Simulation Preorder on Simple Process Algebras

Antonín Kučera*[1] and Richard Mayr[2]

[1] Faculty of Informatics MU, Botanická 68a, 60200 Brno, Czech Republic, tony@fi.muni.cz
[2] Institut für Informatik TUM, Arcisstr. 21, 80290 München, Germany, mayrri@in.tum.de

Abstract. We consider the problem of simulation preorder/equivalence between infinite-state processes and finite-state ones. We prove that simulation preorder (in both directions) and simulation equivalence are *intractable* between all major classes of infinite-state systems and finite-state ones. This result is obtained by showing that the problem whether a BPA (or BPP) process simulates a finite-state one is $PSPACE$-hard, and the other direction is co-$\mathcal{NP}$-hard; consequently, simulation equivalence between BPA (or BPP) and finite-state processes is also co-$\mathcal{NP}$-hard.
The *decidability border* for the mentioned problem is also established. Simulation preorder (in both directions) and simulation equivalence are decidable in $EXPTIME$ between pushdown processes and finite-state ones. On the other hand, simulation preorder is undecidable between PA and finite-state processes in both directions. The obtained results also hold for those PA and finite-state processes which are deterministic and normed, and thus immediately extend to trace preorder. Regularity (finiteness) w.r.t. simulation and trace equivalence is also shown to be undecidable for PA.
Finally, we describe a way how to utilize decidability of bisimulation problems to solve certain instances of undecidable simulation problems. We apply this method to BPP processes.

1 Introduction

We study the decidability and complexity of checking simulation preorder and equivalence between certain infinite-state systems and finite-state ones. The motivation is that the intended behavior of a process can often be easily specified by a finite-state system, while the actual implementation may contain components which are infinite-state (e.g. counters, buffers). The task of formal verification is to prove that the specification and the implementation are equivalent.

The same problem has been studied recently for strong and weak bisimilarity [11,14], and it has been shown that these equivalences are not only *decidable*, but also *tractable* between certain infinite-state processes and finite-state ones. Those issues (namely the complexity ones) are dramatically different from the 'symmetric' case when we compare two infinite-state processes. Here we consider (and answer) analogous questions for simulation, giving a complete overview (see Fig. 1).

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

The state of the art: Simulation preorder/equivalence is known to be undecidable for BPA [7] and BPP [9] processes. An interesting positive result is that simulation preorder (and hence also equivalence) is decidable for Petri nets with at most one unbounded place [1]. In [12] it is shown that simulation preorder between Petri nets and finite-state processes is *decidable* in both directions. Moreover, a related problem of *regularity* (finiteness) of Petri nets w.r.t. simulation equivalence is proved to be undecidable.

Our contribution: In Section 3 we concentrate on the complexity issues for simulation preorder and equivalence with finite-state processes. We prove that the problem whether a BPA (or BPP) process simulates a finite-state one is $PSPACE$-hard, and the other direction is co-$\mathcal{NP}$-hard. Consequently, simulation equivalence between BPA (or BPP) and finite-state processes is also co-$\mathcal{NP}$-hard. Hence, the main message of this section is that simulation with finite-state systems is unfortunately *intractable* for any studied class of infinite-state systems (assuming $\mathcal{P} \neq \mathcal{NP}$)—see Fig. 1. It contrasts sharply with the complexity issues for strong and weak bisimilarity; for example, weak bisimilarity between BPA and finite-state processes, and between normed BPP and finite-state processes is in $\mathcal{P}$ [14].

In Section 4 we establish the decidability border of Fig. 1. First we prove that simulation preorder between PDA processes and finite-state ones is *decidable* in $EXPTIME$ in both directions. Consequently, simulation equivalence is also in $EXPTIME$. Then we show that simulation preorder between PA and finite-state processes is *undecidable* in both directions. It is rather interesting that the undecidability results hold even for those PA and finite-state processes which are *deterministic* and *normed*. Simulation *equivalence* between such processes is decidable (it coincides with bisimilarity [11]); however, as soon as we allow just one nondeterministic state in PA processes, simulation equivalence becomes undecidable. We also show that all the obtained undecidability results can be formulated in a 'stronger' form—it is possible to *fix* a PA or a finite-state process in each of the mentioned undecidable problems. Then we demonstrate that regularity of (normed) PA processes w.r.t. simulation equivalence is also undecidable. Again, it contrasts with regularity w.r.t. bisimilarity for normed PA processes, which is decidable in polynomial time [13]. All the obtained undecidability results also hold for trace preorder and trace equivalence, and therefore they might be also interesting from a point of view of 'classical' automata theory (see the last section for further comments).

Finally, in Section 5 we study the relationship between bisimilarity and simulation equivalence. Our effort is motivated by a general trend that problems for bisimilarity (equivalence, regularity) are often decidable, but the corresponding problems for simulation equivalence are not. We propose a way how to use existing algorithms for 'bisimulation' problems to solve certain instances of the corresponding (and possibly undecidable) 'simulation' ones. Such techniques are interesting from a practical point of view, as only small instances of undecidable problems can be solved in an ad-hoc fashion, and some kind of computer support is absolutely necessary for problems of 'real' size.

In the last section we give a summary of existing results in the area of comparing infinite-state systems with finite-state ones and discuss language-theoretic aspects of the obtained results.

The missing proofs can be found in the full version of the paper [15].

2 Definitions

Let $Act = \{a, b, c, \ldots\}$ and $Const = \{X, Y, Z, \ldots\}$ be disjoint countably infinite sets of *actions* and *process constants*, respectively. The class of *general process expressions* G is defined by $E ::= \epsilon \mid X \mid E \| E \mid E.E$, where $X \in Const$ and ϵ is a special constant that denotes the empty expression. Intuitively, '.' is a sequential composition and '$\|$' is a parallel composition. We do not distinguish between expressions related by *structural congruence* which is given by the following laws: '.' and '$\|$' are associative, '$\|$' is commutative, and 'ϵ' is a unit for '.' and '$\|$'.

A *process rewrite system* [16] is specified by a finite set Δ of *rules* which have the form $E \xrightarrow{a} F$, where $E, F \in G$ and $a \in Act$. $Const(\Delta)$ and $Act(\Delta)$ denote the sets of process constants and actions which are used in the rules of Δ, respectively (note that these sets are finite). Each process rewrite system Δ defines a unique transition system where states are process expressions over $Const(\Delta)$, $Act(\Delta)$ is the set of labels, and transitions are determined by Δ and the following inference rules (remember that '$\|$' is commutative):

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

We extend the notation $E \xrightarrow{a} F$ to elements of Act^* in a standard way. Moreover, we say that F is *reachable* from E if $E \xrightarrow{w} F$ for some $w \in Act^*$.

Various subclasses of process rewrite systems can be obtained by imposing certain restrictions on the form of rules. To specify those restrictions, we first define the classes S and P of *sequential* and *parallel* expressions, composed of all process expressions which do not contain the '$\|$' and the '.' operator, respectively. We also use 1 to denote the set of process constants. The hierarchy of process rewrite systems is presented in Fig. 1; the restrictions are specified by a pair (A, B), where A and B are the classes of expressions which can appear on the left-hand and the right-hand side of rules, respectively. This hierarchy contains almost all classes of infinite state systems which have been studied so far; BPA, BPP, and PA processes are well-known [2], PDA correspond to pushdown processes (as proved by Caucal in [4]), PN correspond to Petri nets, etc. In Fig. 1 we also indicated the decidability/tractability border for simulation preorder and equivalence with finite-state systems which is established in the following sections.

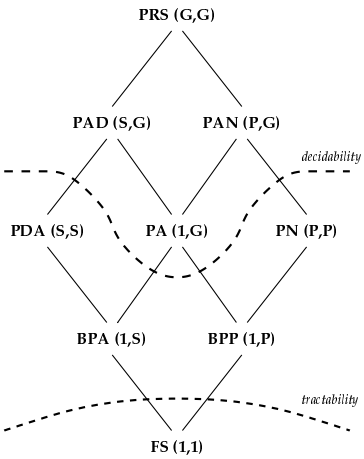

Fig. 1. A hierarchy of PRS

Processes are considered as states in transition systems generated by process rewrite systems. We also assume that for each system Δ there is some distinguished process expression which is considered as the *initial state* of Δ. In what follows, we often identify process rewrite systems with their initial states. A process P is said to be *deterministic*

iff each reachable state of P has at most one a-successor for every $a \in Act$. A PA process Δ is *normed* iff $X \rightarrow^* \epsilon$ for every $X \in Const(\Delta)$.

In this paper we compare infinite-state processes with finite-state ones w.r.t. certain 'behavioral' preorders and equivalences.

Definition 1. *We say that $w \in Act^*$ is a* trace *of a process E iff $E \stackrel{w}{\rightarrow} E'$ for some E'. Let $Tr(E)$ be the set of all traces of E. We write $E \sqsubseteq_t F$ iff $Tr(E) \subseteq Tr(F)$. Moreover, we say that E and F are* trace equivalent, *written $A =_t B$, iff $Tr(E) = Tr(F)$.*

Trace preorder and equivalence are very similar to language inclusion and equivalence of 'classical' automata theory. In concurrency theory, trace equivalence is usually considered as being too coarse. A plethora of finer 'behavioral' equivalences have been proposed [19]. It seems that *simulation* and *bisimulation* equivalence are of special importance, as their accompanying theory has been developed very intensively.

Definition 2. *A binary relation R over process expressions is a* simulation *if whenever $(E, F) \in R$ then for each $a \in Act$: if $E \stackrel{a}{\rightarrow} E'$ then $F \stackrel{a}{\rightarrow} F'$ for some F' s.t. $(E', F') \in R$. A symmetric simulation is called* bisimulation. *A process E is* simulated *by a process F, written $E \sqsubseteq_s F$, if there is a simulation R s.t. $(E, F) \in R$. We say that E and F are* simulation equivalent, *written $E =_s F$, iff $E \sqsubseteq_s F$ and $F \sqsubseteq_s E$. Similarly, we say that E and F are* bisimilar *(or* bisimulation equivalent*), written $E \sim F$, iff there is a bisimulation relating them.*

Another natural (and studied) problem is decidability of *regularity* (i.e. 'semantical finiteness') of processes w.r.t. certain behavioral equivalences. A process E is *regular* w.r.t. bisimulation (or simulation, trace) equivalence iff there is a finite-state process F such that $E \sim F$ (or $E =_s F$, $E =_t F$, respectively).

Almost all undecidability results in this paper are obtained by reduction of the halting problem for Minsky counter machines (the halting problem is undecidable even for Minsky machines with two counters initialized to zero [17]).

Definition 3. A counter machine $\mathcal{M}$ *with nonnegative counters $c_1, c_2, \cdots, c_m$ is a sequence of instructions 1: INS$_1$, $\cdots$, k: INS$_k$ where $k \in \mathbb{N}$, INS$_k$ = `halt`, and every INS$_i$ ($1 \leq i < k$) is in one of the following forms (where $1 \leq l, l', l'' \leq k$, $1 \leq j \leq m$).*

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

3 The Tractability Border

In this section we show that the problem whether a BPA (or BPP) process simulates a finite-state one is $PSPACE$-hard. The other preorder is shown to be co-$\mathcal{NP}$-hard. Consequently, we also obtain co-$\mathcal{NP}$-hardness of simulation equivalence between BPA (or BPP) and finite-state processes. As simulation preorder and equivalence are easily decidable for finite-state processes in polynomial time, the tractability border for simulation preorder/equivalence with finite-state systems of Fig. 1 is established.

Theorem 1. *Let P be a BPA (or BPP) process, F a finite-state process. The problem whether $F \sqsubseteq_s P$ is $PSPACE$-hard.*

Theorem 2. *Let P be a BPA (or BPP) process, F a finite-state process. The problem whether $P \sqsubseteq_s F$ is co-$\mathcal{NP}$-hard.*

Theorem 3. *The problems of simulation equivalence between BPA and finite-state processes, and between BPP and finite-state processes are co-$\mathcal{NP}$-hard.*

4 The Decidability Border

In this section we establish the decidability border of Fig. 1. We show that simulation preorder (in both directions) and simulation equivalence with finite-state processes are decidable for PDA processes in $EXPTIME$. It is possible to reduce each of the mentioned problems to the model-checking problem for an (almost) fixed formula φ of the alternation-free modal μ-calculus (we would like to thank Javier Esparza who observed the idea of our proof).

Then we turn our attention to PA processes. We prove that simulation preorder is *undecidable* between PA processes and finite-state ones in both directions. It is somewhat surprising, as for the subclasses BPP and BPA we have positive decidability results. Moreover, simulation preorder is undecidable even if we consider those PA and finite-state processes which are *deterministic* and *normed*. Thus, our undecidability results immediately extend to trace preorder (which coincides with simulation preorder on deterministic processes). It is worth noting that simulation *equivalence* between deterministic PA and deterministic finite-state processes is decidable, as it coincides with bisimilarity which is known to be decidable [11]. However, as soon as we allow just one nondeterministic state in PA processes, simulation equivalence with finite-state processes becomes undecidable (there is even a fixed normed deterministic finite-state process F such that simulation equivalence with F is undecidable for PA processes). The same applies to trace equivalence.

Finally, we also prove that regularity (finiteness) of PA processes w.r.t. simulation and trace equivalence is undecidable, even for the normed subclass of PA. Again, the role of nondeterminism is very special as regularity of normed deterministic PA processes w.r.t. simulation and trace equivalence coincides with regularity w.r.t. bisimilarity, which is easily decidable in polynomial time [13]. However, just one nondeterministic state in the PA process makes the undecidability proof possible.

Theorem 4. *Simulation preorder is decidable between PDA processes and finite-state ones in $EXPTIME$ (in both directions).*

Corollary 1. *Simulation equivalence between PDA and finite-state processes is decidable in $EXPTIME$.*

Theorem 5. *Let P be a deterministic PA process and F a deterministic finite-state process. It is undecidable whether $P \sqsubseteq_s F$.*

Proof. Let $\mathcal{M}$ be an arbitrary two-counter machine with counters initialized to m_1, m_2. We construct a deterministic PA process $P(\mathcal{M})$ and a deterministic finite-state process $F(\mathcal{M})$ s.t. $P(\mathcal{M}) \sqsubseteq_s F(\mathcal{M})$ iff the machine $\mathcal{M}$ does not halt. Let $Act :=$

$\{zero_1, inc_1, dec_1, zero_2, inc_2, dec_2\}$. The PA process $P(\mathcal{M})$ is defined by the following rules:

$$Z_1 \xrightarrow{zero_1} Z_1 \quad Z_1 \xrightarrow{inc_1} C_1.Z_1 \quad C_1 \xrightarrow{inc_1} C_1.C_1 \quad C_1 \xrightarrow{dec_1} \epsilon$$
$$Z_2 \xrightarrow{zero_2} Z_2 \quad Z_2 \xrightarrow{inc_2} C_2.Z_2 \quad C_2 \xrightarrow{inc_2} C_2.C_2 \quad C_2 \xrightarrow{dec_2} \epsilon$$

The initial state is $(C_1^{m_1}.Z_1) \parallel (C_2^{m_2}.Z_2)$.

The process $F(\mathcal{M})$ corresponds to the finite control of $\mathcal{M}$. For every instruction of the form $n: \ c_i := c_i + 1;$ `goto` n' we have an arc $n \xrightarrow{inc_i} n'$. For every instruction of the form $n:$ `if` $c_i = 0$ `then goto` n' `else` $c_i := c_i - 1;$ `goto` n'' `fi` we have arcs $n \xrightarrow{zero_i} n'$ and $n \xrightarrow{dec_i} n''$. Then we add a new state all and arcs $all \xrightarrow{a} all$ for every $a \in Act$. Finally, we complete the process $F(\mathcal{M})$ in the following way: for every node n, except for the one which corresponds to the final state $halt$ of $\mathcal{M}$, and every $a \in Act$, if there is no arc $n \xrightarrow{a} n'$ for any n', then add an arc $n \xrightarrow{a} all$. The initial state of $F(\mathcal{M})$ corresponds to the initial state of $\mathcal{M}$.

The state of $P(\mathcal{M})$ corresponds to the contents of the counters of $\mathcal{M}$ and the state of $F(\mathcal{M})$ corresponds to the state of the finite control of $\mathcal{M}$. A round in the simulation game corresponds to a computational step of $\mathcal{M}$.

The only problem is that $P(\mathcal{M})$ may do steps that do not correspond to steps of the counter machine, e.g. $P(\mathcal{M})$ does a step dec_1 when the current state in $F(\mathcal{M})$ expects inc_1. In all these cases the construction of $F(\mathcal{M})$ ensures that $F(\mathcal{M})$ can (and must) respond by a step that ends in the state all. After such a step $F(\mathcal{M})$ can simulate anything. It is easy to see that $P(\mathcal{M}) \not\sqsubseteq_s F(\mathcal{M})$ iff $P(\mathcal{M})$ can force $F(\mathcal{M})$ to enter the state $halt$ via a sequence of moves which correspond to the correct simulation of $\mathcal{M}$. Thus, $P(\mathcal{M}) \sqsubseteq_s F(\mathcal{M})$ iff the machine $\mathcal{M}$ does not halt. □

Remark 1. Theorem 7 still holds under an additional condition that both the PA process and the finite-state one are normed. We can make the PA process normed by adding the following rules: $Z_1 \xrightarrow{x_1} \epsilon$, $C_1 \xrightarrow{x_1} \epsilon$, $Z_2 \xrightarrow{x_2} \epsilon$, $C_2 \xrightarrow{x_2} \epsilon$. Observe that the resulting process is still deterministic. To make sure that $F(\mathcal{M})$ can simulate the actions x_1, x_2, we add the rules $n \xrightarrow{x_1} all$ and $n \xrightarrow{x_2} all$ for every state n of $F(\mathcal{M})$ (this also includes the rules $all \xrightarrow{x_1} all$ and $all \xrightarrow{x_2} all$). The process $F(\mathcal{M})$ is made normed by introducing a new state $terminated$ where no action is enabled, and a rule $all \xrightarrow{x} terminated$. It is easy to see that these new systems $P'(\mathcal{M})$ and $F'(\mathcal{M})$ are deterministic and normed, and still satisfy the property that $P'(\mathcal{M}) \sqsubseteq_s F'(\mathcal{M})$ iff the machine $\mathcal{M}$ does not halt.

The halting problem is undecidable even for two-counter machines with counters initialized to zero. The construction of $P(\mathcal{M})$ is then independent of $\mathcal{M}$. Furthermore, there exists a universal Minsky machine $\mathcal{M}'$; the halting problem for $\mathcal{M}'$ (with given input values) is undecidable, and the construction of $F(\mathcal{M}')$ is independent of those input values. Hence, we can conclude the following:

Theorem 6. *There is a normed deterministic PA process $\overline{P}$ and a normed deterministic finite-state process $\overline{F}$ such that*

- *the problem whether $\overline{P} \sqsubseteq_s F$ for a given (normed and deterministic) finite-state process F is undecidable,*

– *the problem whether $P \sqsubseteq_s \overline{F}$ for a given (normed and deterministic) PA process P is undecidable.*

Theorem 7. *Let P be a deterministic PA process and F a deterministic finite-state process. It is undecidable whether $F \sqsubseteq_s P$.*

Proof. Let $\mathcal{M}$ be an arbitrary two-counter machine with counters initialized to m_1, m_2. We construct a deterministic PA process $P(\mathcal{M})$ and a deterministic finite-state system $F(\mathcal{M})$ s.t. $F(\mathcal{M}) \sqsubseteq_s P(\mathcal{M})$ iff the machine $\mathcal{M}$ does not halt.

Let $Act := \{zero_1, inc_1, dec_1, zero_2, inc_2, dec_2, \tau\}$. For the construction of $P(\mathcal{M})$ we start with the same PA process as in Theorem 5 and extend it by the following rules, which handle all the behaviors that are 'illegal' in a given state of $P(\mathcal{M})$ w.r.t. the counter values it represents.

$$Z_1 \xrightarrow{dec_1} A_1 \qquad C_1 \xrightarrow{zero_1} A_1 \qquad A_1 \xrightarrow{a} A_1 \text{ for every } a \in \{zero_1, inc_1, dec_1, \tau\}$$
$$Z_2 \xrightarrow{dec_2} A_2 \qquad C_2 \xrightarrow{zero_2} A_2 \qquad A_2 \xrightarrow{a} A_2 \text{ for every } a \in \{zero_2, inc_2, dec_2, \tau\}$$

The intuition is that an illegal step that concerns the counter i (with $i \in \{1, 2\}$) always introduces the symbol A_i, and from then on everything can be simulated. The initial state is $(C_1^{m_1}.Z_1) \parallel (C_2^{m_2}.Z_2)$. Note that $P(\mathcal{M})$ is deterministic; a term that contains both A_1 and A_2 can do the action τ in two different ways, but the result is always the same.

The system $F(\mathcal{M})$ corresponds to the finite control of $\mathcal{M}$. For every instruction of the form $n:\ c_i := c_i + 1;\ \texttt{goto}\ n'$ we have an arc $n \xrightarrow{inc_i} n'$. For every instruction of the form $n:\ \texttt{if}\ c_i = 0\ \texttt{then goto}\ n'\ \texttt{else}\ c_i := c_i - 1;\ \texttt{goto}\ n''\ \texttt{fi}$ we have arcs $n \xrightarrow{zero_i} n'$ and $n \xrightarrow{dec_i} n''$. For the unique final state $halt$ of the finite control of $\mathcal{M}$ we add the rule $halt \xrightarrow{\tau} halt$. Note that a reachable state of $P(\mathcal{M})$ cannot do τ, unless it contains A_1 or A_2. Every step in the simulation game corresponds to a computational step of $\mathcal{M}$. It follows that $F(\mathcal{M}) \not\sqsubseteq_s P(\mathcal{M})$ iff $F(\mathcal{M})$ can reach the state $halt$ via a sequence of legal steps that correspond to steps of the counter machine (and do not introduce the symbol A_1 or A_2 in $P(\mathcal{M})$). Thus, $F(\mathcal{M}) \sqsubseteq_s P(\mathcal{M})$ iff the machine $\mathcal{M}$ does not halt. □

Remark 2. Theorem 7 still holds under an additional condition that both the PA process and the finite-state one are normed. The system $F(\mathcal{M})$ is made normed as follows: We introduce a new state $terminated$ where no action is enabled, and rules $n \xrightarrow{x} terminated$ for every other state n of $F(\mathcal{M})$. To assure that $P(\mathcal{M})$ can always simulate the action x, we add the rules $Z_1 \xrightarrow{x} \epsilon$, $C_1 \xrightarrow{x} \epsilon$, $A_1 \xrightarrow{x} \epsilon$. To make $P(\mathcal{M})$ normed, it now suffices to add the following: $Z_2 \xrightarrow{y} \epsilon$, $C_2 \xrightarrow{y} \epsilon$, $A_2 \xrightarrow{y} \epsilon$. It is easy to see that these new processes $P'(\mathcal{M})$ and $F'(\mathcal{M})$ are deterministic and normed, and still satisfy the property that $F'(\mathcal{M}) \sqsubseteq_s P'(\mathcal{M})$ iff the machine $\mathcal{M}$ does not halt.

A proof of the following theorem is the same as of Theorem 6:

Theorem 8. *There is a normed deterministic PA process $\overline{P}$ and a normed deterministic finite-state process $\overline{F}$ such that*

– *the problem whether $F \sqsubseteq_s \overline{P}$ for a given (normed and deterministic) finite-state process F is undecidable,*

– *the problem whether $\overline{F} \sqsubseteq_s P$ for a given (normed and deterministic) PA process P is undecidable.*

We have seen that simulation preorder is undecidable between deterministic PA processes and deterministic finite-state ones in both directions. However, simulation *equivalence* (as well as any other equivalence of the linear time/branching time spectrum of [19]) is *decidable* for such a pair of processes, because it coincides with bisimilarity which is known to be decidable [11]. Hence, it is interesting that simulation equivalence becomes *undecidable* as soon as we consider PA processes with just one nondeterministic state; this is proved in the following theorem:

Theorem 9. *There is a fixed normed deterministic finite-state process F s.t. the problem whether $P =_s F$ for a given normed PA process P is undecidable.*

Proof. We reduce the second undecidable problem of Theorem 6 to the problem if $P =_s F$. Let P' be a normed deterministic PA process, $\overline{F}$ be the fixed deterministic normed finite-state system derived from the finite control of the universal counter machine as in Theorem 6. We construct a normed PA process P and a fixed deterministic normed finite-state process F such that $P' \sqsubseteq_s \overline{F}$ iff $P =_s F$. It suffices to define F by $F \stackrel{a}{\rightarrow} \overline{F}$, and P by $P \stackrel{a}{\rightarrow} P'$, $P \stackrel{a}{\rightarrow} \overline{F}$. It follows immediately that $P =_s F$ iff $P' \sqsubseteq_s \overline{F}$. Note that P is not deterministic; however, it contains only one state (the initial state) where an action can be done in two different ways. □

On the other hand, simulation equivalence remains decidable between deterministic PA and *arbitrary* (possibly nondeterministic) finite-state systems. This is a consequence of a more general result—see the next section.

Remark 3. All undecidability results which have been proved in this section immediately extend to trace preorder and trace equivalence, because trace preorder and trace equivalence coincide with simulation preorder and simulation equivalence in the class of deterministic processes, respectively.

Now we prove that regularity w.r.t. simulation and trace equivalence is undecidable for normed PA processes with at least one nondeterministic state. It is interesting that regularity of normed deterministic PA processes w.r.t. any equivalence of the linear time/branching time spectrum of [19] is easily decidable in polynomial time, as it coincides with regularity w.r.t. bisimilarity which is known to have this property [13].

Theorem 10. *Regularity w.r.t. simulation equivalence and trace equivalence is undecidable for normed PA processes.*

5 The Relationship between Simulation and Bisimulation

In this section we concentrate on the relationship between simulation and bisimulation equivalence. It is a general trend that decidability results for bisimulation equivalence are positive, while the 'same' problems for simulation equivalence are undecidable. Major examples of that phenomenon come from the areas of equivalence-checking and regularity-testing (cf. the decidability issues for BPP and BPA processes which are summarized in the last section).

Theorem 11. *For every finitely branching transition systems T_1, T_2 there are finitely branching transition systems $\mathcal{B}(T_1), \mathcal{B}(T_2)$ such that $T_1 =_s \mathcal{B}(T_1)$, $T_2 =_s \mathcal{B}(T_2)$, and $\mathcal{B}(T_1), \mathcal{B}(T_2)$ are simulation equivalent iff they are bisimilar, i.e. $\mathcal{B}(T_1) =_s \mathcal{B}(T_2) \Leftrightarrow \mathcal{B}(T_1) \sim \mathcal{B}(T_2)$.*

Theorem 11 can be used as follows: if we are to decide simulation equivalence between T_1, T_2, we can try to construct $\mathcal{B}(T_1), \mathcal{B}(T_2)$ and decide bisimilarity between them. Similarly, regularity of T w.r.t. $=_s$ can be tested by constructing $\mathcal{B}(T)$ and checking its regularity w.r.t. $\sim$ (the construction of $\mathcal{B}(T)$ is not effective in general, of course).

As simulation preorder between finite-state processes is decidable, the system $\mathcal{B}(T)$ can be effectively constructed for any finite-state system T. Moreover, if T is deterministic then $\mathcal{B}(T) = T$. Thus, as a consequence of Theorem 11 we obtain:

Theorem 12. *Simulation equivalence is decidable between deterministic PA processes and (arbitrary) finite-state ones.*

Theorem 11 can also be applied in a nontrivial way. In a full version of our paper [15] we provide a little 'case-study'. We design a rich subclass of BPP processes where $\mathcal{B}(T)$ is effectively constructible; consequently, simulation equivalence as well as regularity w.r.t. simulation equivalence are decidable in this subclass.

6 Summary and Conclusions

The known decidability results in the area of equivalence/preorder checking between infinite-state processes and finite-state ones are summarized in the table below. The results which have been obtained in this paper are in boldface. In the case of trace preorder/equivalence/regularity we distinguish between deterministic infinite-state processes (left column) and general ones (right column); finite-state systems can be considered as deterministic here, because the subset construction [8] preserves trace equivalence.

	BPA		BPP		PA		PDA		PN	
$\sim$ FS	yes [6]		yes [5]		yes [11]		yes [18]		yes [12]	
reg. $\sim$	yes [3]		yes [10]		?		?		yes [10]	
$\sqsubseteq_s$ FS	**YES**		yes [12]		**NO**		**YES**		yes [12]	
FS $\sqsubseteq_s$	**YES**		yes [12]		**NO**		**YES**		yes [12]	
$=_s$ FS	**YES**		yes [12]		**NO**		**YES**		yes [12]	
reg. $=_s$	?		?		**NO**		?		no [12]	
$\sqsubseteq_t$ FS	yes	yes	yes [12]	yes [12]	**NO**	**NO**	yes	yes	yes [12]	yes [12]
FS $\sqsubseteq_t$	yes	no	yes [12]	yes [12]	**NO**	no	yes	no	yes [12]	yes [12]
$=_t$ FS	yes	no	yes [12]	yes [12]	yes [11]	no	yes	no	yes [12]	yes [12]
reg. $=_t$	yes	no	yes [10]	?	?	no	yes	no	yes [10]	no [12]

The results for trace preorder/equivalence might be also interesting from a point of view of automata theory (trace preorder and equivalence are closely related to language inclusion and equivalence, respectively). All 'trace' results for BPA and PDA are immediate consequences of the 'classical' ones for language equivalence (see [8]). It is interesting to compare those decidability issues with the ones for PA, especially in the deterministic subcase. Trace preorder with finite-state systems tends to be decidable for deterministic

processes; the class PA is the only exception. At the same time, trace *equivalence* with finite-state systems is *decidable* for deterministic PA. The PA processes we used in our undecidability proofs are parallel compositions of two deterministic and normed BPA processes (which can be seen as deterministic CF grammars). The parallel composition corresponds to the *shuffle* operator on languages [8]. Thus, our results bring some new insight into the power of shuffle on (deterministic) CF languages.

Interesting open questions are left in the area of regularity-testing. We can conclude that all the '?' problems are at least semidecidable, as it is possible to enumerate all finite-state systems and decide equivalence with them.

References

1. P.A. Abdulla and K. Čerāns. Simulation is decidable for one-counter nets. In *Proceedings of CONCUR'98*, volume 1466 of *LNCS*, pages 253–268. Springer-Verlag, 1998.
2. J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.
3. O. Burkart, D. Caucal, and B. Steffen. Bisimulation collapse and the process taxonomy. In *Proceedings of CONCUR'96*, volume 1119 of *LNCS*, pages 247–262. Springer-Verlag, 1996.
4. D. Caucal. On the regular structure of prefix rewriting. *Theoretical Computer Science*, 106:61–86, 1992.
5. S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation is decidable for all basic parallel processes. In *Proceedings of CONCUR'93*, volume 715 of *LNCS*, pages 143–157. Springer-Verlag, 1993.
6. S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, 121:143–148, 1995.
7. J.F. Groote and H. Hüttel. Undecidable equivalences for basic process algebra. *Information and Computation*, 115(2):353–371, 1994.
8. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
9. H. Hüttel. Undecidable equivalences for basic parallel processes. In *Proceedings of TACS'94*, volume 789 of *LNCS*, pages 454–464. Springer-Verlag, 1994.
10. P. Jančar and J. Esparza. Deciding finiteness of Petri nets up to bisimilarity. In *Proceedings of ICALP'96*, volume 1099 of *LNCS*, pages 478–489. Springer-Verlag, 1996.
11. P. Jančar, A. Kučera, and R. Mayr. Deciding bisimulation-like equivalences with finite-state processes. In *Proceedings of ICALP'98*, volume 1443 of *LNCS*, pages 200–211. Springer-Verlag, 1998.
12. P. Jančar and F. Moller. Checking regular properties of Petri nets. In *Proceedings of CONCUR'95*, volume 962 of *LNCS*, pages 348–362. Springer-Verlag, 1995.
13. A. Kučera. Regularity is decidable for normed PA processes in polynomial time. In *Proceedings of FST&TCS'96*, volume 1180 of *LNCS*, pages 111–122. Springer-Verlag, 1996.
14. A. Kučera and R. Mayr. Weak bisimilarity with infinite-state systems can be decided in polynomial time. Technical report TUM-I9830, Institut für Informatik, TU-München, 1998.
15. A. Kučera and R. Mayr. Simulation preorder on simple process algebras. Technical report TUM-I9902, Institut für Informatik, TU-München, 1999.
16. R. Mayr. Process rewrite systems. *Information and Computation*. To appear.
17. M.L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.
18. D.E. Muller and P.E. Schupp. The theory of ends, pushdown automata, and second order logic. *Theoretical Computer Science*, 37(1):51–75, 1985.
19. R.J. van Glabbeek. The linear time—branching time spectrum. In *Proceedings of CONCUR'90*, volume 458 of *LNCS*, pages 278–297. Springer-Verlag, 1990.