 Electronic Notes in Theoretical Computer Science 98 (2004) 5–19 

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Completeness Results for Undecidable Bisimilarity Problems

Jiˇr´ı Srba[1](#_bookmark0) ,[2](#_bookmark0)

BRICS*, Department of Computer Science University of Aalborg*

*Fredrik Bajersvej 7E, 9220 Aalborg, Denmark*

Abstract

1

We establish Σ1-completeness (in the analytical hierarchy) of weak bisimilarity checking for infinite- state processes generated by pushdown automata and parallel pushdown automata. The results

imply Σ1-completeness of weak bisimilarity for Petri nets and give a negative answer to the open problem stated by Janˇcar (CAAP’95): “does the problem of weak bisimilarity for Petri nets belong

1

1

to ∆1?”

*Keywords:* Weak bisimilarity, undecidability, process algebra.

# Introduction

Given two (infinite-state) processes, the *equivalence checking problem* is to decide whether the processes are equivalent with regard to some behavioral equivalence. This question has been intensively studied for various classes of infinite-state systems (see e.g. [[1](#_bookmark5),[8](#_bookmark12),[12](#_bookmark16)] for overviews). The notion of *bisimula-*

*tion equivalence* is of particular interest both for the theory and practice.

Strong (and weak) bisimilarity checking of Petri nets (PN) is known to be undecidable [[6](#_bookmark10)]. In the case of strong bisimilarity the problem is Π0-complete in the arithmetical hierarchy (see e.g. [[5](#_bookmark9)]) and in the weak case it is known to be highly undecidable [[5](#_bookmark9)] (i.e. it lies beyond the arithmetical hierarchy).

1

1 Email: [srba@brics.dk](mailto:srba@brics.dk)

2 This research was supported in part by the GACR, grant No. 201/03/1161.

1571-0661 © 2004 Elsevier B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

doi:10.1016/j.entcs.2003.10.003

On the other hand, strong bisimilarity checking of pushdown processes (PDA) remains decidable [[11](#_bookmark15)] while e.g. the language equivalence is not. The weak bisimilarity problem for PDA was recently proved to be undecidable [[14](#_bookmark17)] and we conjectured that the problem lies beyond the arithmetical hierarchy.

In this paper we confirm this conjecture and we strengthen the results of high undecidability of weak bisimilarity for PDA and PN not only to *ω*- hardness in the arithmetical hierarchy but also to Σ1-hardness (the first level of the analytical hierarchy). In the case of Petri nets our proof generalizes the result of Janˇcar [[5](#_bookmark9)] also in another way: the result is demonstrated for a proper subclass of PN called parallel pushdown processes (PPDA) or also multiset automata.

1

As for the upper bounds it is easy to observe that the weak bisimilar- ity problems are contained in Σ1 (see [[5](#_bookmark9)]). Hence Σ1-completeness of weak

1 1

bisimilarity for PDA and PPDA (and PN) is established.

An interesting observation is that PDA, PN and PPDA are not Turing powerful (e.g. reachability remains decidable [[9](#_bookmark13)]) but still the weak bisimilarity problems are surprisingly highly undecidable.

These Σ1-lower bounds contrast to other results in the theory. For exam- ple (weak) trace equivalence checking of PDA and PN remains Π0-complete (see [[5](#_bookmark9)]). On the other hand for the communication-free subclass of PN called basic parallel processes strong bisimilarity is PSPACE-complete [[7](#_bookmark11),[13](#_bookmark18)] and weak bisimilarity is very likely to be decidable, while other equivalences in-

1

1

cluding (strong and weak) trace equivalence are undecidable [[4](#_bookmark8)]. In fact (strong and weak) trace equivalence is Π0-complete [[5](#_bookmark9)]. Similar surprising results are valid also for stateless PDA (called basic process algebra) where strong bisimi- larity is decidable in 2-EXPTIME [[2](#_bookmark6)] and weak bisimilarity is conjectured to be also decidable, while (strong and weak) trace equivalence is undecidable (the formalism describes exactly the class of context-free languages). Again, the problem of (strong and weak) trace equivalence can be seen to be Π0-complete

1

1

by using a construction from [[5](#_bookmark9)].

# Basic Definitions

A *labelled transition system* is a triple (*S,* A*ct,* −→) where *S* is a set of *states* (or *processes*), A*ct* is a set of *labels* (or *actions*), and −→⊆ *S* × A*ct* × *S* is a *transition relation*, written *α* −*a*→ *β*, for (*α, a, β*) ∈−→.

Assume that the set of actions A*ct* contains a distinguished *silent action*

*τ* . The *weak transition relation* =⇒ is defined by =⇒*a*

def

−*τ*→)∗◦ −*a*→ ◦(−*τ*→)∗

if *a* ∈ A*ct* z {*τ* }

, and =

*a* def

⇒

= (

−*τ*→)∗ if *a* = *τ* .

= (

Let (*S,* A*ct,* −→) be a labelled transition system. A binary relation *R* ⊆

*S* × *S* is a *weak bisimulation* iff whenever (*α, β*) ∈ *R* then for each *a* ∈ A*ct*: if *α* −*a*→ *α*' then *β* =*a*⇒ *β*' for some *β*' such that (*α*'*, β*') ∈ *R*; and if *β* −*a*→ *β*'

*a*

then *α* =⇒

*α*' for some *α*' such that (*α*'*, β*') ∈ *R*. Processes *α* and *β* are *weakly*

*bisimilar* (*α* ≈ *β*) iff there is a weak bisimulation *R* such that (*α, β*) ∈ *R*.

Weak bisimilarity has an elegant characterization in terms of *bisimulation games*. A bisimulation game on a pair of processes *α*1 and *α*2 is a two-player game between an ‘attacker’ and a ‘defender’. The game is played in *rounds*. In each round the players change the *current states β*1 and *β*2 (initially *α*1 and *α*2) according to the following rule.

1. The attacker chooses an *i* ∈ {1*,* 2}, *a* ∈ A*ct* and *β*'

∈ *S* such that

*a* ' *i*

*βi* −→ *βi*.

1. The defender responds by choosing a *β*' ∈ *S* such that *β*3− *a*⇒ *β*' .

*i*

1. The states *β*'

1

and *β*'

3−*i*

become the current states.

= 3−*i*

A *play* is a maximal sequence of pairs of states formed by the players according to the rule described above, and starting from the initial states *α*1 and *α*2. The defender is the winner in every infinite play. A finite play is lost by the player who is stuck.

2

The following theorem is a standard one (see e.g. [[16](#_bookmark19),[17](#_bookmark20)]).

Theorem 2.1 *Processes α*1 *and α*2 *are weakly bisimilar iff the defender has a winning strategy (and nonbisimilar iff the attacker has a winning strategy).*

In what follows we shall frequently use a technique called ‘Defender’s Choice’ (abbriviated by DC). The idea is that the attacker in the bisimu- lation game starting from *α* and *β* can be forced by the defender to play a certain transition in the following sense: if the attacker takes any other avail- able transition (either from *α* or *β*), the defender can always answer in such a way that the resulting processes are weakly bisimilar (and hence the attacker loses).

A typical situation may look like

*α,, β ,*

*a cccc*

*cc*

*a ,,,a*

*,,*

*,,,a*

*a ,,,*

*c jc* J

'

*,*v*z*

Jv*z*

*α α*1 *α*2 *β*1 *β*2

where *αi* ≈ *βi* for 1 ≤ *i* ≤ 2 (very often *αi* and *βi* will be even syntactically equal). It is easy to see that in the bisimulation game starting from *α* and *β* the attacker is forced (DC) to take the transition *α* −*a*→ *α*'. In all other possible moves he immediately loses.

Let *Q* = {*p, q,.. .*}, Γ = {*X, Y,.. .*} and A*ct* = {*a, b,.. .*} be finite sets of

*control states*, *stack symbols* and *actions*, respectively, such that *Q* ∩ Γ = ∅

and *τ* ∈ A*ct* is the distinguished *silent action*. A *pushdown automaton* (PDA)

is a finite set ∆ of rewrite rules of the type *p* −*a*→ *qα* or *pX* −*a*→ *qα* where

*a* ∈ A*ct*, *p, q* ∈ *Q*, *α* ∈ Γ∗ and *X* ∈ Γ.

def

A pushdown automaton ∆ generates a labelled transition system *T* (∆) = (*Q* ×Γ∗*,* A*ct,* −→) where *Q* ×Γ∗ is the set of states [3](#_bookmark1) , A*ct* is the set of actions, and the transition relation −→ is defined by prefix-rewriting rules: *pγ* −*a*→ *qαγ* if (*p* −*a*→ *qα*) ∈ ∆, and *pXγ* −*a*→ *qαγ* if (*pX* −*a*→ *qα*) ∈ ∆ for all *γ* ∈ Γ∗.

A *parallel pushdown automaton* (PPDA) is defined in the same way as PDA. The only difference is that the states of the transition system generated by a PPDA system are considered modulo commutativity of the operator for

the composition of stack symbols. Hence rather than a sequential access to the stack (as in the case of PDA) we have a parallel access to all the symbols stored in the stack and the stack can be viewed as a multiset of stack symbols.

Example 2.2 Let ∆ def {*pX* −*a*→ *q*}. For PPDA there is a transition *pYX* −*a*→

=

*qY* but there is no such a transition when ∆ is interpreted as PDA.

Remark 2.3 For technical convenience (and w.l.o.g.) the rewrite rules for PDA and PPDA were introduced in a slightly more general form than usual.

*a*

Different definitions use only rules of the form *pX* −→ *qα*. Nevertheless,

the rules of the form *p* −*a*→ *qα* can be converted into this form by standard techniques.

Let N0

def

= {0*,* 1*,* 2*,* 3*,.. .*}. In what follows we will use the notation *A* for

*i*

0

=

=

a sequence of *i* ∈ N

occurrences of *A* ∈ Γ, i.e., *A*0 def *ϵ* and *Ai*+1 def *AiA*. By

#*A*(*γ*) we denote the number of occurrences of *A* ∈ Γ in the sequence *γ* ∈ Γ∗.

# High Undecidability of Weak Bisimilarity

In this section we prove that weak bisimilarity checking of PPDA and PDA is Σ1-hard. The proofs are by reduction from the recurrence problem of nonde- terministic Minsky machines. We describe first a general idea of the reduction and then show how it applies to PPDA and PDA.

1

* 1. *General Idea*

A *nondeterministic Minsky machine R* with two non-negative counters *c*1 and *c*2 is a finite sequence of *instructions R* = (*I*1*, I*2*, ... , In*) such that *n* ≥ 1 and every instruction *Ii*, 1 ≤ *i* ≤ *n*, is one of the following three types:

3 We write pα instead of (p, α) ∈ Q × Γ∗ where p is a control state and α is the stack content. A state pϵ ∈ Q × Γ∗, where ϵ stands for the *empty stack*, is written as p.

*increment i* : *cr* := *cr* + 1; goto *j*

*test and decrement i* : if *cr* = 0 then goto *j* else *cr* := *cr* − 1;goto *k*

*nondet. branching i* : goto (*j* or *k*)

where 1 ≤ *r* ≤ 2 and 1 ≤ *j, k* ≤ *n*.

Remark 3.1 W.l.o.g. we can assume that *I*1 is of the type ‘increment’.

A *conﬁguration* of *R* is a triple (*i, v*1*, v*2) ∈ {1*,..., n*}× N0 ×N0 where *i* is the label of the instruction to be executed, and *v*1 and *v*2 are the values of the counters *c*1 and *c*2, respectively. A *computational step ‹*→ between configurations is defined in the natural way.

The following recurrence problem P*rec* is Σ1-complete [[3](#_bookmark7)]: “given a nonde- terministic Minsky machine *R*, is there an infinite computation of *R* starting

1

at the instruction label 1 with both counters zero such that the instruction *I*1

is executed infinitely many times?”

We reduce the problem P*rec* to weak bisimilarity checking of PPDA and PDA. Given an instance *P* of P*rec* we construct a PPDA (PDA) system ∆

and a pair of processes *p*1 and *p*'

1

such that the answer to the problem *P* is

yes if and only if *p*1 ≈ *p*' .

1

The intuition is that a configuration (*i, v*1*, v*2) corresponds to a pair of pro- cesses *piγ* and *p*'*γ*' where *γ, γ*' ∈ {*C*1*, C*2*, A*}∗ such that #*C* (*γ*) = #*C* (*γ*') =

*i* 1 1

*v*1, #*C* (*γ*) = #*C* (*γ*') = *v*2, and #*A*(*γ*) = #*A*(*γ*') is the upper bound on

2

2

the number of steps before the instruction *I*1 is executed. In order to check whether *γ* and *γ*' contain the same number of occurrences of *C*1, *C*2 and *A* we shall introduce the following rules.

equal −*c*→1

equal equal −*c*→2

equal equal −*a*→ equal

equal *C*1 −→*c* equal equal *C*2 −*τ*→ equal equal *A* −*τ*→ equal

*C*1

*C*2 *A*

*C*1

*C*1 *C*1 *C*1 *C*1 *C*1

equal *C*1 −*τ*→ equal equal *C*2 −→*c* equal equal *A* −*τ*→ equal

*C*2 *C*2 *C*2

*C*2 *C*2 *C*2

equal *C*1 −*τ*→ equal equal *C*2 −*τ*→ equal equal *A* −→*c*

*A A A A A*

equal*A*

Lemma 3.2 *Let γ, γ*' ∈ {*C*1*, C*2*, A*}∗*. It holds that* equal *γ* ≈ equal *γ*' *if and only if* #*C* (*γ*) = #*C* (*γ*')*,* #*C* (*γ*) = #*C* (*γ*') *and* #*A*(*γ*) = #*A*(*γ*')*. It is*

1

1

2

2

*irrelevant whether the rules are interpreted as PPDA or PDA.*

Proof. In the first round the attacker selects a symbol to be tested by per- forming the action *c*1, *c*2 or *a*. In the successive rounds every occurrence of the selected symbol becomes visible under the action *c*. The *τ* rules simply remove the remaining symbols (these rules are needed only in the case of PDA).

Our aim is to design a set of rewrite rules ∆ such that both the attacker and the defender have the possibility to force the opponent to faithfully simulate the computation of *R*.

A single computational step (*i, v*1*, v*2)*‹*→(*i*'*, v*' *, v*' ) of the machine *R* is sim-

1 2

ulated by a finite number of rounds in the bisimulation game starting from

*piγ* and *p*'*γ*' such that *γ, γ*' ∈ {*C*1*, C*2*, A*}∗ where #*C* (*γ*) = #*C* (*γ*') = *v*1,

*i* ' ' 1 1

#*C*2 (*γ*) = #*C*2 (*γ* ) = *v*2 and #*A*(*γ*) = #*A*(*γ* ). Such a simulation consists of two phases: a *counting phase* and an *execution phase*.

In the counting phase the players move from *p*-control states *piγ* and *p*' *γ*' to *q*-control states *qiδ* and *q*'*δ*' such that the number of occurrences of the symbol *A* is altered while the number of occurrences of *C*1 and *C*2 is preserved. This phase depends on whether *i* = 1 (in this case the defender has the possibility to add an arbitrary number of the symbols *A* to both *γ* and *γ*') or whether *i >* 1 (in this case one occurrence of *A* is deleted from *γ* and *γ*').

*i*

*i*

In the execution phase starting from the *q*-control states *qiδ* and *q*'*δ*' the players execute the corresponding instruction *Ii* and modify the number of occurrences of *C*1 and *C*2 accordingly (hence reaching a new pair of *p*-control

*i*

' ' ' ' ' '

states *pi*' *ω* and *pi*' *ω* such that #*C*1 (*ω*) = #*C*1 (*ω* ) = *v*1, #*C*2 (*ω*) = #*C*2 (*ω* ) = *v*2

and #*A*(*ω*) = #*A*(*ω*') = #*A*(*δ*) = #*A*(*δ*')). In the case of nondeterministic branching the continuation of the game is determined by the defender (using DC).

This concludes the simulation of one computational step of *R* and the same

'

'

game repeats starting from *pi*' *ω* and *pi*' *ω*

executed in this step).

(the instruction *Ii*' is going to be

Since the players can force one another to follow the two phases described above, we are able to argue for the correctness of our reduction as follows.

* If there is an infinite computation of *R* where *I*1 is executed infinitely many times then let us fix such a computation. The defender can now force the attacker to simulate this computation in the bisimulation game from *p*1 and

' (initially both counters are empty). Moreover the defender is able to

*p*

1

add a sufficient number of the symbols *A* whenever the instruction *I*1 is executed and hence it is always possible to delete one occurrence of *A* in the counting phase of instructions different from *I*1. The bisimulation game becomes infinite and hence winning for the defender.

* If there is no infinite computation of *R* where *I*1 occurs infinitely often then the attacker can force the defender to simulate a particular computation (selected by the defender) and after finitely many rounds it is the case that the instruction *I*1 cannot be executed from that point (irrelevant of the choices for nondeterministic branching). Now the attacker continues to simulate the computation of *R*. Every computational step decreases the

number of occurrences of the symbol *A*. Hence after finitely many rounds the attacker wins (all occurrences of *A* are removed and this can be checked).

* 1. Σ1*-Completeness of Weak Bisimilarity for PPDA*

1

We shall now provide the reader with the necessary details of the construction described above. The PPDA rewrite rules are constructed in such a way that they enable a quick adaptation into PDA rules later on.

We start with the counting phase (i.e. moving from *p*-control states to

*q*-control states). The rules that prepare the execution of *I*1 are as follows.

*p*1 −*a*→ *r*1 *p*' −*a*→ *t*'

1

1

*p*1 −*a*→ *t*'

*t*' −*r*→ *t*' *A t*'

−*r*→ *r*'

1 1 1 1 1

*r*1 −*a*→ *s*1 *r*' −*a*→ *q*'

1 1

*s*1 −*r*→ *s*1*A s*1*A* −*r*→ *s*1 *s*1 −*r*→ *q*1 *r*' −*a*→ *s*1

1

*q*1 *c*−*h*→*eck* equal *q*' *c*−*h*→*eck* equal

1

Consider a bisimulation game starting from *p*1*γ* and *p*' *γ*' for some *γ, γ*' ∈

1

{*C*1*, C*2*, A*}∗ such that the number of occurrences of *C*1, *C*2 and *A* in *γ* and *γ*'

are equal.

*a*

In the first round the attacker is forced to play *p*1*γ* −→ *r*1*γ* (DC) and the

defender can answer by *p*' *γ*' =*a*⇒ ' *l*' ' for some *l*' ∈ N and hence add an

1 *r*1*A γ* 0

arbitrary number of the symbols *A*. If the defender stays in a state of the

form *t*' *Al*' *γ*' the attacker simply continues by using the rule *t*' −*r*→ *r*' and

1 1 1

since there are no *τ* -moves from *r*1*γ* both players can force the other one to

'

reach a pair of states *r γ* and ' *l* ' and it is the defender who chose *l*' ∈ N .

1 *r*1*A γ* 0

In the next round the attacker is forced to play '

*l*' ' −*a*→ '

*l*' ' (DC

*r*1*A γ*

*r*

*q*1*A γ*

– here the rule *s*1*A* −→ *s*1 is necessary) and the defender can answer by

*r*1*γ* =⇒*a q*1*Alγ* for some *l* ∈ N0 (in fact even some number of symbols *A* from

*γ* can be deleted but in this case the attacker wins as argued later on in this paragraph). As before, if the defender stays in the *s*1-state the attacker uses

the rule *s* −*r*→ *q* and since there are no *τ* rules out of the state ' *l*' ' the

1 1 *q*1*A γ*

'

game continues from the pair of *q*-control states *q Alγ* and ' *l* '. If *l* /= *l*'

1 *q*1*A γ*

then the attacker has the possibility to perform the action *check* and he wins

because of Lemma [3.2](#_bookmark3).

This means that after finitely many rounds the players can force the op-

ponent to reach a pair of states *q*1*Alγ* and *q*' *Alγ*' and it is the defender who is allowed to choose the number of occurrences of *A*.

1

The following rules decrease the number of occurrences of *A* by one and prepare the execution of the instructions *I*2*,..., In*. In the following rules let *i* range over {2*,..., n*} and let stop be a particular control state from which

no transitions are possible.

*p A c*−*ou*→*nt q p*' *A c*−*ou*→*nt q*'

*i i i* *i*

1. *c*−*h*→*eck* stop *p*'*A c*−*h*→*eck* stop

*i*

*i*

Consider a bisimulation game starting from *piγ* and *p*'*γ*' for some *γ, γ*' ∈

*i*

{*C*1*, C*2*, A*}∗ and 1 *< i* ≤ *n* such that the number of occurrences of *C*1, *C*2 and

*A* in *γ* and *γ*' are equal.

If #*A*(*γ*) = #*A*(*γ*') *>* 1 then after one round the players perform the action

*count* and reach the pair *q δ* and *q*'*δ*' such that *Aδ* = *γ* and *Aδ*' = *γ*' as desired.

*i* *i*

Should the attacker choose the action *check* the defender wins immediately.

On the other hand if # (*γ*) = # (*γ*') = 0 then the attacker wins by using the rule *p c*−*h*→*eck* stop to which the defender has no answer.

*i*

*A A*

We proceed by the execution phase (i.e. moving from *q*-control states to

*p*-control states).

For every *i*, 1 ≤ *i* ≤ *n*, such that *Ii* is of the type

*i* : *cr* := *cr* + 1; goto *j*

we have the following rules.

1. −*in*→*c*

*i*

*pjCr*

*q*' −*in*→*c*

*p*' *Cr*

In one round of the game starting from *qiδ* and *q*'*δ*' the players have only

*i*

*j*

' *i* '

one way to continue and reach the pair *pjCrδ* and *pj Crδ* . Hence they faithfully

simulate the corresponding computational step of the machine *R*.

For every *i*, 1 ≤ *i* ≤ *n*, such that *Ii* is of the type

*i* : goto (*j* or *k*)

we have the following rules.

*q* −*a*→ *qchoice*

*q*' −*a*→ *qleft*

*i i i* *i*

1. −*a*→ *qleft q*' −*a*→ *qright*

*i i i* *i*

1. −*a*→ *qright*

*i*

*i*

*qchoice*

*left*

*left*

*left* '

*left right*

*i* −→ *pj qi*

−→ *pj qi*

−→ *pk*

*qchoice right*

*right right* '

*right*

*left*

*i* −→ *pk qi*

−→ *pk qi*

−→ *pj*

Consider a bisimulation game starting from *qiδ* and *q*'*δ*' for some *δ, δ*' ∈

{*C*1*, C*2*, A*}∗

*i*

such that the number of occurrences of *C*1, *C*2 and *A* in *δ* and *δ*'

are equal. We claim that after two rounds of the game the players can force

the opponent to reach either *pjδ* and *p*' *δ*' or *pkδ* and *p*' *δ*' and it is the defender

*j k*

who decides between these two alternatives.

In the first round the attacker is forced to play *q δ* −*a*→ *qchoice δ* (DC) and

' ' *a*

*left* '

*i* *i*

' ' *a*

*right* '

the defender answers by (i) *qiδ* −→ *qi δ* or (ii) *qiδ* −→ *qi δ* . In the

second round starting from (i) *qchoiceδ* and *qleft δ*' or (ii) *qchoiceδ* and *qright δ*' the

*i i i* *i*

attacker is forced (DC) to play the action *left* in case (i) or the action *right* in

case (ii). This means that after two rounds the players reach the pair (i) *pjδ*

and *p*' *δ*' or (ii) *pkδ* and *p*' *δ*' according to the defender’s choice.

*j k*

The rules for the ‘test and decrement’ instructions start with similar rules as those for nondeterministic branching. First, the defender has the choice to determine whether the relevant counter is empty or not and the game continues according to this decision. After the defender’s move, the attacker has the possibility to check the correctness of the defender’s decision.

Hence for every *i*, 1 ≤ *i* ≤ *n*, such that *Ii* is of the type

*i* : if *cr* = 0 then goto *j* else *cr* := *cr* − 1; goto *k*

we have the following rules.

*q* −*a*→ *qchoice*

*q*' −*a*→ *qleft*

*i i i* *i*

*q* −*a*→ *qleft q*' −*a*→ *qright*

*i i i* *i*

*q* −*a*→ *qright*

*i*

*i*

*qchoice*

*left*

*left*

*left* '

*left right*

*i* −→ zero*i qi*

−→ zero*i qi*

−→ nonzero*i*

*qchoice right*

*right right*

' *right*

*left*

*i* −→ nonzero*i qi*

−→ nonzero*i qi*

−→ zero*i*

zero*i*

*i*

*p*

*k*

*k*

−*ze*→*ro p*

zero' −*ze*→*ro* '

nonzero*i*

*j*

*r*

*C* −*d*→*ec p*

nonzero' *Cr*

−*d*→*ec* '

zero

*c*−*h*→*eck z*

zero' *c*−*h*→*eck z*'

*i*

*zrCr*

−→*b*

*r* *i*

stop *z*' *Cr*

*r*

−→*c*

*r*

stop

*z C* −*r*→ *z*

*z A* −*r*→ *z*

*z*' *C*

−*r*→ *z*'

*z*' *A* −*r*→ *z*'

*r* 3−*r r r r*

*r* 3−*r r r r*

nonzero

*c*−*h*→*eck n*

nonzero' *c*−*h*→*eck n*'

*i r i r*

*n* −→*b*

*r*

stop *nrCr*

−→*c*

stop *n*' *Cr*

−→*b*

stop *n*'

−→*c*

stop

Consider a bisimulation game starting from *qiδ* and *q*'*δ*' for some *δ, δ*' ∈

*i*

*p*

*r*

*r*

*i*

*j*

{*C*1*, C*2*, A*}∗ such that the number of occurrences of *C*1, *C*2 and *A* in *δ* and *δ*'

are equal.

The same arguments as before apply to show that after two rounds of the game the players can force the opponent to reach the pair (i) zero*i δ* and

zero' *δ*' or (ii) nonzero*i δ* and nonzero' *δ*' according to the defender’s choice.

*i* *i*

The attacker can now verify the correctness of the defender’s decision by playing the action *check* and thus forcing the defender to reach a pair of states

starting with either (i) *zr* and *z*' or (ii) *nr* and *n*' . The following two lemmas

*r*

*r*

show that in this case the attacker wins if and only if the defender cheated.

Lemma 3.3 *Let δ, δ*' ∈ {*C*1*, C*2*, A*}∗ *and r* ∈ {1*,* 2}*. It holds that zrδ* ≈ *z*' *δ*'

*r*

*if and only if* 0 = #*C* (*δ*) = #*C* (*δ*')*. It is irrelevant whether the rules are*

*r r*

*interpreted as PPDA or PDA.*

Proof. Obvious.

Lemma 3.4 *Let δ, δ*' ∈ {*C*1*, C*2*, A*}∗ *and r* ∈ {1*,* 2}*. It holds (in the case of PPDA) that nrδ* ≈ *n*' *δ*' *if and only if* 1 ≤ #*C* (*δ*)*,* #*C* (*δ*')*. It also holds (in*

*r r r*

*the case of PDA) that nrδ* ≈ *n*' *δ*' *if and only if both δ and δ*' *begin with Cr.*

*r*

Proof. Obvious.

In order to finish the simulation of the ‘test and decrement’ instruction the players have only one continuation of the game from (i) zero*i δ* and zero' *δ*' or

*i*

(ii) nonzero*i δ* and nonzero' *δ*'. In case (i) they perform the action *zero* and

*i* ' '

reach a new pair of states *pjδ* and *pjδ* . In case (ii) they perform the action *dec* and decrease the number of occurrences of *Cr* by one. After this they continue from the pair *pkω* and *p*' *ω*' such that *Crω* = *δ* and *Crω*' = *δ*'.

*k*

In order to conclude the proof recall that both players can force the op- ponent to faithfully simulate computational steps of the machine *R* and the defender has the choice in the case of nondeterministic branching. Moreover, whenever the instruction *I*1 is performed, the defender can generate enough of the symbols *A* to ensure that he does not lose before the instruction *I*1 is executed again.

If there is a computation of the machine *R* which visits *I*1 infinitely many times, the defender simulates such a computation in the bisimulation game and he wins (either because the attacker decides not to cooperate during the simulation, or because the game becomes infinite). On the other hand, if along every computational path the instruction *I*1 is executed only finitely many times, the attacker wins since the symbols *A* generated by the defender will be exhausted eventually.

Hence the answer to the given recurrence problem of nondeterministic Minsky machines is positive if and only if *p*1 ≈ *p*' .

1

Theorem 3.5 *Weak bisimilarity checking of PPDA (and PN) is* Σ1*-complete.*

1

Proof. The hardness of the problem follows from the construction described above, and the containment of the problem for PN (and PPDA) in Σ1 was established in [[5](#_bookmark9)].

1

* 1. Σ1*-Completeness of Weak Bisimilarity for PDA*

1

We will now proceed by showing how to adapt the PPDA rules for the case of PDA. Obviously, all the PPDA rules defined above that do not remove any symbol from the stack can be used also for PDA. There are, however, three situations where a symbol is removed from the stack. First, there is the rule *s*1*A* −*r*→ *s*1 in the counting phase of the instruction *I*1. Since it is sufficient that the rule removes only the occurrences of *A* added in the previous round (and hence on the top of the stack), no change is needed in this case. The second place where a symbol is removed is in the counting phase of the instructions *I*2*,... , In* and the third place are the rules for ‘test and decrement’ instructions

in the case of nonzero value of the corresponding counter (recall that *I*1 is of

the type ‘increment’ by Remark [3.1](#_bookmark2)).

Fortunately, the outlined problems can be solved by one modification in the construction. The idea is that before each counting phase, the defender is allowed to rearrange the content of the stacks (while preserving the number of occurrences of *C*1, *C*2 and *A*) in such a way that the stacks are equal (in order to apply DC) and the symbol *A* is on the top of them. If the execution phase continues by a ‘test and decrement’ instruction and the corresponding counter *cr* is nonempty, the defender makes sure that the symbol *Cr* follows

immediately after *A*.

This is formally described as follows. For all *i*, 1 *< i* ≤ *n*, we remove the

*count*

*p A* −→

PPDA rules for the counting phase (i.e. the rules

*q* , *p*' *A c*−*ou*→*nt*

*q*',

*i i i* *i*

*i*

*p c*−*h*→*eck*

*i*

stop and *p*'*A c*−*h*→*eck*

stop) and replace them with the following rules

where *X* ranges over the set {*C*1*, C*2*, A*}.

*p* −*a*→ *r p*' −*a*→ *t*'

*i i i* *i*

*p* −*a*→ *t*' *t*' −*r*→ *t*' *X t*'*X* −*r*→ *t*'

*i i i i i* *i*

*t*' −*r*→ *r*'

*i* *i*

1. *c*−*h*→*eck* equal *r*' *c*−*h*→*eck* equal

*i*

*i*

1. −*a*→ *s r*' −*a*→ *u*'

*i* *i*

1. −*r*→ *s X s X* −*r*→ *s*

*i i* *i*

*i* *i*

*r*' −*a*→ *s*

*i i* *i*

*s* −*r*→ *u*

*i* *i*

*u c*−*h*→*eck* equal *u*' *c*−*h*→*eck* equal

*i*

*i*

*u A c*−*ou*→*nt q u*' *A c*−*ou*→*nt q*'

*i i i* *i*

*u* −→*b* stop *u A* −→*c* stop *u*'*A* −→*b* stop *u*' −→*c*

*i*

*i*

*i*

*i*

stop

Consider a bisimulation game played from *piγ* and *p*'*γ*' such that #*C* (*γ*) =

*i* 1

#*C* (*γ*'), #*C* (*γ*) = #*C* (*γ*') and #*A*(*γ*) = #*A*(*γ*'). We claim that after two

1

2

2

rounds of the game the players can force the opponent to reach a pair of states

*u δ* and *u*'*δ*' such that # (*δ*) = #

(*δ*') = # (*γ*) = # (*γ*'), #

(*δ*) =

*i i C*1 *C*1

*C*1 *C*1 *C*2

#*C* (*δ*') = #*C* (*γ*) = #*C* (*γ*') and #*A*(*δ*) = #*A*(*δ*') = #*A*(*γ*) = #*A*(*γ*') and it

2

2

2

is the defender who selects such *δ* and *δ*'.

In the first round starting from *piγ* and *p*'*γ*' the attacker is forced (DC) to

*a i a*

play *piγ* −→ *riγ*. The defender answers by *p*'*γ*' =⇒ *r*'*δ*' (should the defender

' ' *a* ' '

*i* *i*

' ' *r* ' '

answer only by *piγ*

=⇒ *tiδ* , the attacker plays *tiδ*

−→ *riδ*

and the defender

can only respond by staying in *riγ*).

The game continues by the second round from the states *riγ* and *r*'*δ*'. (If #*C* (*γ*) /= #*C* (*δ*'), or #*C* (*γ*) /= #*C* (*δ*'), or #*A*(*γ*) /= #*A*(*δ*') the attacker

*i*

1

1

2

2

plays the action *check* and wins from the states equal *γ* and equal *δ*' because

of Lemma [3.2](#_bookmark3).)

In the second round the attacker is forced (DC) to play *r*'*δ*' −*a*→ *u*'*δ*' and

*a i* *i*

the defender answers by *riγ* =⇒ *uiδ* (should the defender answer only by

*r γ* =*a*⇒ *s δ*, the attacker plays *s δ* −*r*→ *u δ* and the defender can only respond

*i i i* *i*

by staying in *u*'*δ*').

*i*

Again, by performing the action *check* the attacker can validate that the number of occurrences of *C*1, *C*2 and *A* in *δ* and *δ*' are the same.

Hence after two rounds the players can force the opponent to reach the states *uiδ* and *u*'*δ*' such that #*C* (*δ*) = #*C* (*δ*') = #*C* (*γ*) = #*C* (*γ*'), #*C* (*δ*) =

' *i* ' 1 1 ' 1 1 ' 2

#*C*2 (*δ* ) = #*C*2 (*γ*) = #*C*2 (*γ* ) and #*A*(*δ*) = #*A*(*δ* ) = #*A*(*γ*) = #*A*(*γ* ) and it

was the defender who rearranged the stack contents. In particular the defender can ensure that the stacks are equal in order to apply DC.

This also means that the defender had the chance to place the symbol *A* on the top of the stacks *δ* and *δ*' and hence the game continues by performing the action *count* as in the case of PPDA. If the defender didn’t place the symbol

*A* on the top of both stacks or there were no *A*’s in *γ* and *γ*', it is easy to see that the attacker wins by performing either the action *b* or *c*.

Moreover, if the instruction *Ii* is of the type ‘test and decrement’ and the tested counter *cr* is nonempty, the defender was forced to place the sym- bols *Cr* on both stacks as the second from the top so that the rewrite rules

*i*

nonzero*i Cr*

−*d*→*ec*

*pk* and nonzero' *Cr*

−*d*→*ec*

' are applicable later on. If not then

*check*

*p*

*k*

the defender loses because the attacker can use the rules nonzero*i* −→ *nr* and

nonzero' *c*−*h*→*eck n*'

and he wins because of Lemma [3.4](#_bookmark4).

*i r*

To sum up, after the presented modification, the arguments for the cor- rectness of our reduction are valid also for PDA.

Theorem 3.6 *Weak bisimilarity checking of PDA is* Σ1*-complete.*

1

Proof. The hardness of the problem was shown above, and the containment of the problem in Σ1 follows from [[5](#_bookmark9)].

1

Remark 3.7 A general reduction from weak bisimilarity of PDA to weak bisimilarity of normed (from every reachable configuration it is possible to empty the stack) PDA described in [[14](#_bookmark17)] implies that weak bisimilarity of normed PDA is also Σ1-complete. A similar reduction works also for PPDA which are normed in the same sense as PDA.

1

# Conclusion

We have proved that weak bisimilarity problems for PDA and PPDA (and also for PN) are Σ1-hard and hence Σ1-completeness of the problems is established.

1 1

We believe that the ideas of the presented reductions will find their appli- cations also in other classes of infinite-state systems. A particular challenge is to show high undecidability or even Σ1-completeness of weak bisimilarity checking for process formalisms like PA-processes and one-counter processes. These problems are known to be undecidable [[15](#_bookmark21),[10](#_bookmark14)] but their classification in the hierarchy of undecidable problems is open.

1

Acknowledgment

I would like to thank Moshe Vardi for drawing my attention to the topic, Petr Janˇcar for many comments and suggestions, and the anonymous referees for their remarks.

# References

1. Burkart, O., D. Caucal, F. Moller and B. Steffen, *Veriﬁcation on inﬁnite structures*, in:

J. Bergstra, A. Ponse and S. Smolka, editors, *Handbook of Process Algebra*, Elsevier Science, 2001 pp. 545–623.

1. Burkart, O., D. Caucal and B. Steffen, *An elementary decision procedure for arbitrary context-free processes*, in: *Proceedings of the 20th International Symposium on Mathematical Foundations of Computer Science (MFCS’95)*, LNCS 969 (1995), pp. 423–433.
2. Harel, D., *Effective transformations on inﬁnite trees, with applications to high undecidability, dominoes, and fairness*, Journal of the ACM (JACM) 33 (1986), pp. 224–248.
3. Hu¨ttel, H., *Undecidable equivalences for basic parallel processes*, in: *Proceedings of the 2nd International Symposium on Theoretical Aspects of Computer Software (TACS’94)*, LNCS 789 (1994), pp. 454–464.
4. Janˇcar, P., *High undecidability of weak bisimilarity for Petri nets*, in: *Proceedings of Colloquium on Trees in Algebra and Programming (CAAP’95)*, LNCS 915 (1995), pp. 349–363.
5. Janˇcar, P., *Undecidability of bisimilarity for Petri nets and some related problems*, Theoretical Computer Science 148 (1995), pp. 281–301.
6. Janˇcar, P., *Strong bisimilarity on basic parallel processes is PSPACE-complete*, in: *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS’03)*, (2003), pp. 218–227.
7. Kuˇcera, A. and P. Janˇcar, *Equivalence-checking with inﬁnite-state systems: Techniques and* *results*, in: *Proceedings of the 29th Annual Conference on Current Trends in Theory and Practice of Informatics (SOFSEM’02)*, LNCS 2540 (2002), pp. 41–73.
8. Mayr, R., *Process rewrite systems*, Information and Computation 156(1) (2000), pp. 264–286.
9. Mayr, R., *Undecidability of weak bisimulation equivalence for 1-counter processes*, in: *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming (ICALP’03)*, LNCS 2719 (2003), pp. 570–583.
10. S´enizergues, G., *Decidability of bisimulation equivalence for equational graphs of ﬁnite out- degree*, in: *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS’98)* (1998), pp. 120–129.
11. Srba, J., *Roadmap of inﬁnite results*, Bulletin of the European Association for Theoretical Computer Science (Columns: Concurrency) 78 (2002), pp. 163–175, updated online version: <http://www.brics.dk/~srba/roadmap>.
12. Srba, J., *Strong bisimilarity and regularity of basic parallel processes is PSPACE-hard*, in: *Proceedings of the 19th International Symposium on Theoretical Aspects of Computer Science (STACS’02)*, LNCS 2285 (2002), pp. 535–546.
13. Srba, J., *Undecidability of weak bisimilarity for pushdown processes*, in: *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR’02)*, LNCS 2421 (2002),

pp. 579–593.

1. Srba, J., *Undecidability of weak bisimilarity for PA-processes*, in: *Proceedings of the 6th International Conference on Developments in Language Theory (DLT’02)*, LNCS 2450 (2003),

pp. 197–208.

1. Stirling, C., *Local model checking games*, in: *Proceedings of the 6th International Conference on Concurrency Theory (CONCUR’95)*, LNCS 962 (1995), pp. 1–11.
2. Thomas, W., *On the Ehrenfeucht-Fra¨ıss´e game in theoretical computer science (extended abstract)*, in: *Proceedings of the 4th International Joint Conference CAAP/FASE, Theory and Practice of Software Development (TAPSOFT’93)*, LNCS 668 (1993), pp. 559–568.