FI MU

Faculty of Informatics
Masaryk University

Bisimilarity of Processes with Finite-state Systems

by

Petr Jančar
Antonín Kučera

FI MU Report Series FIMU-RS-97-02

May 1997

Bisimilarity of Processes with Finite-state Systems[*]

Petr Jančar[†]
e-mail: jancar@osu.cz

Antonín Kučera[‡]
e-mail: tony@fi.muni.cz

Abstract

We describe a general method for deciding bisimilarity for pairs of processes where one process has finitely many states. We apply this method to pushdown processes and to PA processes. We also demonstrate that the mentioned problem is undecidable for 'state-extended' PA processes.

1 Introduction

The aim of this paper is to highlight an approach for deciding bisimulation equivalence between (some) infinite-state systems and finite-state ones. Previous results like [JM95], [AK95] and [JE96] in fact employed special instances of the general method described in this paper. Furthermore, we present two (new) applications to the classes of pushdown processes and PA processes. As an immediate consequence we obtain semi-decidability of regularity for these process classes. On the other hand, if we extend PA processes with a finite-state control unit, we obtain a calculus with full Turing power and the mentioned problems become undecidable.

[*]Supported by GA ČR, grant number 201/97/0456

[†]Dept. of Computer Science, Univ. of Ostrava and Techn. Univ. of Ostrava, Czech Republic

[‡]Faculty of Informatics, Masaryk University, Botanická 68a, 60200 Brno, Czech Republic

The first result indicating that decidability issues for bisimilarity are rather different from the ones for language equivalence is due to Baeten, Bergstra, and Klop. They proved in [BBK87, BBK93] that bisimilarity is decidable for context-free grammars in GNF (this class of processes is also known under the name 'normed BPA'). Much simpler proofs of this were later given in [Cau88], [HS91] and [Gro91]. In [HS91] Hüttel and Stirling used a tableau decision method and gave also sound and complete equational theory.

If we replace the binary sequential operator with the parallel operator, we obtain BPP processes. They can thus be seen as simple parallel programs. Christensen, Hirshfeld and Moller proved in [CHM93] that bisimilarity is decidable for BPP processes.

Another positive result [Sti96] is due to Stirling—it says that bisimilarity is decidable for normed PDA processes.

Jančar demonstrated in [Jan95] that bisimilarity is undecidable for labelled Petri nets. However, if one of those nets is bounded (i.e., finite-state), bisimilarity becomes decidable (see [JM95]).

Abdulla and Kindahl proved in [AK95] that bisimilarity is decidable between lossy channel systems and finite-state processes.

In this paper we show that bisimilarity is decidable for any pair of processes such that one process of this pair is a (general) PDA or PA process and the other process has finitely many states. Moreover, we also show that bisimilarity cannot be checked effectively between state-extended PA processes and finite-state ones.

Another interesting property of processes is *regularity*. A process is regular if it is bisimilar to some finite-state one. Jančar and Esparza proved in [JE96] that regularity is decidable for labelled Petri nets. Consequently, it is also decidable for BPP processes. Burkart, Caucal and Steffen demonstrated in [BCS96] that regularity is decidable for BPA processes. Another class of normed PA processes has been studied by Kučera in [Kuč96]—regularity is decidable even in polynomial time. A recent result [Jan97] due to Jančar says that regularity is decidable for one-counter processes.

Our results on decidability of bisimilarity between PDA (or PA) processes and finite-state processes immediately imply semi-decidability of regularity for PDA and PA processes. On the other hand, regularity of state-extended PA processes is shown to be undecidable.

2 Definitions

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

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

As usual, we write $s \xrightarrow{a} t$ instead of $(s, a, t) \in \rightarrow$ and we extend this notation to elements of Act^* in an obvious way (we sometimes write $s \rightarrow^* t$ instead of $s \xrightarrow{w} t$ if $w \in Act^*$ is irrelevant). A state t is *reachable* from a state s if $s \rightarrow^* t$.

A transition system $\mathcal{T} = (S, Act, \rightarrow)$ is *finitely-branching* if each state has finitely many immediate successors. $\mathcal{T}$ is *image-finite* is the set $\{t \mid s \xrightarrow{a} t\}$ is finite for each $s \in S$ and $a \in Act$.

Definition 2 (bisimilarity). *Let $\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1)$, $\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2)$ be transition systems. A binary relation $\mathcal{R} \subseteq S_1 \times S_2$ is a* bisimulation *if whenever $(s, t) \in \mathcal{R}$, then for each $a \in Act_1 \cup Act_2$:*

- *if $s \xrightarrow{a}_1 s'$, then $t \xrightarrow{a}_2 t'$ for some t' such that $(s', t') \in \mathcal{R}$.*
- *if $t \xrightarrow{a}_2 t'$, then $s \xrightarrow{a}_1 s'$ for some s' such that $(s', t') \in \mathcal{R}$.*

States $s \in S_1$ and $t \in S_2$ are bisimulation equivalent *(or* bisimilar*), written $s \sim t$, if there is a bisimulation relating them.*

In case of finitely-branching (and even image-finite) transition systems bisimilarity is characterizable using the following sequence of approximations (the symbol N_0 denotes the set of nonnegative integers):

Definition 3. *Let $\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1)$, $\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2)$ be transition systems. The family $\{\sim_i \mid i \in N_0\}$ is defined inductively as follows:*

- *$s \sim_0 t$ for all $s \in S_1$ and $t \in S_2$.*
- *$s \sim_{i+1} t$ iff for each $a \in Act_1 \cup Act_2$:*

- *if $s \xrightarrow{a}_1 s'$ then $t \xrightarrow{a}_2 t'$ for some t' such that $s' \sim_i t'$.*
- *if $t \xrightarrow{a}_2 t'$ then $s \xrightarrow{a}_1 s'$ for some s' such that $s' \sim_i t'$.*

It is easy to see that each $\sim_i$ is an equivalence relation; moreover, it is easily decidable in case of finitely-branching transition systems.[1] The following proposition is standard.

Proposition 1. *Let $\mathcal{T}_1, \mathcal{T}_2$ be image-finite transition systems and let s and t be states of T_1 and T_2, respectively. Then $s \sim t$ iff $s \sim_i t$ for each $i \in N_0$.*

2.1 PA processes

Let $Act = \{a, b, c, \ldots\}$ be a countably infinite set of *atomic actions*. Let $Var = \{X, Y, Z, \ldots\}$ be a countably infinite set of *variables* such that $Var \cap Act = \emptyset$. The class of PA expressions is defined by the following abstract syntax equation:

$$E_{PA} \ ::= \ \epsilon \ \mid \ a \ \mid \ aE_{PA} \ \mid \ E_{PA} \| E_{PA} \ \mid \ E_{PA} \lfloor\!\lfloor E_{PA} \ \mid \ E_{PA}.E_{PA} \ \mid \ E_{PA} + E_{PA}$$

Here a ranges over Act and X ranges over Var. In the rest of this paper we do not distinguish between expressions related by *structural congruence* which is the smallest congruence relation over PA expressions such that the following laws hold:

- associativity for '.', '$\|$' and '$+$'
- 'ϵ' as a unit for '.', '$\|$', '$\lfloor\!\lfloor$' and '$+$'
- commutativity for '$\|$' and '$+$'
- $a\epsilon = a$

As usual, we restrict our attention to *guarded* expressions; a PA expression E is guarded if there is a PA expression E' such that E and E' are structurally congruent and every variable occurrence in E' is within the scope of an atomic action.

[1]We assume that transition systems are defined in a "reasonable" way, i.e., the sets of states and labels are recursive, and the set of immediate successors of each state is effectively constructible.

$$\frac{}{aE \xrightarrow{a} E} \qquad \frac{E \xrightarrow{a} E'}{E.F \xrightarrow{a} E'.F} \qquad \frac{E \xrightarrow{a} E'}{E+F \xrightarrow{a} E'}$$

$$\frac{F \xrightarrow{a} F'}{E+F \xrightarrow{a} F'} \qquad \frac{E \xrightarrow{a} E'}{E\|F \xrightarrow{a} E'\|F} \qquad \frac{F \xrightarrow{a} F'}{E\|F \xrightarrow{a} E\|F'}$$

$$\frac{E \xrightarrow{a} E'}{E \mathbin{\lfloor\!\lfloor} F \xrightarrow{a} E'\|F} \qquad \frac{E \xrightarrow{a} E'}{X \xrightarrow{a} E'} \; (X \overset{def}{=} E \in \Delta)$$

Figure 1: SOS rules

A *PA system* is defined by a finite family Δ of recursive process equations

$$\Delta = \{X_i \overset{def}{=} E_i \mid 1 \leq i \leq n\}$$

where X_i are distinct elements of *Var* and E_i are guarded PA expressions, containing variables from $\{X_1, \ldots, X_n\}$. The set of variables which appear in Δ is denoted by $Var(\Delta)$.

Each PA system Δ determines a transition system whose states are PA expressions with variables from $Var(\Delta)$ (called *PA processes*), *Act* is the set of labels, and the transition relation is determined by the SOS rules of Figure 1.

We can even suppose each PA system to be in normal form (which is called *Greibach normal form* by analogy with CF grammars).

Definition 4 (GNF for PA systems). *A PA system Δ is in* Greibach normal form (GNF) *if each defining equation from Δ is of the form*

$$X_i \overset{def}{=} \sum_j a_{ij}\alpha_{ij}$$

where $a_{ij} \in Act$ and α_{ij} is a PA expression over the signature $\{\|, \lfloor\!\lfloor, .\}$, including ϵ (the set of all such expressions is denoted $VPA(\Delta)$). The set $\bigcup_{i,j}\{(X_i, a_{ij}, \alpha_{ij})\}$ is denoted by BT_Δ (Basic Transitions of Δ).

Given a PA system Δ and a PA process E, it is possible to construct a PA system Δ' in GNF and a PA process $E' \in VPA(\Delta')$ such that $E \sim E'$ (see [BEH95]). Hence the assumptions that Δ is in GNF and PA processes are elements of $VPA(\Delta)$ can be used w.l.o.g. Each transition $\alpha \xrightarrow{a} \beta$ is then due to a unique element of $BT_{\Delta'}$ which is denoted $Step(\alpha \xrightarrow{a} \beta)$.

If we omit the '$\|$' and '$\lfloor\!\lfloor$' operators from the definition of PA systems, we get an important subclass of *BPA systems*. Greibach normal form for BPA allows to assume that BPA processes are sequences of variables; in the next subsection we extend PA systems with finite-state control unit. If we apply the same construction to BPA systems, we obtain exactly the class of pushdown (PDA) systems.

2.2 State-extended PA processes

A *state-extended PA system* is a triple $\psi = (\Delta, Q, BT_{St})$ where Δ is a PA system in GNF, Q is a finite set of *states*, and $BT_{St} \subseteq BT_\Delta \times Q \times Q$ is a set of *state-extended basic transitions*.

The transition system generated by a state-extended PA process $\psi = (\Delta, Q, BT_{St})$ has $Q \times VPA(\Delta)$ as the set of states (its elements are called *state-extended PA processes*, or *StExt(PA)* processes for short), *Act* is the set of labels, and the transition relation is determined by the rule

$$(p, \alpha) \xrightarrow{a} (q, \beta) \text{ iff } \alpha \xrightarrow{a} \beta \text{ and } (Step(\alpha \xrightarrow{a} \beta), p, q,) \in BT_{St}$$

As we already mentioned in the previous section, the class of pushdown (PDA) systems can be obtained by extending BPA with a finite-state control unit; PDA processes are thus *StExt(BPA)* processes in fact.

3 The general method

In this section we describe the promised general method for deciding bisimilarity between two processes where one process has finitely many states. For notation simplification, we adopt the following conventions:

- $\mathcal{F}$ denotes a finite-state transition system with k states.
- $\mathcal{G}$ denotes a (general) transition system.
- Labels of $\mathcal{F}$ and $\mathcal{G}$ are elements of a finite set *Act*.

Note that each PA or *StExt(PA)* system actually contains only finitely many actions, hence the set *Act* can be considered as finite.

Lemma 1. *Let f_1, f_2 be two states of $\mathcal{F}$. Then $f_1 \sim f_2$ iff $f_1 \sim_{k-1} f_2$.*

Proof:
'$\Rightarrow$': Obvious.
'$\Leftarrow$': As $\mathcal{F}$ has k states, $\sim_{k-1} = \sim_k$ (note that $\sim_i$ is a refinement of $\sim_{i-1}$ for each $i \in N$) and hence $\sim_{k-1} = \sim_k = \sim$. $\square$

Now we come to the crucial proposition.

Proposition 2. *Let g and f be states of $\mathcal{G}$ and $\mathcal{F}$, respectively. Then $g \sim f$ iff $g \sim_k f$ and for each g' such that $g \rightarrow^* g'$ there is a state f' of $\mathcal{F}$ with $g' \sim_k f'$.*

Proof:
'$\Rightarrow$': Obvious.
'$\Leftarrow$': We prove that the relation

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

is a bisimulation. Let $(g', f') \in \mathcal{R}$ and let $g' \xrightarrow{a} g''$ (the case when $f' \xrightarrow{a} f''$ is handled is the same way). By definition of $\sim_k$, there is 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 $\overline{f}$ of $\mathcal{F}$ such that $g'' \sim_k \overline{f}$. By transitivity of $\sim_{k-1}$ we have $\overline{f} \sim_{k-1} f''$, hence $\overline{f} \sim_k f''$ (due to Lemma 1). Now $g'' \sim_k \overline{f} \sim_k f''$ and thus $g'' \sim_k f''$ as required. Clearly $(g, f) \in \mathcal{R}$ and the proof is finished. $\square$

Proposition 2 enables the following *general strategy* for deciding whether or not $g \sim f$:

1. Decide whether $g \sim_k f$ (if not then $g \not\sim f$).

2. Check whether g can reach a state g' such that $g' \not\sim_k f'$ for *each* state f' of $\mathcal{F}$ (if there is such a g', then $g \not\sim f$; otherwise $g \sim f$).

As we deal with processes associated with finitely-branching transition systems, the first condition is easily decidable. We can thus concentrate on the latter one. The aim of the following definition is to characterize all 'k-step' behaviours.

Definition 5 (Tree-process). *For each $i \in N_0$ we define the set of* Trees *over Act with* depth *at most i as follows:*

- *The only Tree with depth* 0 *is a tree with singleton node and no arcs.*

- *A Tree with depth at most $i+1$ is any directed tree with root r whose arcs are labelled with elements of Act which fulfills the following conditions:*
  - *There is no arc ending in r.*
  - *If $r \xrightarrow{a} s$, then the subtree rooted by s is a Tree with depth at most i.*
  - *If $r \xrightarrow{a} s$ and $r \xrightarrow{a} s'$, then the subtrees rooted by s and s' are non-isomorphic.*

Each Tree can be seen as a transition system. Tree-processes *are associated with roots of Trees.*

It is clear that for any finite set Act and $i \in N_0$ there are only finitely many Trees (and hence also Tree-processes) over Act with depth at most i (up to isomorphism). More precisely, the total number of such Trees (denoted $NT(i)$) is given by

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

States of $\mathcal{G}$ which can be distinguished from any state of $\mathcal{F}$ within the first k steps can be characterized by the following set of Trees:

Definition 6 ($INC^{\mathcal{F}}$). *The set $INC^{\mathcal{F}}$ of Tree-processes which are incompatible with $\mathcal{F}$ is defined as follows:*

$$\begin{aligned} INC^{\mathcal{F}} \;=\; & \{T \mid T \textit{ is a Tree-process with depth at most } k \\ & \textit{ and } T \not\sim_k f \textit{ for each state } f \textit{ of } \mathcal{F}\} \end{aligned}$$

It is obvious that the general method can be applied to a class of processes $\mathcal{P}$ if the following ‘reachability’ problem is decidable:

The R-problem

Instance:	$[k, P, T]$ where $k \in N$, P is a process of $\mathcal{P}$ and T is a Tree with depth at most k
Question:	Is there a state P' such that $P \to^* P'$ and $P' \sim_k T$?

4 Applications

In this section we apply the previously described general method to PDA processes and PA processes. In both cases we just demonstrate decidability of the R-problem.

4.1 PDA processes

We prove that the R-problem for PDA processes can be reduced to the problem whether an extended pushdown automaton[2] accepts a nonempty language (this problem is known to be decidable—see e.g., [HU79] for general introduction to automata theory).

Theorem 1. *Bisimilarity is decidable between PDA processes and finite-state processes.*

Proof: Let $[k, p\alpha, T]$ be an instance of R-problem and let $\psi = (\Delta, Q, BT_{St})$ be the PDA system (i.e., *StExt(BPA)* system) associated with $p\alpha$. As the 'k-step' behaviour of each PDA process $q\gamma$ is completely determined by the first k symbols of γ, each process which is related with T by $\sim_k$ has a representative in the following finite set:

$$Rep = \{q\gamma \mid length(\gamma) \leq k \text{ and } q\gamma \sim_k T\}$$

The set *Rep* is effectively constructible. Now we want to check whether $p\alpha$ can reach a state $r\beta$ such that one of the following conditions holds:

- $length(\beta) \leq k$ and $r\beta \in Rep$
- $length(\beta) > k$ and Rep contains an element $r\eta$ where η is the prefix of β of length k.

To do this, we construct an extended pushdown automaton $\mathcal{A}$ which has $Q \cup \{start, final\}$ as the set of states, and $Var(\Delta) \cup \{Z_0\}$ as the stack alphabet (*start* is the initial state, *final* is the only final state, and Z_0 is the stack bottom). The transition function δ is determined as follows:

[2]An extended pushdown automaton is a nondeterministic pushdown automaton which can "see" a bounded prefix of its stack. It can be simulated by an effectively constructible pushdown automaton.

- $\delta(start, \epsilon, Z_0) = \{(p, \alpha Z_0)\}$
- if $qX \xrightarrow{a} r\gamma$ then $(r, \gamma) \in \delta(q, a, X)$
- $\delta(q, \epsilon, \gamma) = \{(\mathit{final}, \gamma)\}$ for each $q\gamma \in \mathit{Rep}$ such that $\mathit{length}(\gamma) = k$
- $\delta(q, \epsilon, \gamma Z_0) = \{(\mathit{final}, \gamma Z_0)\}$ for each $q\gamma \in \mathit{Rep}$ such that $\mathit{length}(\gamma) < k$

The automaton $\mathcal{A}$ enters a final state (i.e., accepts a word) iff the process $p\alpha$ can reach a state which is related with T by $\sim_k$. This reduction proves the theorem. $\square$

4.2 PA processes

Before we prove an analogous theorem for PA processes, we need to introduce further notation. Let $T_1 = (S_1, \mathit{Act}, \rightarrow_1, r_1)$, $T_2 = (S_2, \mathit{Act}, \rightarrow_2, r_2)$ be two Trees with depth at most k (remember that Trees can be seen as rooted transition systems—the first three elements of the tuple are interpreted in the same way as in case of transition systems; the last one denotes the root). Furthermore, we assume that $S_1 \cap S_2 = \emptyset$. Processes $T_1 \| T_2$, $T_1.T_2$ and $T_1 + T_2$ are defined as follows:

- $T_1 \| T_2$ is associated with the node (r_1, r_2) in the transition system $(S_1 \times S_2, \mathit{Act}, \rightarrow)$ where $\rightarrow$ is the least transition relation satisfying the following rules:
  - $s \xrightarrow{a}_1 s' \;\Rightarrow\; (s, t) \xrightarrow{a} (s', t)$ for each $t \in S_2$.
  - $t \xrightarrow{a}_2 t' \;\Rightarrow\; (s, t) \xrightarrow{a} (s, t')$ for each $s \in S_1$.
- $T_1.T_2$ is associated with the node r_1 in the transition system $(S_1 \cup S_2, \mathit{Act}, \rightarrow)$ where $\rightarrow$ is the least transition relation satisfying the following rules:
  - $s \xrightarrow{a}_1 s' \;\wedge\; s'$ is not a leaf $\;\Rightarrow\; s \xrightarrow{a} s'$
  - $s \xrightarrow{a}_1 s' \;\wedge\; s'$ is a leaf $\;\Rightarrow\; s \xrightarrow{a} r_2$
  - $t \xrightarrow{a}_2 t' \;\Rightarrow\; t \xrightarrow{a} t'$
- $T_1 + T_2$ is associated with the node r_1 in the transition system $(S_1 \cup S_2 - \{r_2\}, \mathit{Act}, \rightarrow)$ where $\rightarrow$ is the least transition relation satisfying the following rules:

- $s \xrightarrow{a}_1 s' \;\Rightarrow\; s \xrightarrow{a} s'$
- $r_2 \xrightarrow{a}_2 t \;\Rightarrow\; r_1 \xrightarrow{a} t$
- $t \xrightarrow{a}_2 t' \wedge t \neq r_2 \;\Rightarrow\; t \xrightarrow{a} t'$

In the proof of the following theorem we employ a general technique known as *tableau system*, a goal-directed proof method. It is specified by a finite system of *inference rules* of the form

$$\frac{goal}{subgoal_1 \cdots subgoal_n} \; (side\ conditions)$$

A tableau for a goal G is a maximal proof tree whose root is labelled G and where immediate successors of each node are determined by application of one of the rules (side conditions optionally specify some restrictions). These rules are applied only to nodes which are not *terminal.* Terminal nodes are either *successful* or *unsuccessful*; a *successful tableau* is a finite tableau where all leaves are successful terminals. Other tableaux are *unsuccessful.*

If we want to demonstrate decidability of some problem $\mathcal{P}$ by means of a tableau system, it suffices to prove that the tableau system fulfills the following conditions:

1. Each tableau is finite and there are only finitely many tableaux with a given root (finiteness).

2. If there is a successful tableau rooted by an instance P of the problem $\mathcal{P}$, then P is a positive instance (soundness).

3. For each positive instance P of the problem $\mathcal{P}$ there is a successful tableau rooted by P (completeness).

If all the mentioned conditions are true, we can decide $\mathcal{P}$ by an exhaustive search for a successful tableau.

Theorem 2. *Bisimilarity is decidable between PA processes and finite-state processes.*

Proof: Decidability of R-problem will be demonstrated by a tableau system specified by the rules of Figure 2. Let $[k, E, T]$ be an instance of R-problem and let Δ be the PA system associated with E (we do not require

$$\frac{aE;\ T}{E;\ T}\quad (*) \qquad\qquad \frac{\overline{aE};\ T}{E;\ T}\quad (*)$$

$$\frac{E_1 + E_2;\ T}{\overline{E_1};\ T} \qquad\qquad \frac{E_1 + E_2;\ T}{\overline{E_2};\ T}$$

$$\frac{\overline{E_1 + E_2};\ T}{\overline{\overline{E_1}};\ T} \qquad\qquad \frac{\overline{E_1 + E_2};\ T}{\overline{\overline{E_2}};\ T}$$

$$\frac{E_1.E_2;\ T}{E_1;\ T_1}\,(T \sim_k T_1.T_2\ \wedge\ E_2 \sim_k T_2) \qquad \frac{E_1.E_2;\ T}{E_2;\ T}\,(E_1 \rightarrow^* \epsilon)$$

$$\frac{\overline{E_1.E_2};\ T}{\overline{E_1};\ T_1}\,(T \sim_k T_1.T_2\ \wedge\ E_2 \sim_k T_2) \qquad \frac{\overline{E_1.E_2};\ T}{E_2;\ T}\,(E_1 \rightarrow^* \epsilon)$$

$$\frac{E_1 \| E_2;\ T}{E_1;\ T_1 \qquad E_2;\ T_2}\,(T \sim_k T_1 \| T_2) \qquad \frac{\overline{E_1 \| E_2};\ T}{\overline{\overline{E_1}};\ T_1 \qquad E_2;\ T_2}\,(T \sim_k T_1 \| T_2)$$

$$\frac{\overline{E_1 \| E_2};\ T}{E_1;\ T_1 \qquad \overline{\overline{E_2}};\ T_2}\,(T \sim_k T_1 \| T_2) \qquad \frac{E_1 \mathbin{\underline{\|}} E_2;\ T}{\overline{\overline{E_1}};\ T_1 \qquad E_2;\ T_2}\,(T \sim_k T_1 \| T_2)$$

$$\frac{\overline{E_1 \mathbin{\underline{\|}} E_2};\ T}{\overline{\overline{E_1}};\ T_1 \qquad E_2;\ T_2}\,(T \sim_k T_1 \| T_2) \qquad \frac{X;\ T}{E;\ T}\,(X \overset{def}{=} E \in \Delta)$$

$$\frac{\overline{X};\ T}{\overline{\overline{E}};\ T}\,(X \overset{def}{=} E \in \Delta)$$

Figure 2: Tableau rules for the proof of Theorem 2

Δ to be in GNF). We determine whether $E \rightarrow^* E'$ for some E' such that $E' \sim_k T$ by constructing a tableau rooted by $E; T$.

Nodes of each tableau are labelled by expressions of the form $E; T$ or $\overline{E}; T$, where E is a PA process and T is a Tree with depth at most k. Side conditions place some restrictions on Trees which can be used in subgoals.[3] Terminal nodes are defined as follows:

- A *successful terminal* is a node $E; T$ such that $E \sim_k T$ (note that nodes of the form $\overline{E}; T$ cannot be successful terminals).
- *Unsuccessful terminals* can be divided into two groups as follows:
    1. A node of the form $\epsilon; T$ or $\overline{\epsilon}; T$ where $\epsilon \not\sim_k T$.
    2. A node for which there is another node with the same label above (along the path from the root).

Intuition which stands behind the design of tableau rules is formally expressed by the following predicate *Pr* of nodes:

- $Pr(E; T) = true$ iff $E \rightarrow^* E'$ for some E' such that $E' \sim_k T$.
- $Pr(\overline{E}; T) = true$ iff $E \overset{w}{\rightarrow} E'$ for some E' and $w \in Act^*$ such that $E' \sim_k T$ and $length(w) \geq 1$.

To finish the proof, we need to show *finiteness*, *soundness* and *completeness* of the tableau method.

Nodes are labelled by pairs of the form $E; T$ or $\overline{E}; T$, where E is a subexpression of a PA expression contained either in the root or in some defining equation from Δ, and T is Tree with depth at most k. As there are only finitely many such subexpressions and the set of all Trees with depth at most k is also finite, there are only finitely many (potential) labels. This gives an obvious bound on the depth of each tableau (recall the definition of unsuccessful terminal). As each tableau is finitely branching, it must be finite (due to König's lemma). For the same reason there are only finitely many tableaux with a given root.

For soundness, it suffices to prove that the root of each successful tableau satisfies the predicate *Pr*. This is rather straightforward—terminal

[3]The side condition $E \rightarrow^* \epsilon$ means that E can reach the empty expression; this property is also known as *normedness* and it is easily decidable.

nodes clearly satisfy *Pr* and each rule of Figure 2 is *backward sound* in the sense that if all subgoals satisfy the predicate *Pr*, then the goal satisfies *Pr*.

Completeness is slightly more complicated. We need to show that if $[k, E, T]$ is a positive instance of R-problem, then there is a successful tableau with the root $E; T$. To do this, realize the following fact: if a node labelled $E'; T'$ satisfies *Pr*, then it is possible to apply an instance of one of the rules of Figure 2 in such a way that all newly-added subgoals satisfy *Pr*. This is easy to check. Each such instance is called a *good* instance. Naturally, there can be many good instances for one node—to build a successful tableau, we always choose an instance with minimal *cost*. A cost of a good instance is defined to be the sum of *distances* of all subgoals, where the distance of a subgoal $E''; T''$ is defined as

$$\min\{length(w) \mid E'' \xrightarrow{w} F \text{ where } F \sim_k T''\}$$

A tableau for $[k, E, T]$ which is built according to this strategy cannot contain an unsuccessful terminal of the type 1. Moreover, it cannot contain an unsuccessful terminal of the type 2; this follows from an observation that one of the (*) rules has to be applied at least once before the same label (say $E'; T'$) occurs again—and it contradicts minimality of cost of the good instance which was applied to the first (upper) occurrence of $E'; T'$. □

5 An undecidability result

In this section we prove that bisimilarity is undecidable between *StExt(PA)* processes and finite-state processes. We also demonstrate undecidability of the regularity problem for *StExt(PA)* processes. These results are simple consequences of the fact that an arbitrary Minsky machine can be simulated by an effectively constructible *StExt(PA)* process. In other words, *StExt(PA)* is a calculus with full Turing power.

5.1 The Minsky machine

The Minsky machine (denoted here by $\mathcal{M}$) is equipped with two counters C_1, C_2 which can store nonnegative integers. The behaviour of $\mathcal{M}$ is determined by a finite-state program, composed of $m \in N$ labelled statements

$$\begin{array}{lll} l_1 & : & s_1 \\ l_2 & : & s_2 \\ \vdots & & \\ l_{m-1} & : & s_{m-1} \\ l_m & : & \texttt{HALT} \end{array}$$

where for each i, $1 \leq i < m$ the statement s_i is of one of the following forms:

$$s_i = \begin{cases} C_j = C_j + 1; \texttt{goto}\ l_k \\ \texttt{if}\ C_j = 0\ \texttt{then}\ \ \texttt{goto}\ l_k\ \texttt{else}\ C_j = C_j - 1; \texttt{goto}\ l_n; \end{cases}$$

where $j \in \{1, 2\}$. The machine $\mathcal{M}$ starts its execution (with given input values on C_1, C_2) from the command l_1. $\mathcal{M}$ *halts* if it reaches the command ‘HALT’ in a finite number of steps, and *diverges* otherwise. Minsky has shown in [Min67] that an arbitrary Turing machine can be simulated by an effectively constructible Minsky machine. This implies that the halting problem of Minsky machine is generally undecidable.

5.2 The simulation

Let $\mathcal{M}$ be an arbitrary Minsky machine whose program has m statements and counters initialized to v_1 and v_2. We construct a *StExt(PA)* system $\psi = (\Delta, Q, BT_{St})$ as follows:

- Δ contains the following equations:

$$Z_1 \overset{def}{=} a(I_1.Z_1) + aZ_1$$
$$I_1 \overset{def}{=} a(I_1.I_1) + a$$
$$Z_2 \overset{def}{=} a(I_2.Z_2) + aZ_2$$
$$I_2 \overset{def}{=} a(I_2.I_2) + a$$

- $Q = \{q_1, \ldots, q_m\}$
- BT_{St} is determined by the following rules:

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

     $l_i: \ C_j = C_j + 1; \ \texttt{goto}\ l_k$

     then BT_{St} contains the elements $q_iZ_j \xrightarrow{a} q_k(I_j.Z_j)$ and $q_iI_j \xrightarrow{a} q_k(I_j.I_j)$.

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

   $$l_i : \texttt{ if } C_j = 0 \texttt{ then goto } l_k \texttt{ else } C_j = C_j - 1; \texttt{goto } l_n$$

   then BT_{St} contains the elements $q_i Z_j \xrightarrow{a} q_k Z_j$ and $q_i I_j \xrightarrow{a} q_n$.
3. Each element of BT_{St} can be derived using the rule 1 or 2.

The machine $\mathcal{M}$ is simulated by the *StExt(PA)* process

$$\varphi \equiv q_1((\underbrace{I_1.\cdots.I_1}_{v_1}.Z_1)\|(\underbrace{I_2.\cdots.I_2}_{v_2}.Z_2))$$

Intuitively, counters of $\mathcal{M}$ are simulated by two BPA processes which are combined in parallel on the 'stack' and the program of $\mathcal{M}$ is simulated by the finite-state control unit of ψ . Each step of $\mathcal{M}$ is mimicked by φ which emits the action a. Let Y be a process defined by $Y \overset{def}{=} aY$. If the machine $\mathcal{M}$ diverges then $\varphi \sim Y$. If the machine $\mathcal{M}$ halts then $\varphi \not\sim Y$, because φ emits the action a only finitely many times (note that $\mathcal{M}$ is deterministic). This reduction proves the following theorem:

Theorem 3. *Bisimilarity is undecidable between StExt(PA) processes and finite-state processes.*

Remark 1. *Bisimilarity is not the only behavioural equivalence which appeared in the literature; Rob van Glabbeek presented in [vG90] a hierarchy of equivalences, relating them w.r.t. their* coarseness. *The finest equivalence in this hierarchy is bisimilarity, and at the very bottom is* trace equivalence. *It is easy to see that the previous theorem can be generalized to* all *equivalences of van Glabbeek's hierarchy because if $\mathcal{M}$ does not halt, then φ and Y are even not* trace equivalent.

Now we can easily prove that *regularity* of *StExt(PA)* processes is also undecidable. The following definition recalls the notion.

Definition 7. *A process X is* regular *if there is a process X' with finitely many states such that $X \sim X'$.*

Theorem 4. *Regularity is undecidable for StExt(PA) processes.*

Proof: We use a similar reduction as in the previous theorem—given a Minsky machine $\mathcal{M}$, we construct the *StExt(PA)* system ψ. Now we modify the system ψ slightly—we add a new state q' which can be entered only from q_m (for any contents of the 'stack'). The state q' can manipulate the 'stack' in such a way that there are infinitely many states (up to bisimilarity) reachable from the process $q'\gamma$ for any $\gamma \in \mathit{VPA}(\Delta)$. The resulting system is denoted ψ'. If $\mathcal{M}$ does not halt, then the process φ is regular, because it is again bisimilar to $Y \overset{def}{=} aY$. If $\mathcal{M}$ halts, then φ is non-regular as it can reach a state of the form $q'\gamma$. □

6 Conclusions

We described a general method for deciding bisimilarity between (some) infinite-state processes and finite-state ones. Successful application of this method to the classes of PDA and PA processes immediately imply semi-decidability of regularity (by exhaustive search for bisimilar finite-state process). Decidability of regularity (i.e., semi-decidability of the negative subcase) is left open. Furthermore, we also demonstrated that if we extend PA processes with a finite-state control unit, we obtain a calculus with full Turing power and the mentioned problems become undecidable.

It is worth mentioning that a similar method can be designed for deciding *weak bisimilarity* (see e.g., [Mil89]) between some infinite-state processes and finite-state ones. The problem whether this method can be applied to PA and / or PDA processes is a part of ongoing research. The same problem was shown to be decidable for BPP processes in [May96], and undecidable for Petri nets and lossy channel systems in [JE96] and [AK95], respectively.

References

[AK95] P.A. Abdulla and M. Kindahl. Decidability of simulation and bisimulation between lossy channel systems and finite state systems. In *Proceedings of CONCUR'95* [Con95], pages 333–347.

[BBK87] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. In *Proceedings of PARLE'87*, volume 259 of *LNCS*, pages 93–114. Springer-Verlag, 1987.

[BBK93] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. *Journal of the Association for Computing Machinery*, 40:653–682, 1993.

[BCS96] O. Burkart, D. Caucal, and B. Steffen. Bisimulation collapse and the process taxonomy. In *Proceedings of CONCUR'96* [Con96], pages 247–262.

[BEH95] A. Bouajjani, R. Echahed, and P. Habermehl. Verifying infinite state processes with sequential and parallel composition. In *Proceedings of POPL'95*, pages 95–106. ACM Press, 1995.

[Cau88] D. Caucal. Graphes canoniques de graphes algebriques. Rapport de Recherche 872, INRIA, 1988.

[CHM93] 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.

[Con95] *Proceedings of CONCUR'95*, volume 962 of *LNCS*. Springer-Verlag, 1995.

[Con96] *Proceedings of CONCUR'96*, volume 1119 of *LNCS*. Springer-Verlag, 1996.

[FST96] *Proceedings of FST&TCS'96*, volume 1180 of *LNCS*. Springer-Verlag, 1996.

[Gro91] J.F. Groote. A short proof of the decidability of bisimulation for normed BPA processes. *Information Processing Letters*, 42:167–171, 1991.

[HS91] H. Hüttel and C. Stirling. Actions speak louder than words: Proving bisimilarity for context-free processes. In *Proceedings of LICS'91*, pages 376–386. IEEE Computer Society Press, 1991.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[Jan95] P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301, 1995.

[Jan97] P. Jančar. Bisimulation equivalence is decidable for one-counter processes. To appear in Proc. of ICALP'97. LNCS. Springer-Verlag, 1997.

[JE96] 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.

[JM95] P. Jančar and F. Moller. Checking regular properties of Petri nets. In *Proceedings of CONCUR'95* [Con95], pages 348–362.

[Kuč96] A. Kučera. Regularity is decidable for normed PA processes in polynomial time. In *Proceedings of FST&TCS'96* [FST96], pages 111–122.

[May96] R. Mayr. Weak bisimulation and model checking for basic parallel processes. In *Proceedings of FST&TCS'96* [FST96], pages 88–99.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[Min67] M.L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

[Sti96] C. Stirling. Decidability of bisimulation equivalence for normed pushdown processes. In *Proceedings of CONCUR'96* [Con96], pages 217–232.

[vG90] 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.