 2002*. Lecture Notes in Computer Science. Springer, 197–208.

Srba, J. 2002e. Undecidability of weak bisimilarity for pushdown processes. In *Proceedings of CONCUR 2002*. Lecture Notes in Computer Science, vol. 2421. Springer, 579–593.

Srba, J. 2003. Complexity of weak bisimilarity and regularity for BPA and BPP. *Mathematical Structures in Computer Science 13,* 4, 567–587.

Srba, J. 2004. Completeness results for undecidable bisimilarity problems. *Electronic Notes in Theoretical Computer Science 98*, 5–19.

Steffen, B. and Ingólfsdóttir, A. 1994. Characteristic formulae for processes with divergence. *Information and Computation 110,* 1, 149–163.

STIRLING, C. 1998. Decidability of bisimulation equivalence for normed pushdown processes. *Theoretical Computer Science 195*, 113–131.

STIRLING, C. 2001. *Modal and Temporal Properties of Processes*. Springer.

STIRLING, C. 2002. Deciding DPDA equivalence is primitive recursive. In *Proceedings of ICALP 2002*. Lecture Notes in Computer Science, vol. 2380. Springer, 821–832.

STŘÍBRNÁ, J. 1998. Hardness results for weak bisimilarity of simple process algebras. *Electronic Notes in Theoretical Computer Science 18*.

THOMAS, W. 1993. On the Ehrenfeucht-Fraïssé game in theoretical computer science. In *Proceedings of TAPSOFT'93*. Lecture Notes in Computer Science, vol. 668. Springer, 559–568.

VAN GLABBEEK, R. 1993. The linear time—branching time spectrum II: The semantics of sequential systems with silent moves. In *Proceedings of CONCUR'93*. Lecture Notes in Computer Science, vol. 715. Springer, 66–81.

VAN GLABBEEK, R. 1999. The linear time—branching time spectrum. *Handbook of Process Algebra*, 3–99.

WALUKIEWICZ, I. 2000. Model checking CTL properties of pushdown systems. In *Proceedings of FST&TCS'2000*. Lecture Notes in Computer Science, vol. 1974. Springer, 127–138.

WALUKIEWICZ, I. 2001. Pushdown processes: Games and model-checking. *Information and Computation 164*, 2, 234–263.