HIGHLY UNDECIDABLE QUESTIONS FOR PROCESS ALGEBRAS[*]

Petr Jančar
Department of Computer Science, Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava - Poruba, Czech Republic
Petr.Jancar@vsb.cz

Jiří Srba
BRICS[†], *Department of Computer Science, University of Aalborg*
Fredrik Bajersvej 7B, 9220 Aalborg East, Denmark
srba@brics.dk

Abstract We show Σ_1^1-completeness of weak bisimilarity for PA (process algebra), and of weak simulation preorder/equivalence for PDA (pushdown automata), PA and PN (Petri nets). We also show Π_1^1-hardness of weak ω-trace equivalence for the (sub)classes BPA (basic process algebra) and BPP (basic parallel processes).

Keywords: Weak bisimilarity, simulation, trace preorder, high undecidability

1. Introduction

In the area of verification, the possibilities of checking behavioural equivalences and/or preorders of systems are a natural object to study, which includes various decidability and complexity questions. A part of research effort has been aimed at bisimulation equivalence (bisimilarity) and simulation preorder, since these had been recognized as fundamental notions. We are interested in infinite-state systems, for which recent surveys of results have been given, e.g., in [Burkart et al., 2001, Kučera and Jančar, 2002, Srba, 2002].

The systems we study can be uniformly defined by means of process rewrite systems (PRS) — see Figure 1 for the PRS-hierarchy from [Mayr, 2000]; the second and the third level from the bottom is the focus of our interest. We now

[*]Both authors are partly supported by the Grant Agency of the Czech Rep., grant No. 201/03/1161.
[†]**B**asic **R**esearch in **C**omputer **S**cience,
Centre of the Danish National Research Foundation.

provide a selection of some results relevant to our paper (all references can be found in [Srba, 2002]).

(Strong) bisimilarity is already well known to be decidable for the class BPA (basic process algebra, or basic sequential processes), i.e., the class of labelled transition systems generated by left-most derivations of context-free grammars in Greibach normal form; the states correspond to finite sequences of nonterminals which are composed sequentially and only the first one, say *X*, can be rewritten according to a rule $X \xrightarrow{a} \alpha$ while emitting an action a (so for a state $X\beta$ we have $X\beta \xrightarrow{a} \alpha\beta$). Bisimilarity is also known to be decidable for BPP (basic parallel processes); the only difference with BPA is that nonterminals are viewed as composed in parallel, i.e., each can be rewritten. (We can mention also the recent result [Jančar et al., 2003] showing the decidability for the union of BPA and BPP.) An involved result by Sénizergues (later strengthened and simplified by Stirling) showed the decidability even for PDA – labelled transition systems generated by pushdown automata (where a state (p, α) comprises a control state and a sequence of stack symbols). For PN (labelled place/transition Petri nets) bisimilarity is known to be undecidable; this even holds for the subclass PPDA (pushdown automata with stack symbols composed in parallel), which lies strictly between BPP and PN. For the class PA (where the right-hand sides of grammar rules can contain a mixture of sequential and parallel compositions), the decidability question is still open. *(Strong) simulation preorder* is undecidable (already) for both BPA and BPP – as well as classical language equivalence and its modification called *trace equivalence.*

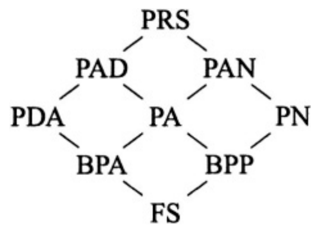

Figure 1. PRS-hierarchy

We can naturally ask similar questions for models with silent (internal) actions, and explore weak bisimilarity and weak simulation. Decidability of *weak bisimilarity* is still open for both BPA and BPP. From [Srba, 2003a] it is known to be highly undecidable for PDA and PN, more precisely, complete for the level Σ^1_1 of the analytical hierarchy (i.e., it can be described by a formula $\exists X.\phi(\ldots, X, \ldots)$ where ϕ is a first-order arithmetical formula containing the predicate *X*; we refer to [Rogers, 1967] for further details about arithmetical and analytical hierarchies). For PA, weak bisimilarity was recently proved undecidable in [Srba, 2003b] but the absence of a control unit seemed to prevent a reduction showing Σ^1_1-hardness; so this problem was left open. In fact, such questions might not seem very relevant from the 'practical' point of view, nevertheless we believe that categorizing undecidable problems according to their degrees of undecidability is still useful for deeper understanding of the studied problems. We can also recall the general experience that the 'natural' undecidable problems (in computer science) are either on the lowest levels of the arithmetical hierarchy or on the lowest levels of the analytical hierarchy (see, e.g., [Harel, 1986]).

In this paper we succeeded in modelling a sufficient fragment of the (missing) finite-control unit, which enabled us to show Σ_1^1-completeness of weak bisimilarity also for PA. We then use some modifications of the developed reductions to show Σ_1^1-completeness of *weak simulation preorder/equivalence* for all the classes PDA, PA and PN (in fact, again even for PPDA).

Weak trace preorder/equivalence is easily shown to be in Π_1^0, i.e., (very) low in the arithmetical hierarchy. This seems to contradict the experience from the strong case (without silent actions) where the complexity increases in the direction: bisimulation – simulation – trace. We give some results indicating that when taking infinite traces (ω-traces) into account, the mentioned 'contradiction' disappears; in particular we show Π_1^1-hardness of *weak ω-trace preorder/equivalence* for both BPA and BPP.

We also show that *weak regularity checking* (checking if a given system is weakly bisimilar to some finite-state one) is 'easier', by which we mean at most hyperarithmetical, for any reasonable process algebra. Finally we add a few observations about Σ_1^1-completeness of *branching bisimilarity* for PDA and PPDA.

Note: a full version of this paper appears as [Jančar and Srba, 2004].

2. Basic Definitions

A *labelled transition system* (LTS) is a triple $(S, Act, \longrightarrow)$ where S is a set of *states* (or *processes*), Act is a set of *labels* (or *actions*), and $\longrightarrow \subseteq S \times Act \times S$ is a *transition relation;* for each $a \in Act$, we view $\xrightarrow{a}$ as a relation on S where $\alpha \xrightarrow{a} \beta$ iff $(\alpha, a, \beta) \in \longrightarrow$. We assume that Act contains a distinguished *silent action* τ. The *weak transition relation* $\Longrightarrow$ is defined by $\overset{a}{\Longrightarrow} \overset{\text{def}}{=} (\xrightarrow{\tau})^* \circ \xrightarrow{a} \circ (\xrightarrow{\tau})^*$ for $a \in Act \smallsetminus \{\tau\}$, and $\overset{a}{\Longrightarrow} \overset{\text{def}}{=} (\xrightarrow{\tau})^*$ for $a = \tau$.

Given $(S, Act, \longrightarrow)$, a binary relation $R \subseteq S \times S$ is a *weak simulation* iff for each $(\alpha, \beta) \in R$, $a \in Act$, and α' such that $\alpha \xrightarrow{a} \alpha'$ there is β' such that $\beta \overset{a}{\Longrightarrow} \beta'$ and $(\alpha', \beta') \in R$. A *weak bisimulation* is a weak simulation which is a symmetric relation. We say that a process α *is simulated* by a process β, denoted $\alpha \sqsubseteq_s \beta$, if there is a weak simulation containing (α, β). Processes α and β are *simulation equivalent,* denoted $\alpha =_s \beta$, if $\alpha \sqsubseteq_s \beta$ and $\beta \sqsubseteq_s \alpha$. Processes α and β are *weakly bisimilar,* denoted $\alpha \approx \beta$, if there is a weak bisimulation containing (α, β).

We shall use standard game-theoretic characterizations of the introduced notions. A (weak) *bisimulation game* on a pair of processes α_1 and α_2 is a two-player game between 'Attacker' and '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 Attacker chooses $i \in \{1,2\}$, $a \in Act$ and $\beta_i' \in S$ such that $\beta_i \xrightarrow{a} \beta_i'$.

2 Defender responds by choosing $\beta_{3-i}' \in S$ such that $\beta_{3-i} \xRightarrow{a} \beta_{3-i}'$.

3 States β_1' and β_2' become the current states.

A *play* is a maximal sequence of pairs of states formed by the players according to the rule described above, starting from the initial states α_1 and α_2. Defender is the winner in every infinite play. A finite play is lost by the player who is stuck. A (weak) *simulation game* is played similarly, the only change is that Attacker is bound to choose $i = 1$ (thus playing in the "left process" only).

PROPOSITION 1 *It holds that $\alpha_1 \approx \alpha_2$ (resp. $\alpha_1 \sqsubseteq_s \alpha_2$) iff Defender has a winning strategy in the bisimulation (resp. simulation) game from α_1 and α_2.*

PA-processes

Let $Const$ be a set of *process constants.* The class of *process expressions* over $Const$ is given by $E ::= \epsilon \mid X \mid E.E \mid E\|E$ where 'ϵ' is the *empty process, X* ranges over $Const$, '.' is the operator of *sequential composition,* and '$\|$' stands for a *parallel composition.* We do not distinguish between process expressions related by a *structural congruence,* which is the smallest congruence respecting that '.' is associative, '$\|$' is associative and commutative, and 'ϵ' is a unit for '.' and '$\|$'. We shall adopt the convention that the sequential operator binds tighter than the parallel one. Thus, for example, $X.Y\|Z$ means $(X.Y)\|Z$.

A *PA process rewrite system* ((1, *G*)-PRS in the terminology of [Mayr, 2000]) Δ is a finite set of *rules* of the form $X \xrightarrow{a} E$, where $X \in Const$, $a \in Act$ and *E* is a process expression. Let us denote the set of actions and the set of process constants that appear in Δ as $Act(\Delta)$ and $Const(\Delta)$, respectively. (Note that these sets are finite).

A PA system Δ determines a labelled transition system where the process expressions over $Const(\Delta)$ are the states and $Act(\Delta)$ is the set of labels. The *transition relation* is the least relation satisfying the following SOS rules (recall that '$\|$' is commutative):

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

A process constant $D \in Const(\Delta)$ is called a *deadlock* iff Δ contains no rule $D \xrightarrow{a} E$ for any *E.* In the usual presentation of PA it is often assumed that Δ contains no deadlocks.

PDA, PPDA, BPA and BPP processes

Let $Q = \{p, q, \ldots\}$, $\Gamma = \{X, Y, \ldots\}$ and $Act = \{a, b, \ldots\}$ be finite sets of *control states, stack symbols* and *actions,* respectively, such that $Q \cap \Gamma = \emptyset$ and $\tau \in Act$ is the distinguished silent action. A *PDA system* (or a pushdown automaton) Δ is a finite set of rewrite rules of the type $p \xrightarrow{a} q\alpha$ or $pX \xrightarrow{a} q\alpha$ where $a \in Act$, $p, q \in Q$, $X \in \Gamma$ and $\alpha \in \Gamma^*$. Such a PDA system generates a labelled transition system where $Q \times \Gamma^*$ is the set of states, Act is the set of actions, and the transition relation is defined by prefix-rewriting rules: $(p \xrightarrow{a} q\alpha) \in \Delta$ $((pX \xrightarrow{a} q\alpha) \in \Delta)$ implies $p\gamma \xrightarrow{a} q\alpha\gamma$ $(pX\gamma \xrightarrow{a} q\alpha\gamma)$ for all $\gamma \in \Gamma^*$. A *PPDA system* (a parallel pushdown automaton) is defined in the same way as a PDA system but the composition of stack symbols is now viewed as commutative, i.e., 'parallel'. (So each symbol stored in the stack is directly accessible and the stack can be viewed as a multiset of stack symbols.) A PDA (resp. PPDA) system is called BPA for *basic process algebra* (resp. BPP for *basic parallel processes*) whenever the set of control states is singleton. The classes BPA, BPP, PDA and PA correspond directly to the classes from the PRS hierarchy in Figure 1. The class PPDA is positioned strictly between BPP and PN. Hence all the lower bounds we shall prove for PPDA immediately apply also to PN.

Defender's Choice Technique

In what follows we shall frequently use a technique called 'Defender's Choice' (abbreviated by DC). The idea is that Attacker in the (bi)simulation game starting from α and β can be forced by Defender to play a certain transition in the following sense: if Attacker takes any other available transition, Defender can answer in such a way that the resulting processes are guaranteed to be (bi)similar (and hence Attacker loses). A typical situation in the case of bisimilarity may look like in Figure 2 part a) where $\alpha_i \approx \beta_i$ for all $i \geq 1$ (very often α_i and β_i will be even syntactically equal). It is easy to see that in the bisimulation game starting from α and β Attacker is forced (DC) to take the transition $\alpha \xrightarrow{a} \alpha'$. In all other possible moves he loses.

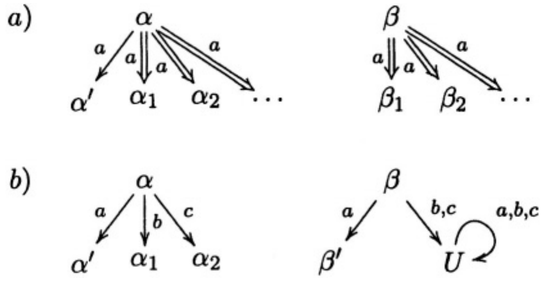

Figure 2. **Defender's Choice**

In the case of simulation game, Defender can also use another way to force Attacker to perform a certain move. Defender can threaten to enter a *universal state,* i.e., a state where all available actions are constantly enabled. The situation may look like in Figure 2 part b). Obviously Attacker who is playing in the left process is forced (DC) to perform the action a to which Defender can answer only by the same action; the players then continue from the pair α' and β'. Should Attacker play b or c in the first round, Defender answers by the same action and enters the universal state *U*. From now on Defender can answer to all Attacker's moves and clearly wins.

3. Σ_1^1-completeness of weak (bi)similarity problems

From [Srba, 2003a] we know that weak bisimilarity is Σ_1^1-complete on PDA and PPDA. For PA only undecidability was known [Srba, 2003b] and it was not clear how to simulate "finite-control unit features" which would allow to derive high undecidability as well. Here we answer this question by showing Σ_1^1-completeness also for PA. We then add the Σ_1^1-completeness results for weak simulation preorder (and equivalence) on all the classes PDA, PA and PPDA. Finally we sketch an extension of the results to branching bisimilarity on PDA and PPDA.

We first observe that the mentioned problems are in Σ_1^1: the expression "there exists a set of pairs which contains (P_1, P_2) and is a weak bisimulation (a weak simulation)" can be routinely transformed into a Σ_1^1-formula. For this, it is sufficient that the relations $\xrightarrow{a}$ and $\overset{a}{\Longrightarrow}$ are arithmetical (which is obviously true for any reasonable process algebra like PRS); in fact, these relations are even decidable for the classes PDA, PA and PPDA which we are primarily interested in.

The Σ_1^1-hardness results are achieved by (algorithmic) reductions from suitable problems which are known to be Σ_1^1-complete. One of them is the following:

Problem: *Recurrent Post's correspondence problem (rPCP)*
Instance: Two sequences $A = [u_1, u_2, \ldots, u_n]$, $B = [v_1, v_2, \ldots, v_n]$ $(n \geq 1)$ of nonempty words over an alphabet Σ such that $|u_i| \leq |v_i|$ for all i, $1 \leq i \leq n$.
Question: Is there an infinite sequence of indices $i_1, i_2, i_3, \ldots$ from the set $\{1, 2, \ldots, n\}$ in which the index 1 appears infinitely often and for which the infinite words $u_{i_1} u_{i_2} u_{i_3} \cdots$ and $v_{i_1} v_{i_2} v_{i_3} \cdots$ are equal ?

Such an infinite sequence $i_1, i_2, i_3, \ldots$ is called a *solution* of the instance *(A, B)*. Any finite sequence $i_1, i_2, \ldots, i_m$ is called a *partial solution* of *(A, B)* iff $u_{i_1} u_{i_2} \cdots u_{i_m}$ is a prefix of $v_{i_1} v_{i_2} \cdots v_{i_m}$.

REMARK 2 *The problem rPCP is usually defined without the condition* $|u_i| \leq |v_i|$*; we have included this additional requirement since it is technically convenient and can be easily shown not to affect the following theorem.*

THEOREM 3 *([Harel, 1986]) Problem rPCP is Σ_1^1-complete.*

Let us now fix an instance *(A, B)* of rPCP, over an alphabet Σ, where $A = [u_1, \dots, u_n]$ and $B = [v_1, \dots, v_n]$. A solution of *(A, B),* if it exists, can be naturally represented by an infinite sequence of process constants from $\{V_1, V_2, \dots, V_n\}$; the sequence can be divided into finite segments, where a *segment* is defined as a sequence from $\{V_2, V_3, \dots, V_n\}^* \cdot \{V_1\}$. We note that an infinite sequence composed from segments represents a solution of *(A, B)* iff all its finite prefixes represent partial solutions, which is equivalent to saying that infinitely many of its finite prefixes represent partial solutions.

A general idea behind our reductions can be described as the following game (which is then concretely implemented in the particular cases we study). Starting from the empty sequence (viewed as a partial solution), Attacker can repeatedly request Defender to prolong the so far constructed partial solution by adding a further segment (for which the implementations will use sequences of τ-moves). Besides the mentioned request, Attacker has also a possibility to enter a checking phase to verify that the (so far) constructed sequence indeed represents a partial solution – if it does not then Attacker wins, and if it does then Defender wins. This means that Defender has a winning strategy if and only if there is an (infinite) solution of the (*A*, *B*)-instance.

We now describe a concrete implementation for weak bisimilarity of PA. We show an (algorithmic) construction of a PA system Δ with a pair of processes P_1 and P_2 such that

$$(A, B) \text{ has a solution } \Longleftrightarrow P_1 \approx P_2 . \qquad (1)$$

We present Δ in a stepwise manner, always giving a piece of it together with several useful observations (which should make the verification of the desired property straightforward).

In the construction we use a distinguished process constant *D* which is a *deadlock,* i.e., there are no rules with *D* on the left-hand side. Particularly useful for us is to note that $\alpha.D.\beta \parallel \gamma \approx \alpha \parallel \gamma$. Later on we show that using the deadlock is not essential (just technically convenient).

Our first intention is to arrange that the bisimulation game will start from the pair (X, X') and continue through some pairs $(X.\alpha_1, X'.\alpha_1)$, $(X.\alpha_2.\alpha_1, X'.\alpha_2.\alpha_1)$, $(X.\alpha_3.\alpha_2.\alpha_1, X'.\alpha_3.\alpha_2.\alpha_1)$, ...where α_i's are reversed segments which are chosen by Defender (using DC, i.e. Defender's Choice technique). Let us look at the rules in the groups I and II.

I	$X \xrightarrow{a} X_1'$	$X' \xrightarrow{a} X_1'$	
		$X_1' \xrightarrow{\tau} X_1'.V_i$	for each $i \in \{2, 3, \dots, n\}$
	$X \xrightarrow{a} Y$	$X_1' \xrightarrow{\tau} Y'.V_1$	
II	$Y \xrightarrow{a} Y_1.D$	$Y' \xrightarrow{a} Y_1.D$	
	$Y_1 \xrightarrow{\tau} Y_1.V_i$		for each $i \in \{2, 3, \dots, n\}$
	$Y_1 \xrightarrow{\tau} X.V_1$	$Y' \xrightarrow{a} X'$	

According to these rules, when starting from the pair $(X.\alpha, X'.\alpha)$, Attacker is forced (DC) to perform $X.\alpha \xrightarrow{a} Y.\alpha$, otherwise Defender can reach a syntactic equality. Defender can be then viewed as forced to respond by $X' \overset{a}{\Longrightarrow} Y'.\beta.\alpha$ for a (reversed) segment β of his choice. If he does not finish by using the rule $X'_1 \xrightarrow{\tau} Y'.V_1$, Attacker can perform a move according to this rule in the next round — thus installing a pair $(Y.\alpha, Y'.\beta.\alpha)$ anyway.

Rules in II make clear that Attacker is now forced (DC) to move $Y'.\beta.\alpha \xrightarrow{a} X'.\beta.\alpha$ and Defender can respond by $Y.\alpha \overset{a}{\Longrightarrow} X.\beta.\alpha.D.\alpha$; since D is a deadlock, we can view the installed pair as $(X.\beta.\alpha, X'.\beta.\alpha)$. Similarly as above, Defender cannot gain by not using the rule $Y_1 \xrightarrow{\tau} X.V_1$. As we shall see later, he neither can gain by installing $X.\gamma$ for $\gamma \neq \beta.\alpha$.

To enable Attacker to enter the checking phase, we add the following rules.

III	$X \xrightarrow{c} R_1.D$	$X' \xrightarrow{c} R_1.D$	
		$X' \xrightarrow{c} Z$	
	$R_1 \xrightarrow{\tau} R_1.U_i$	$R_1 \xrightarrow{\tau} R_2$	for all $i \in \{1, 2, \ldots, n\}$
	$R_2 \xrightarrow{\tau} R_2.L_s$	$R_2 \xrightarrow{\tau} Z$	for all $s \in \Sigma$

Having a pair $(X.\alpha, X'.\alpha)$, Attacker can thus also choose to play a **c-action** (instead of an **a-action)**; in this case he is obviously forced (DC) to play $X'.\alpha \xrightarrow{c} Z.\alpha$. Defender can respond by $X.\alpha \overset{c}{\Longrightarrow} Z.\gamma.D.\alpha$ for some $\gamma \in \{L_s \mid s \in \Sigma\}^* \cdot \{U_1, U_2, \ldots, U_n\}^*$ where L_s and U_i are new process constants (we recall that Σ is the alphabet of the instance $(A,\ B)$). In the whole PA system Δ, there will be only one rule with the action d, namely $Z \xrightarrow{d} Z$ (in group V). By inspecting the rules it is easy to verify that if Defender chooses not to finish his move by using the rule $R_2 \xrightarrow{\tau} Z$, Attacker can play $Z \xrightarrow{d} Z$ in the next round and thus, in fact, force reaching a pair $(Z.\gamma.D.\alpha,\ Z.\alpha)$.

We now want to arrange that the above mentioned Defender's response $(X.\alpha \overset{c}{\Longrightarrow} Z.\gamma.D.\alpha)$ can be successful if and only if $\alpha = V_{i_m}.V_{i_{m-1}}.\ldots.V_{i_1}$ represents a partial solution; and in this case the response must be such that $\gamma = L_{a_\ell}.L_{a_{\ell-1}}.\ldots.L_{a_1}.U_{i_m}.U_{i_{m-1}}.\ldots.U_{i_1}$ where

$$u_{i_1}u_{i_2}\ldots u_{i_m}a_1a_2\ldots a_\ell = v_{i_1}v_{i_2}\ldots v_{i_m}. \tag{2}$$

In order to achieve that, we define the set $\mathcal{T} \overset{\text{def}}{=} \{T_w \mid w \text{ is a suffix of some } (u_i)^R \text{ or } (v_i)^R\}$ of new process constants (where $(.)^R$ denotes the reversal operation), and we add the following rules.

IV	$U_k \xrightarrow{\iota_k} \epsilon$	$V_k \xrightarrow{\iota_k} \epsilon$	for each $k \in \{1, 2, \ldots, n\}$
	$U_k \xrightarrow{\tau} T_{(u_k)^R}$	$V_k \xrightarrow{\tau} T_{(v_k)^R}$	for each $k \in \{1, 2, \ldots, n\}$
	$T_{sw} \xrightarrow{s} T_w$	$T_{sw} \xrightarrow{\tau} T_w$	for $T_{sw} \in \mathcal{T}$ and $s \in \Sigma$
		$T_\epsilon \xrightarrow{\tau} \epsilon$	
	$L_s \xrightarrow{s} \epsilon$	$L_s \xrightarrow{\tau} \epsilon$	for all $s \in \Sigma$

We can easily verify that a necessary condition for the processes $L_{a_\ell}.L_{a_{\ell-1}}.\dots.L_{a_1}.U_{i_m}.U_{i_{m-1}}.\dots.U_{i_1}$ and $V_{j_{m'}}.V_{j_{m'-1}}.\dots.V_{j_1}$ to be weakly bisimilar is that $m = m', i_1 = j_1, i_2 = j_2, \dots, i_m = j_m$, and (2) holds. But due to the possible mixing of 'letter-actions' and 'index-actions', the condition is not sufficient. That is why the above processes are preceded by Z in our bisimulation game. If Z can be somehow used to implement a 'switch' for Attacker by which he binds himself to checking either only the index-actions or only the letter-actions then our goal is reached.

We first note that the outcomes of such switching can be modeled by composing in parallel either a process constant C_1 (which masks all letter-actions) or C_2 (which masks all index-actions). So we add the rules for C_1, C_2, and also all the rules for Z (whose meaning will become clear later).

V	$C_1 \xrightarrow{s} C_1$	for each $s \in \Sigma$
	$C_2 \xrightarrow{\iota_k} C_2$	for each $k \in \{1, 2, \dots, n\}$
	$Z \xrightarrow{z} \epsilon$	
	$Z \xrightarrow{\tau} D$	
	$Z \xrightarrow{d} Z$	

The following propositions are now easy to verify.

PROPOSITION 4 *It holds that* $Z.L_{a_\ell}.L_{a_{\ell-1}} \dots L_{a_1}.U_{i_m}.U_{i_{m-1}} \dots U_{i_1} \parallel C_1 \approx Z.V_{j_{m'}}.V_{j_{m'-1}} \dots V_{j_1} \parallel C_1$ *if and only if* $m = m'$ *and* $i_k = j_k$ *for all* k, $1 \le k \le m$.

PROPOSITION 5 *It holds that* $Z.L_{a_\ell}.L_{a_{\ell-1}} \dots L_{a_1}.U_{i_m}.U_{i_{m-1}} \dots U_{i_1} \parallel C_2 \approx Z.V_{j_{m'}}.V_{j_{m'-1}} \dots V_{j_1} \parallel C_2$ *if and only if* $u_{i_1}u_{i_2} \dots u_{i_m}a_1a_2 \dots a_\ell = v_{j_1}v_{j_2} \dots v_{j_{m'}}$.

In order to realize the above discussed 'switch', we add the final group ofrules.

VI	$C \xrightarrow{c_1} C_1$	$C \xrightarrow{c_2} C_2$	$C \xrightarrow{z} C \parallel W$
	$W \xrightarrow{\tau} W.U_k$	$W \xrightarrow{\tau} W.V_k$	for each $k \in \{1, 2, \dots, n\}$
	$W \xrightarrow{\tau} W.L_s$		for all $s \in \Sigma$
	$W \xrightarrow{\tau} \epsilon$		

Now the pair of processes $(X \parallel C, X' \parallel C)$ is the pair (P_1, P_2) we were aiming to construct according to equation (1). This is confirmed by the following two lemmas (the proofs are in the full version of the paper).

LEMMA 6 *If the rPCP instance (A, B) has no solution then* $X \parallel C \not\approx X' \parallel C$.

LEMMA 7 *If the rPCP instance (A, B) has a solution then* $X \parallel C \approx X' \parallel C$.

Now we state the main theorem, which assumes the usual class PA, i.e., without deadlocks.

THEOREM 8 *Weak bisimilarity on PA is Σ^1_1-complete.*

Proof. The membership in Σ^1_1 was already discussed; Σ^1_1-hardness follows from the construction we described and from Lemmas 6 and 7 – on condition that we handle the question of deadlocks. However, there is a straightforward (polynomial-time) reduction from weak bisimilarity of PA with deadlocks to PA without deadlocks (described in [Srba, 2003b]). ■

Combining with the results of [Srba, 2003a] (for PDA and PPDA), we can conclude that weak bisimilarity problems for all PRS-classes on the third level of the hierarchy (and above) are Σ^1_1-complete. Using a similar general strategy, we can show the same results also for weak simulation preorder and equivalence:

THEOREM 9 *Weak simulation preorder/equivalence on PDA, PA and PPDA is Σ^1_1-complete.*

The constructions are more straightforward in this case, where each player is given a fixed system to play in. Here Defender can influence Attacker's moves by threatening to enter a 'universal' process, which enables all actions forever. Problem rPCP is convenient for reductions in the cases of PDA and PA; in the case of PPDA, the recurrent problem for nondeterministic Minsky machines is more suitable. (It asks whether there is an infinite computation which uses a distinguished instruction infinitely often.) A detailed proof is given in the full version of the paper.

A natural conjecture is now that all relations subsuming weak bisimilarity and being subsumed in weak simulation preorder are also Σ^1_1-hard. Such claims, for general relations $R_1 \subseteq R_2$ are usually proven by reduction (from a suitable problem $\mathcal{P}$) constructing two processes P_1 and P_2 such that $(P_1, P_2) \in R_1$ if the answer (for the instance of $\mathcal{P}$ being reduced) is YES and $(P_1, P_2) \notin R_2$ if the answer is NO.

So far we do not see how to modify our constructions to satisfy this. However, in the case of PDA and PPDA, we could in this way derive Σ^1_1-hardness for all relations between weak bisimilarity and branching bisimilarity. A branching bisimulation (as introduced by van Glabbeek and Weijland, see, e.g., [van Glabbeek and Weijland, 1996]) is a symmetric relation R where, for each $(\alpha, \beta) \in R$, each (Attacker's) move $\alpha \xrightarrow{a} \alpha'$ can be matched by a (Defender's) move $\beta \xrightarrow{\tau}^* \beta_1 \xrightarrow{a} \beta_2 \xrightarrow{\tau}^* \beta'$ where we require $(\alpha', \beta') \in R$ and also $(\alpha, \beta_1) \in R$, $(\alpha', \beta_2) \in R$; Defender's move can be empty in the case $a = \tau$ (then $(\alpha', \beta) \in R$).

CLAIM 10 *All relations subsuming branching bisimilarity and being subsumed in weak bisimilarity are Σ^1_1-hard on PDA and PPDA.*

We do not provide a detailed proof since it would require to repeat the constructions used in [Srba, 2003a], with some slight modifications. The point is that the long τ-moves (of Defender) can be made reversible (e.g., for setting a counter value there are τ-actions for both increasing and decreasing). This can be achieved easily in the presence of a finite-control unit (like in case of PDA and PPDA). Such a reversibility is not present in our construction for PA, and it is unclear whether PA can model these features in an alternative way.

4. Other semantic equivalences

A natural question to ask is about the complexity of other well-known semantic equivalences (like those in [van Glabbeek, 2001] or, more relevantly for us, in [van Glabbeek, 1993]). Of particular interest is the question whether some other equivalences are also highly undecidable (i.e., beyond (hyper)arithmetical hierarchy). We provide a few results and notes about this.

For a finite or infinite $w = a_1 a_2 \dots$ we write $\alpha_0 \overset{w}{\Longrightarrow}$ iff there are $\alpha_1, \alpha_2, \dots$ such that $\alpha_i \overset{a_{i+1}}{\Longrightarrow} \alpha_{i+1}$ for all $i = 0, 1, 2, \dots$. The coarsest equivalence among the studied action-based semantic equivalences is the trace equivalence: two processes α and β are *weakly trace equivalent* iff $\forall w \in (Act \smallsetminus \{\tau\})^* : \alpha \overset{w}{\Longrightarrow} \Leftrightarrow \beta \overset{w}{\Longrightarrow}$ (i.e., α and β enable the *same finite* observable traces).

We can immediately see that the problem is at a very low level in the arithmetical hierarchy even for very general classes of labelled transition systems. We call a labelled transition system (LTS) *recursively enumerable* if the set of states S and the set of actions Act are both (represented as) recursively enumerable sets of strings in some finite alphabets and the set $\{(\alpha, a, \beta) \mid \alpha, \beta \in S, a \in Act, \alpha \xrightarrow{a} \beta\}$ is also recursively enumerable. The respective algorithms (Turing machines) can serve as finite descriptions of such an LTS.

We can easily observe that given a recursively enumerable LTS (where Act includes τ), the set $\{(\alpha, w) \mid w \in (Act \smallsetminus \{\tau\})^*, \alpha \overset{w}{\Longrightarrow}\}$ is also recursively enumerable. More generally, the set of all triples (L, α, w), where L is (a description of) a recursively enumerable LTS, α one of its states and w a finite sequence of its (observable) actions such that $\alpha \overset{w}{\Longrightarrow}$ (in L), can be defined by some Σ^0_1-formula $\exists x.\phi(L, \alpha, w, x)$ where ϕ is recursive (with the parameters coded by natural numbers).

PROPOSITION 11 *The set of all triples (L, α, β), where L is (a description of) a recursively enumerable LTS and α, β two weakly trace equivalent states, is in Π^0_2.*

REMARK 12 *In fact, for the classes like PDA, PA and PN the set $\{(L, \alpha, w) \mid \alpha \overset{w}{\Longrightarrow}\}$ is even recursive. For PDA and PA this follows, e.g., from [Büchi, 1964] and [Lugiez and Schnoebelen, 2002] and for PN it can be decided by standard constructions from Petri net theory (reducing to the coverability problem). This means that weak trace equivalence for such classes is in Π^0_1.*

For other equivalences based on trace-like finite behaviours (sometimes called ‘decorated traces’), i.e., failure equivalence, ready equivalence, ready-trace equivalence etc., we can make similar observations. This means that in fact all these (weak) equivalences are very low in the arithmetical hierarchy.

In some sense, this might seem as a surprising fact. In the strong case (without τ-actions), complexity of the equivalence problems is decreasing in the direction: trace – simulation – bisimulation. On the other hand in the weak case the situation now seems to look different. However, the right way for such a comparison is to take also infinite traces (i.e., ω-traces) into account. Then the above complexity-decreasing chain is restored as illustrated below.

REMARK 13 *For image-finite labelled transition systems (like those generated by PRS systems in the strong case), the finite-trace equivalence implies also the ω-trace equivalence. This is, however, not true for non-image-finite systems, which are easily generated by PRS systems in the weak case.*

We shall focus on the classes BPP and BPA. For BPP weak bisimilarity is known to be semidecidable [Esparza, 1997], so it belongs to the class Σ^0_1. In fact, it seems even well possible that the problem is decidable (see [Jančar, 2003] where PSPACE-completeness of strong bisimilarity is established). Simulation preorder/equivalence (as well as trace preorder/equivalence) is undecidable even in the strong case [Hüttel, 1994]. Weak simulation preorder/equivalence is surely in Σ^1_1 (the best estimate we can derive at the moment) while we can prove that weak ω-trace preorder/equivalence is Π^1_1-hard:

THEOREM 14 *Weak ω-trace preorder/equivalence on BPP is Π^1_1-hard.*

Given a nondeterministic Minsky machine, the nonexistence of an infinite computation using instruction 1 infinitely often can be reduced to the weak ω-trace preorder (equivalence) problem. In order to prove this we modify a known construction showing undecidability of trace preorder in the strong case (which can be found in [Hirshfeld, 1994]). A more detailed sketch of the proof is in the full version of the paper.

For BPA, the situation is roughly similar though a bit more unclear. Both weak bisimilarity and weak similarity are surely in Σ^1_1 but otherwise we only know that weak bisimilarity is EXPTIME-hard [Mayr, 2003] and weak similarity undecidable; the latter follows from undecidability of (even) strong similarity [Groote and Hüttel, 1994]. There are some reasons to conjecture that weak bisimilarity of BPA might be decidable. The (obvious) membership in Σ^1_1 thus seems to be a very rough upper bound, and one might start to try to strenghten this by showing that the problem is in the hyperarithmetical hierarchy, i.e., in the intersection of Σ^1_1 and Π^1_1. Nevertheless, it seems that a deeper insight would be needed even for this less ambitious goal.

The undecidability of strong trace equivalence for BPA follows easily from classical results for context-free langauges. Moreover, similarly as in the case of BPP, we can show:

THEOREM 15 *Weak ω-trace preorder/equivalence on BPA is Π^1_1-hard.*

The theorem holds even when one BPA-process is a fixed finite-state process. The proof uses the recurrent problem for nondeterministic Turing machines and builds on the classical context-free grammar generating all words which do not correspond to correct computations of a Turing machine (where all even configurations are written in the reverse order). More details are in the full version of the paper. We also add an analogy to Proposition 11:

PROPOSITION 16 *The set of all triples (L, α, β), where L is (a description of) a recursively enumerable LTS and α, β two weakly ω-trace equivalent states, is in Π^1_2.*

5. Regularity is in the hyperarithmetical hierarchy

Here we look at some more specialized problems, namely the question of equivalence (of a general process) with a given finite-state process, and the question of regularity, which asks whether a given (general) process is equivalent (weakly bisimilar in our case) to an (unspecified) finite-state process.

Denoting the collection of all sets which are recursively enumerable in *TA* (truth in mathematics) by $\Sigma^0_{\omega+1}$, we can show:

PROPOSITION 17 *The problem of weak regularity of recursively enumerable labelled transition systems is in $\Sigma^0_{\omega+1}$.*

Though the stated result is not too practical, it still separates weak bisimilarity checking from weak regularity checking for the classes like PDA, PA and PPDA (because $\Sigma^0_{\omega+1}$ is a proper subclass of $\Sigma^1_1 \cap \Pi^1_1$). Recalling the general experience that natural problems (in computer science) are either at low levels of the arithmetical hierarchy or at low levels of the analytical hierarchy, we have at least some indication in what direction the results for regularity can be possibly strengthened.

References

[Büchi, 1964] Büchi, J.R. (1964). Regular canonical systems. *Arch. Math. Logik u. Grundlagenforschung,* 6:91–111.

[Burkart et al., 2001] Burkart, O., Caucal, D., Moller, F., and Steffen, B. (2001). Verification on infinite structures. In Bergstra, J., Ponse, A., and Smolka, S., editors, *Handbook of Process Algebra,* chapter 9, pages 545–623. Elsevier Science.

[Esparza, 1997] Esparza, J. (1997). Petri nets, commutative context-free grammars, and basic parallel processes. *Fundamenta Informaticae,* 31:13–26.

[Groote and Hüttel, 1994] Groote, J.F. and Hüttel, H. (1994). Undecidable equivalences for basic process algebra. *Information and Computation,* 115(2):353–371.

[Harel, 1986] Harel, D. (1986). Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness. *Journal of the ACM (JACM),* 33(1):224–248.

[Hirshfeld, 1994] Hirshfeld, Y. (1994). Deciding equivalences in simple process algebras. Tech. report ECS-LFCS-94-294, Dept. of Computer Science, University of Edinburgh.

[Hüttel, 1994] Hüttel, H. (1994). Undecidable equivalences for basic parallel processes. In *Proc. of TACS'94,* volume 789 of *LNCS,* pages 454–464. Springer-Verlag.

[Jančar, 2003] Jančar, P. (2003). Strong bisimilarity on basic parallel processes is PSPACE-complete. In *Proc. of LICS'03,* pages 218–227. IEEE Computer Society Press.

[Jančar et al., 2003] Jančar, P., Kučera, A., and Moller, F. (2003). Deciding bisimilarity between bpa and bpp processes. In *Proc. of CONCUR'03,* volume 2761 of *LNCS,* pages 159–173. Springer-Verlag.

[Jančar and Srba, 2004] Jančar, P. and Srba, J. (2004). Highly Undecidable questions for process algebras. Tech. Report RS-04-8, BRICS Research Series.

[Kučera and Jančar, 2002] Kučera, A. and Jančar, P. (2002). Equivalence-checking with infinite-state systems: Techniques and results. In *Proc. of SOFSEM'02,* volume 2540 of *LNCS,* pages 41–73. Springer-Verlag.

[Lugiez and Schnoebelen, 2002] Lugiez, D. and Schnoebelen, Ph. (2002). The regular view-point on pa-processes. *Theoretical Computer Science,* 274(1–2): 89–115.

[Mayr, 2000] Mayr, R. (2000). Process rewrite systems. *Information and Computation,* 156(1):264–286.

[Mayr, 2003] Mayr, R. (2003). Weak bisimilarity and regularity of BPA is EXPTIME-hard. In *Proc. of EXPRESS'03,* pages 160–143.

[Rogers, 1967] Rogers, H. (1967). *Theory of Recursive Functions and Effective Computability.* McGraw-Hill.

[Srba, 2002] Srba, J. (2002). Roadmap of infinite results. *Bulletin of the European Association for Theoretical Computer Science (Columns: Concurrency),* 78:163-175. Updated online version: http://www.brics.dk/~srba/roadmap.

[Srba, 2003a] Srba, J. (2003a). Completeness results for undecidable bisimilarity problems. In *Proc. of INFINITY'03,* pages 9–22.

[Srba, 2003b] Srba, J. (2003b). Undecidability of weak bisimilarity for PA-processes. In *Proc. of DLT'02,* volume 2450 of *LNCS,* pages 197-208. Springer-Verlag.

[van Glabbeek, 1993] van Glabbeek, R.J. (1993). The linear time – branching time spectrum II (the semantics of sequential systems with silent moves). In *Proc. of CONCUR '93,* volume 715 of *LNCS,* pages 66–81. Springer-Verlag.

[van Glabbeek, 2001] van Glabbeek, R.J. (2001). The linear time - branching time spectrum I: The semantics of concrete, sequential processes. In *Handbook of Process Algebra,* chapter 1, pages 3–99. Elsevier Science.

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