Undecidability Results for Bisimilarity on Prefix Rewrite Systems

Petr Jančar[1,⋆] and Jiří Srba[2,⋆⋆]

[1] Center of Applied Cybernetics, Department of Computer Science, Technical University of Ostrava, Czech Republic
[2] **BRICS**[⋆⋆⋆] Department of Computer Science, Aalborg University, Denmark

Abstract. We answer an open question related to bisimilarity checking on labelled transition systems generated by prefix rewrite rules on words. Stirling (1996, 1998) proved the decidability of bisimilarity for normed pushdown processes. This result was substantially extended by Sénizergues (1998, 2005) who showed the decidability for regular (or equational) graphs of finite out-degree (which include unnormed pushdown processes). The question of decidability of bisimilarity for a more general class of so called Type -1 systems (generated by prefix rewrite rules of the form $R \xrightarrow{a} w$ where R is a regular language) was left open; this was repeatedly indicated by both Stirling and Sénizergues. Here we answer the question negatively, i.e., we show undecidability of bisimilarity on Type -1 systems, even in the normed case. We complete the picture by considering classes of systems that use rewrite rules of the form $w \xrightarrow{a} R$ and $R_1 \xrightarrow{a} R_2$ and show when they yield low undecidability (Π^0_1-completeness) and when high undecidability (Σ^1_1-completeness), all with and without the assumption of normedness.

1 Introduction

Bisimilarity [17], or bisimulation equivalence, has been recognized as a fundamental notion in concurrency theory, in verification of behaviour of (reactive) systems, and in other areas. This has initiated several research directions, one of them exploring the decidability and complexity questions for bisimilarity. The obtained results differ from those known in the case of classical language equivalence; we can refer to surveys like [3, 25].

Bisimilarity is defined on labelled transition systems which can be viewed as (possibly infinite) edge-labelled directed graphs. Particular classes of graphs which have been in the focus of researchers are defined by prefix rewrite systems. We refer to a hierarchy defined by Stirling [27], which is interconnected with the work of Caucal [7, 5, 9]. We focus on (subclasses of) so called Type -2 systems;

⋆ The author is supported by the Czech Ministry of Education, Grant No. 1M0567.
⋆⋆ The author is supported in part by the research center ITI, project No. 1M0545.
⋆⋆⋆ **B**asic **R**esearch in **C**omputer **S**cience, Centre of the Danish National Research Foundation.

such a system is given by a finite set of rewrite rules of the form $R_1 \xrightarrow{a} R_2$ where a is an action name (i.e. edge-label) and R_1, R_2 are regular languages over a finite alphabet. Processes (or states in the respective labelled transition system) are finite sequences of alphabet symbols. A rule $R_1 \xrightarrow{a} R_2$ stands for a potentially infinite set of rules $\{w \xrightarrow{a} w' \mid w \in R_1,\ w' \in R_2\}$, where $w \xrightarrow{a} w'$ can be applied to a process v iff w is a prefix of v (which is then replaced by w'). A process v is called *normed* iff each (finite) path from v can be prolonged to reach the empty process (word) ε.

Important subclasses of Type -2 graphs are called Type -1 and Type 0, where rules are of the form $R \xrightarrow{a} w$ and $w \xrightarrow{a} w'$, respectively. The class of Type 0 systems is isomorphic to the class of pushdown graphs [7], also called Type $1\frac{1}{2}$ by Stirling. By imposing further restrictions we get Type 2 graphs which correspond to BPA (Basic Process Algebra) and Type 3 graphs which coincide with finite-state transition systems.

Several nontrivial results achieved for pushdown (Type 0) processes turned out to be extendable to a superclass of Type -2 systems, namely the class of prefix-recognizable graphs, also called REC_{RAT} in [9]. This includes e.g. decidability of monadic second order logic [9] and the existence of uniform winning strategies for parity games [6]. The decidability questions for bisimilarity are, however, more intricate.

In 1995 Caucal [8] formulated three open questions about decidability of bisimilarity for (1) pushdown graphs, (2) regular graphs of finite out-degree, and (3) regular graphs. A bit later, Stirling showed the decidability of bisimilarity for restricted, namely normed, pushdown processes [27]. He stated the following two questions:

– Is bisimilarity decidable for Type -1 systems?
– Is bisimilarity decidable for Type -2 systems?

The initial hope was that the technique for normed pushdown processes might be extendable to these classes, in the normed case at least. Caucal's problem (1) was answered positively (in the full, i.e., unrestricted case) by Sénizergues [19] (who extended the technique used in his famous result for deterministic pushdown automata [20]). Stirling later presented a shorter proof in [29]. The result of Sénizergues also gives a positive answer to Caucal's problem (2); a complete journal version appeared recently in [21]. Due to a terminology mismatch, it was incorrectly indicated in [19] that the positive decidability result applies to Type -1 systems as well; this was later corrected in [21] by noting that it is valid (just) for a significant subclass of Type -1 graphs. More precisely, in [21] it was shown that regular graphs of finite out-degree (for which the decidability result was obtained) coincide up to isomorphism with collapsed graphs of pushdown automata with deterministic and popping only ε-transitions, and this is not the full class of Type -1 systems (where nondeterministic popping is allowed).

Remark. In the full paper (available as a BRICS technical report) we show that the class of regular graphs of finite out-degree can be characterized by Type -1 rules $R \xrightarrow{a} w$ with the restriction that R is a *prefix-free* (regular) language.

Thus Stirling's question about decidability of bisimilarity for Type -1 systems, also in the normed case, remained open (as several times explicitly indicated in the literature, most recently in [21]).

Our contribution. The main contribution of the present paper is the result showing the undecidability of bisimilarity on Type -1 systems, even in the normed case. Hence we have answered negatively the two open problems formulated by Stirling. Besides this, we have performed a more detailed analysis of the undecidability on related process classes.

We have also slightly extended the considered hierarchy. We view Stirling's Type -1 rules $R \xrightarrow{a} w$ as Type -1a, and we introduce a complementary class called Type -1b to denote rules of the type $w \xrightarrow{a} R$. Such a comparative study provides a deeper insight into prefix rewriting systems by classifying the respective undecidability degrees.

Remark. In the full paper we show that the classes -1a and -1b are incomparable w.r.t. bisimilarity and strictly above Type 0 and below Type -2 systems.

Let us recall a general experience that the 'natural' undecidable problems we encounter in computer science are either 'lowly' undecidable, i.e., at the first levels of arithmetical hierarchy — typically equivalent to the halting problem or its complement (Σ^0_1-complete or Π^0_1-complete), or 'highly' undecidable — typically complete for the first levels of analytical hierarchy (Σ^1_1-complete or Π^1_1-complete).

We demonstrate that bisimilarity is undecidable for normed Type -1a and Type -1b processes. More precisely, we establish Π^0_1-completeness of bisimilarity, both in the normed case and the unrestricted case, for Type -1a systems and in the normed case also for Type -1b systems. High undecidability, in fact Σ^1_1-completeness, is shown in the unrestricted case for Type -1b systems, and in the normed case and the unrestricted case for Type -2 systems. These results are completed by an observation that normedness of a given (Type -2) process is decidable.

Last but not least, our results are achieved in a uniform way, using reductions from two variants of Post's Correspondence Problem (one Π^0_1-complete, the other Σ^1_1-complete). An important technical ingredient of our reductions is the so called Defender's Choice technique (related to bisimulation games), which we most recently used in [14].

Remark. The techniques from the undecidability proofs of weak bisimilarity for pushdown automata [23, 24] can be used to prove undecidability (Σ^1_1-completeness) of strong bisimilarity also for Type -2 systems (no conflict between visible and ε-transitions means that ε-collapsed PDA graphs coincide with Type -2 graphs [29]; even though the pushdown rules in [23, 24] do not avoid the mentioned conflict, they can be modified to suppress clashes between visible and ε-moves). Nevertheless, the constructions from [23, 24] use pushdown processes the collapsed graphs of which have infinite out-degree and it is not straightforward to adapt the reductions to work also for Type -1a systems.

2 Preliminaries

A *labelled transition system* (LTS) is a triple $(S, \mathcal{A}ct, \longrightarrow)$ where S is a set of *states* (or *processes*), $\mathcal{A}ct$ is a set of *labels* (or *actions*), and $\longrightarrow \subseteq S \times \mathcal{A}ct \times S$ is a *transition relation*; for each $a \in \mathcal{A}ct$, we view $\xrightarrow{a}$ as a binary relation on S where $\alpha \xrightarrow{a} \beta$ iff $(\alpha, a, \beta) \in \longrightarrow$. The notation can be naturally extended to $\alpha \xrightarrow{s} \beta$ for finite sequences of actions s; and by $\alpha \longrightarrow^* \beta$ we mean that there is s such that $\alpha \xrightarrow{s} \beta$.

Given $(S, \mathcal{A}ct, \longrightarrow)$, a binary relation $R \subseteq S \times S$ is a *simulation* iff for each $(\alpha, \beta) \in R$, $a \in \mathcal{A}ct$, and α' such that $\alpha \xrightarrow{a} \alpha'$ there is β' such that $\beta \xrightarrow{a} \beta'$ and $(\alpha', \beta') \in R$. A *bisimulation* is a simulation which is symmetric. Processes α and β are *bisimilar*, denoted $\alpha \sim \beta$, if there is a bisimulation containing (α, β). We note that bisimilarity is an equivalence relation.

We shall use a standard game-theoretic characterization of bisimilarity [31, 26]. A *bisimulation game* on a pair of processes (α_1, α_2) is a two-player game between *Attacker* and *Defender*. The game is played in *rounds*. In each round (consisting of two moves) the players change the *current pair of states* (β_1, β_2) (initially $\beta_1 = \alpha_1$ and $\beta_2 = \alpha_2$) according to the following rule:

1. Attacker chooses $i \in \{1, 2\}$, $a \in \mathcal{A}ct$ and $\beta_i' \in S$ such that $\beta_i \xrightarrow{a} \beta_i'$. He thus creates an *intermediate pair* which is (β_1', β_2) in the case $i = 1$, and (β_1, β_2') in the case $i = 2$.
2. Defender responds by choosing $\beta_{3-i}' \in S$ such that $\beta_{3-i} \xrightarrow{a} \beta_{3-i}'$.
3. The pair (β_1', β_2') becomes the (new) current pair of states.

Any *play* (of the bisimulation game) thus corresponds to a sequence of pairs of states such that Attacker is making a move from every odd position and Defender from every even one (under the same action as in the previous Attacker's move).

A play (and the corresponding sequence) is finite iff one of the players gets stuck (cannot make a move); the player who got stuck lost the play and the other player is the winner. (A play finishing in an intermediate pair on an even position is winning for Attacker and a play finishing on an odd position is winning for Defender.) If the play is infinite then Defender is the winner. We use the following standard fact.

Proposition 1. *It holds that $\alpha_1 \sim \alpha_2$ iff Defender has a winning strategy in the bisimulation game starting with the pair (α_1, α_2), and $\alpha_1 \not\sim \alpha_2$ iff Attacker has a winning strategy.*

We shall now demonstrate a simple idea to establish semidecidability of nonbisimilarity for a particular class of LTSs; the idea slightly extends the standard argument used for image-finite systems [12]. (For example, for normed systems of Type -1b considered in the proof of Theorem 2, we are able to argue about the semidecidability of nonbisimilarity even though the systems are not necessarily image-finite.)

We say that a labelled transition system $(S, \mathcal{A}ct, \longrightarrow)$ is *effective* iff both S and $\mathcal{A}ct$ are decidable subsets of the set of all finite strings in a given finite alphabet and the relation $\longrightarrow$ is decidable.

An LTS $(S, \mathcal{A}ct, \longrightarrow)$ is called *finitely over-approximable* (w.r.t. bisimilarity) iff for any $\alpha, \beta \in S$, $a \in \mathcal{A}ct$, a finite set $E_{(\alpha,\beta,a)} \subseteq S$ can be effectively constructed such that whenever $\beta \xrightarrow{a} \beta'$ and $\beta' \sim \alpha$ then $\beta' \in E_{(\alpha,\beta,a)}$. Thus the (finite) set $E_{(\alpha,\beta,a)}$ over-approximates the set of a-successors of β which are bisimilar with α.

Proposition 2. *The problem of nonbisimilarity on effective and finitely over-approximable labelled transition systems is semidecidable, i.e., the bisimilarity problem is in Π^0_1.*

Proof. (Sketch) It is sufficient to provide a procedure which halts, for a given pair (α_1, α_2) of a given effective and finitely over-approximable system, iff there is a winning strategy for Attacker. Such a strategy can be naturally viewed as a tree where each vertex is labelled by a pair of processes and each edge, labelled by an action, corresponds to a move (of Attacker or Defender). While each vertex on an odd level has just one outgoing edge (Attacker's moves are fixed by the strategy), the vertices on even levels (corresponding to the 'intermediate' pairs) can have more successors (corresponding to Defender's choices). Each branch of the tree corresponds to a possible play when Attacker plays according to the assumed winning strategy; each branch is thus finite (finishing by Defender's getting stuck). Due to the assumed finite over-approximability, it is always sufficient to consider only finitely many possibilities for Defender's moves; the respective strategy-tree is then finitely branching and thus finite. So it is sufficient to systematically generate all finite trees and check for each of them if it happens to represent a winning strategy of Attacker; the checking can be done algorithmically due to our effectiveness assumptions. □

2.1 A Hierarchy of Regular Prefix Rewriting

We are interested in special labelled transition systems, namely those generated by systems of prefix rewrite rules. We now provide the relevant definitions.

The most general systems we consider are *Type -2 systems*. Such a system $\mathcal{S}$ can be viewed as a triple $\mathcal{S} = (\Gamma, \mathcal{A}ct, \Delta)$ where Γ is a finite set of *process symbols*, $\mathcal{A}ct$ is a finite set of *actions*, and Δ is a finite set of *rewrite rules*. Each rewrite rule is of the form $R_1 \xrightarrow{a} R_2$ where $a \in \mathcal{A}ct$ and R_1 and R_2 are regular languages over Γ such that $\varepsilon \notin R_1$; for concreteness, we can assume that R_1, R_2 are given by regular expressions.

We view the system $\mathcal{S}$ as generating a certain LTS $(\Gamma^*, \mathcal{A}ct, \longrightarrow)$. A process (or a state in the respective LTS) is any finite sequence of process symbols, i.e., any element of Γ^*; we shall use $u, v, w, \ldots$ for denoting elements of Γ^*, and ε for denoting the empty sequence. The transition relation $\longrightarrow$ (i.e., the collection of relations $\xrightarrow{a}$) is defined by the following derivation rule:

$$\frac{(R_1 \xrightarrow{a} R_2) \in \Delta, \quad w \in R_1, \quad w' \in R_2, \quad u \in \Gamma^*}{wu \xrightarrow{a} w'u}$$

Thus any rule $(R_1 \xrightarrow{a} R_2) \in \Delta$ represents possibly infinitely many rewrite rules $w \xrightarrow{a} w'$ where $w \in R_1$ and $w' \in R_2$.

We shall also need the notion of normedness. We say that a process $w \in \Gamma^*$ is *normed* if for any w' such that $w \longrightarrow^* w'$ we have $w' \longrightarrow^* \varepsilon$. In other words, a process w is normed iff any path from w in the respective LTS can be prolonged to finish in ε. A *norm* of a normed process w, denoted by $norm(w)$, is the length of the shortest action sequence s such that $w \xrightarrow{s} \varepsilon$.

Proposition 3. *If two normed processes are bisimilar then they have the same norm.*

Proof. Assume normed u, v with $norm(u) < norm(v)$. For the shortest sequence s such that $u \xrightarrow{s} \varepsilon$ we have: if $v \xrightarrow{s} v'$ then v' is normed and $v' \neq \varepsilon$ (thus v' can perform an action). This implies that u and v are not bisimilar. $\square$

Proposition 4. *There is an algorithm which given a Type -2 process v decides whether v is normed, and computes its norm in the positive case.*

Proof. (Sketch) We can base the algorithm on the well-known fact regarding (classical) pushdown automata: given a pushdown automaton and an initial state $\times$ stack configuration, the set of all state $\times$ stack configurations reachable from the initial one is regular, and its representation can be effectively constructed [2, 10].

We observe (see also [29]) that applying a rule $R_1 \xrightarrow{a} R_2$ to v, i.e., replacing a prefix $w \in R_1$ of v by $w' \in R_2$, can be implemented by a series of ε-moves of a pushdown automaton (whose control unit includes finite automata for R_1, R_2).

In this way we can easily derive that, given a Type -2 system and a process v, the set $post^*(v)$ — consisting of all processes reachable from v — is an effectively constructible regular set. Similarly, the set $pre^*(\varepsilon)$ — consisting of all processes from which ε is reachable — is an effectively constructible regular set. Checking normedness of v now amounts to verifying whether $post^*(v) \subseteq pre^*(\varepsilon)$.

Computing $norm(v)$ for a normed v can be accomplished by stepwise constructing $pre(\varepsilon)$, $pre(\,pre(\varepsilon)\,)$, $pre(\,pre(\,pre(\varepsilon)\,)\,)$, $\ldots$ until v is included; here $pre(R)$ denotes the set of processes from which some $u \in R$ is reachable in one step. Such a computation can be again easily reduced to computing the sets of reachable configurations of pushdown automata. $\square$

The other systems we consider arise from the above defined Type -2 systems by restricting the form of rewrite rules. We use the terminology introduced by Stirling (see, e.g., [30]). In the following table, R_1, R_2 and R stand for regular sets over Γ; w, w' stand for elements of Γ^* (the respective regular languages are thus singletons); and X, Y, p, q stand for elements of Γ. We have added Type -1b to Stirling's table; his Type -1 coincides with our Type -1a. In the full paper it is shown that Type -1a and -1b classes are incomparable w.r.t. bisimilarity and strictly above Type 0 and below Type -2 systems.

Type	Form of Rewrite Rules
Type -2	$R_1 \xrightarrow{a} R_2$
Type -1a/-1b	$R \xrightarrow{a} w \ / \ w \xrightarrow{a} R$
Type 0	$w \xrightarrow{a} w'$
Type $1\frac{1}{2}$	$pX \xrightarrow{a} qw$
Type 2	$X \xrightarrow{a} w$
Type 3	$X \xrightarrow{a} Y, \ X \xrightarrow{a} \varepsilon$

$$\begin{tikzcd}
& \text{Type -2} \arrow[dl, dash] \arrow[dr, dash] & \\
\text{Type -1a} \arrow[dr, dash] & & \text{Type -1b} \arrow[dl, dash] \\
& \text{Type 0} = \text{Type } 1\frac{1}{2} \arrow[d, dash] & \\
& \text{Type 2} \arrow[d, dash] & \\
& \text{Type 3} &
\end{tikzcd}$$

We can note that Type $1\frac{1}{2}$ rules are classical pushdown rules (p, q are 'highlighted' as finite control states and are disjoint with the stack alphabet); this class was shown to coincide up to isomorphism with Type 0 systems [7]. Type 2 systems are also called BPA (Basic Process Algebra) systems, and Type 3 systems correspond to finite labelled transition systems.

2.2 Versions of Post's Correspondence Problem

Here we recall the versions of Post's Correspondence Problem (PCP) which will be used in the later reductions.

A *PCP-instance* is a nonempty sequence $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)$ of pairs of nonempty words in the alphabet $\{A, B\}$ such that $|u_i| \leq |v_i|$ for all i, $1 \leq i \leq n$ (where $|u|$ denotes the length of u).

An *infinite initial solution* of the given instance is an infinite sequence of indices $i_1, i_2, i_3, \ldots$ from the set $\{1, 2, \ldots, n\}$ such that $i_1{=}1$ and 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. An *infinite recurrent solution* is an infinite initial solution in which the index 1 appears infinitely often.

By *inf-PCP* we denote the problem to decide whether a given PCP instance has an infinite initial solution; *rec-PCP* denotes the problem to decide whether a given PCP instance has an infinite recurrent solution.

Proposition 5. *Problems inf-PCP and rec-PCP are Π^0_1-complete and Σ^1_1-complete, respectively.*

These facts can be easily established from well-known results but we can refer, e.g., to [18] for the (low) undecidability and to [11] for the high undecidability. Our (additional) requirement $|u_i| \leq |v_i|$ is non-standard but it can be easily checked to be harmless; we use it for its technical convenience. (The harmlessness of the extra requirement follows directly from the standard textbook reduction from Turing machines to PCP. The reduction produces an instance of PCP which satisfies our requirement, except for the last category of pairs of strings that are used to equalize the lengths of the two generated words in case that an accepting configuration is reached. As our question is about the existence of an infinite computation, we can safely omit the pairs from the last category.)

It is also useful to note the following obvious fact.

Proposition 6. *Given a PCP-instance $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)$ and a sequence $i_1, i_2, i_3, \ldots$ of indices where $i_1 = 1$, the following three conditions are equivalent:*

- *$i_1, i_2, i_3, \ldots$ is an infinite initial solution,*
- *for each $\ell = 1, 2, 3, \ldots$, the word $u_{i_1} u_{i_2} \ldots u_{i_\ell}$ is a prefix of $v_{i_1} v_{i_2} \ldots v_{i_\ell}$,*
- *for infinitely many $\ell \geq 1$, the word $u_{i_1} u_{i_2} \ldots u_{i_\ell}$ is a prefix of $v_{i_1} v_{i_2} \ldots v_{i_\ell}$.*

3 Proof Strategy

A crucial point in proving completeness results for bisimilarity on prefix rewriting systems are the hardness reductions, from inf-PCP or rec-PCP to the respective bisimilarity problems. Here we describe the general idea of these reductions, which will be implemented later by suitable sets of rewrite rules.

In each particular case studied in this paper, we assume a fixed PCP-instance $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)$ (over the alphabet $\{A, B\}$) and show how to construct an appropriate rewrite system $(\Gamma, \mathcal{A}ct, \Delta)$. The set of process symbols Γ will always contain the alphabet symbols A, B, 'index-symbols' $I_1, I_2, \ldots, I_n$, and auxiliary symbols $X, X', \bot$ and others.

Our constructions will guarantee that $XI_1\bot \sim X'I_1\bot$ (and hence that Defender has a winning strategy from the pair $(XI_1\bot\,,\ X'I_1\bot)$) if and only if there is an infinite initial solution or an infinite recurrent solution — depending on the source problem (inf-PCP or rec-PCP).

Our intention is that each play starting from $(\,XI_1\bot\,,\ X'I_1\bot\,)$ will begin with a *generating phase*: this phase produces pairs (of current states) of the form

$$(\,XI_{i_\ell}I_{i_{\ell-1}} \ldots I_{i_1}\bot\,,\ X'I_{i_\ell}I_{i_{\ell-1}} \ldots I_{i_1}\bot\,) \quad \text{(where } i_1{=}1\text{)} \qquad (1)$$

where the players are stepwise building longer and longer prefixes of an infinite sequence $i_1, i_2, i_3, \ldots$; this means that the pair (1) can only be 'prolonged', i.e., followed by the pair

$$(XI_{i_m}I_{i_{m-1}} \ldots I_{i_{\ell+1}}I_{i_\ell}I_{i_{\ell-1}} \ldots I_{i_1}\bot\,,\ X'I_{i_m}I_{i_{m-1}} \ldots I_{i_{\ell+1}}I_{i_\ell}I_{i_{\ell-1}} \ldots I_{i_1}\bot)$$

for $m > \ell$. Moreover, in the case of rec-PCP we will guarantee that $i_m = 1$, which ensures that the first index has to be repeatedly inserted into the states.

Remark. Due to the nature of prefix rewrite rules, we represent generating of a sequence $I_{i_1}, I_{i_2}, I_{i_3}, \ldots$ by using the 'right-to-left' direction.

A subtle point is that the elements of the sequence $i_1, i_2, i_3, \ldots$ arising during the generating phase must be freely chosen by Defender. We implement this by a variant of so-called Defender's Choice technique (which we used, e.g., in [14]).

The generating phase can go on arbitrarily long, maybe forever (in which case Defender wins). Attacker will always have the possibility to finish this phase by *switching* to a *verification phase*; our rules will guarantee that Attacker can thus force his win from the current pair (1) if and only if $u_{i_1} u_{i_2} \ldots u_{i_\ell}$ is not a prefix of $v_{i_1} v_{i_2} \ldots v_{i_\ell}$.

The correctness of the described general strategy follows from Proposition 6.

4 (Low) Undecidability Results

In this section we show that bisimilarity is Π^0_1-complete for both normed and unrestricted Type -1a systems, and for normed Type -1b systems; this in particular entails the undecidability for (normed) Type -1a systems (i.e., Stirling's Type -1 systems).

As explained in Section 3, in what follows we assume a fixed PCP-instance $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)$ (over the alphabet $\{A, B\}$); the instance is now viewed as an input of inf-PCP.

4.1 Π^0_1-Completeness of Normed and Unrestricted Type-1a Systems

We first provide the rules and then explain how they implement the above described strategy. For the generating phase we add auxiliary process symbols Y and $Y_1, Y_2, \ldots, Y_n$, actions $1, 2, \ldots, n$ and c, and

(G1) rules:
$$\begin{array}{llll} X \xrightarrow{c} Y & & & \\ X \xrightarrow{c} Y_i & X' \xrightarrow{c} Y_i & & \text{for all } i \in \{1, 2, \ldots, n\} \\ Y \xrightarrow{i} XI_i & Y_i \xrightarrow{i} X'I_i & & \text{for all } i \in \{1, 2, \ldots, n\} \\ & Y_i \xrightarrow{j} XI_j & & \text{for all } i, j \in \{1, 2, \ldots, n\},\ i \neq j. \end{array}$$

For switching we add the symbols C, C', an action d, and

(S1) rules:
$$\begin{array}{lll} X \xrightarrow{d} C & & \\ X(I^*)I_i \xrightarrow{d} C'w & X'(I^*)I_i \xrightarrow{d} C'w & \text{for all } i \in \{1, 2, \ldots, n\} \\ & & \text{and all suffixes } w \text{ of } v_i^R. \end{array}$$

Notation. I^* stands for the regular expression $(I_1 + I_2 + \cdots + I_n)^*$; this uses the possibility allowed by Type -1a rules. For a word u, by u^R we denote the reverse image of u.

For the verification phase we add actions a, b, e, and the following rules in which we use a further piece of notation.
Notation. By *head*(w) we denote the first symbol of w; *tail*(w) is the rest of w. By $h(w)$ (head-action) we mean a when *head*$(w) = A$, and b when *head*$(w) = B$.

(V1) rules:
$$\begin{array}{ll} CA \xrightarrow{a} C & C'A \xrightarrow{a} C' \\ CB \xrightarrow{b} C & C'B \xrightarrow{b} C' \\ C\bot \xrightarrow{e} \varepsilon & C'\bot \xrightarrow{e} \varepsilon \\ CI_i \xrightarrow{h(u_i^R)} C\ \mathit{tail}(u_i^R) & C'I_i \xrightarrow{h(v_i^R)} C'\ \mathit{tail}(v_i^R) \end{array}$$
$$\text{for all } i \in \{1, 2, \ldots, n\}$$

Let us now consider the system with the rules (G1), (S1), (V1), and the pair $(XI_1\bot,\ X'I_1\bot)$. Attacker can decide to perform a step of the generating phase by choosing the action c. He then has to use the rule $X \xrightarrow{c} Y$; any other c-move could be followed by Defender's response reaching syntactically equal processes

$(Y_iI_1\bot\,,\ Y_iI_1\bot\,)$ (which are trivially bisimilar). So it is Defender who (after Attacker's move $X \xrightarrow{c} Y$) chooses some $i \in \{1,2,\ldots,n\}$ by performing the rule $X' \xrightarrow{c} Y_i$. We thus get $(YI_1\bot\,,\ Y_iI_1\bot\,)$. The next action will be some $j \in \{1,2,\ldots,n\}$. If Attacker chooses $j \neq i$ then Defender installs syntactic equality $(\,XI_jI_1\bot\,,\ XI_jI_1\bot\,)$; so Attacker is forced to use the action i which means that the current round finishes in $(\,XI_iI_1\bot\,,\ X'I_iI_1\bot\,)$.

Attacker can decide to prolong the generating phase arbitrarily long but he always has the possibility to switch, by choosing the action d. In such case, from a current pair $(\,XI_{i_\ell}I_{i_{\ell-1}}\ldots I_{i_1}\bot\,,\ X'I_{i_\ell}I_{i_{\ell-1}}\ldots I_{i_1}\bot\,)$ he is forced to perform $X \xrightarrow{d} C$ to avoid syntactic equality. So the 'left-hand side' process becomes $CI_{i_\ell}I_{i_{\ell-1}}\ldots I_{i_1}\bot$, and Defender installs some $C'wI_{i_m}I_{i_{m-1}}\ldots I_{i_1}\bot$ on the 'right-hand side', where $m < \ell$ and w is a suffix of $v^R_{i_{m+1}}$.

One can easily check that the rules (V1) guarantee

$$CI_{i_\ell}I_{i_{\ell-1}}\ldots I_{i_1}\bot \sim C'wI_{i_m}I_{i_{m-1}}\ldots I_{i_1}\bot \ \text{ iff } \ u_{i_1}\ldots u_{i_\ell} = v_{i_1}\ldots v_{i_m}w^R$$

and that Defender has the possibility to install such a bisimilar pair (by using the rule $X'(I^*)I_i \xrightarrow{d} C'w$) iff $u_{i_1}\ldots u_{i_\ell}$ is a prefix of $v_{i_1}\ldots v_{i_\ell}$.

We also observe that $XI_1\bot$ and $X'I_1\bot$ are normed; we have thus proved the following lemma.

Lemma 1. *Problem inf-PCP is reducible to bisimilarity on normed Type -1a systems.*

Theorem 1. *Bisimilarity on Type -1a systems is Π^0_1-complete in both the normed case and the unrestricted case.*

Proof. Lemma 1 shows that bisimilarity is Π^0_1-hard already for normed Type -1a systems. Since (unrestricted) Type -1a systems are effectively image-finite, i.e., for each process w and every action a the set of a-successors of w is finite and effectively constructible, semidecidability of nonbisimilarity follows from Proposition 2. □

4.2 Π^0_1-Completeness of Normed Type -1b Systems

Normed Type -1b systems are handled very similarly as Type -1a, we only use different switching rules.

$$\text{(S2) rules:} \qquad \begin{array}{ll} & X' \xrightarrow{d} C' \\ X \xrightarrow{d} C(A+B)^* & X' \xrightarrow{d} C(A+B)^* \end{array}$$

When now, i.e., in the system (G1), (S2), (V1), Attacker decides to switch to the verification phase, in a current pair $(\,XI_{i_\ell}I_{i_{\ell-1}}\ldots I_{i_1}\bot\,,\ X'I_{i_\ell}I_{i_{\ell-1}}\ldots I_{i_1}\bot\,)$, he is forced to use $X' \xrightarrow{d} C'$ (on the right-hand side); Defender responds by extending the left-hand side. It is clear that there is an extension which guarantees Defender's win if and only if $u_{i_1}u_{i_2}\ldots u_{i_\ell}$ is a prefix of $v_{i_1}v_{i_2}\ldots v_{i_\ell}$.

Since $XI_1\bot$ and $X'I_1\bot$ are normed also in the system (G1), (S2), (V1), we have shown the following lemma.

Lemma 2. *Problem inf-PCP is reducible to bisimilarity on normed Type -1b systems.*

Theorem 2. *Bisimilarity on normed Type -1b systems is Π_1^0-complete.*

Proof. Π_1^0-hardness follows from Lemma 2. Type -1b systems are obviously effective, and for semidecidability of nonbisimilarity it is thus sufficient to show that *normed* Type -1b systems are finitely over-approximable and then use Proposition 2. (Note that Type -1b systems are in general not image-finite and hence the standard argument about semidecidability of the negative case does not directly apply.)

We recall that normed bisimilar processes must have equal norms (Proposition 3), and we note that $norm(u) \geq |u|/k$ where k is the length of the longest left-hand side in the rules $w \xrightarrow{a} R$ of the respective Type -1b system. Since $norm(u)$ is computable by Proposition 4, the required (finite) set $E_{(u,v,a)}$ for given processes u, v and an action a can be defined as $\{\, v' \mid |v'| \leq k \cdot norm(u) \,\}$. □

5 High Undecidability Results

We first note that (unrestricted) Type -2 systems represent a class of LTSs which satisfies the (straightforward) general criteria guaranteeing that the bisimilarity problem is in Σ_1^1 (see, e.g., [14]). (Processes u and v are bisimilar iff *there exists a set* of pairs which contains (u, v) and satisfies the conditions required by the definition of bisimulation.) So the main point in the following completeness results is to show Σ_1^1-hardness.

We again assume a fixed PCP-instance $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)$ (over the alphabet $\{A, B\}$); the instance is now viewed as an input of rec-PCP.

5.1 Σ_1^1-Completeness of (Unrestricted) Type -1b Systems

We modify the previously defined (normed) Type -1b system (G1), (S2), (V1). We first replace the (generating) rules (G1) with the following variant (G2), which repeatedly forces Defender to include the index 1 into the generated sequence. As before, I^* denotes $(I_1 + I_2 + \cdots + I_n)^*$.

(G2) rules:

$$\begin{array}{ll}
X \xrightarrow{c} Y & \\
X \xrightarrow{c} Y' I_1 I^* & X' \xrightarrow{c} Y' I_1 I^* \\
 & Y' \xrightarrow{c} X' \\
Y \xrightarrow{c} X I^* \bot & Y' \xrightarrow{c} X I^* \bot
\end{array}$$

Given a pair $(\, X I_{i_\ell} I_{i_{\ell-1}} \ldots I_{i_1} \bot \,,\; X' I_{i_\ell} I_{i_{\ell-1}} \ldots I_{i_1} \bot \,)$ (with $i_1 = i_\ell = 1$), if Attacker decides to continue the generating phase (i.e., chooses the action c) then he is forced to perform $X \xrightarrow{c} Y$ (on the left-hand side), otherwise he loses. Defender responds by the rule $X' \xrightarrow{c} Y' I_1 I^*$ (on the right-hand side), i.e., he prolongs the right-hand side sequence by an arbitrarily chosen finite segment finishing with I_1 (viewed from right to left). So we get the pair $(\,YI_{i_\ell}I_{i_{\ell-1}}\dots I_{i_1}\bot\,,\;Y'I_{i_m}I_{i_{m-1}}\dots I_{i_{\ell+1}}I_{i_\ell}I_{i_{\ell-1}}\dots I_{i_1}\bot\,)$ where $m>\ell$ and $i_m{=}1$.

Now unnormedness comes 'into play'. Our rules maintain the property that the suffix after (the leftmost) $\bot$ does not matter. So Attacker is forced to perform $Y'\xrightarrow{c}X'$ (on the right-hand side). Defender responds by 'killing' the left-hand side sequence (using a new occurrence of $\bot$) and creating a completely new one; important is that he has the possibility to install the pair

$$(\,XI_{i_m}I_{i_{m-1}}\dots I_{i_1}\bot w\,,\;X'I_{i_m}I_{i_{m-1}}\dots I_{i_1}\bot\,)$$

where w is unimportant and can be deemed omitted (which leaves the process in the same bisimilarity class).

However, we are not done yet. Unlike (G1), the new rules (G2) allow Defender to install an index sequence on the left-hand side which differs from the sequence he previously installed on the right-hand side. To prevent Defender from such 'cheating', we add some further switching and verification rules to (S2) and (V1). For this purpose we add new process symbols Z, Z' and an action f.

(S2') rules: (S2) and $\quad X\xrightarrow{f}Z \qquad X'\xrightarrow{f}Z'$

(V1') rules: (V1) and
$$\begin{array}{lll} ZI_i\xrightarrow{i}Z & Z'I_i\xrightarrow{i}Z' & \text{for all } i\in\{1,2,\dots,n\}\\ Z\bot\xrightarrow{e}\varepsilon & Z'\bot\xrightarrow{e}\varepsilon & \end{array}$$

We remind the reader of the fact that even though the last two rules above remove the symbol $\bot$ from the sequence of process symbols, whatever remains after $\bot$ can only start with some process symbol from the set $\{I_1,\dots,I_n\}$ and hence the remaining process is stuck (no left-hand side of any rule in our system begins with any I_i). Type -1b system (G2), (S2'), (V1') thus proves the next lemma, from which the following theorem is derived.

Lemma 3. *Problem rec-PCP is reducible to bisimilarity on (unrestricted) Type -1b systems.*

Theorem 3. *Bisimilarity on (unrestricted) Type -1b systems is Σ^1_1-complete.*

5.2 Σ^1_1-Completeness of Normed and Unrestricted Type -2 Systems

Σ^1_1-completeness for (unrestricted) Type -2 systems follows immediately from the previous results (Type -1b is a subclass of Type -2). So we just have to show that normedness does not make the problem easier in this case.

We recall the unnormed Type -1b system (G2), (S2'), (V1'). It is sufficient to replace the last two rules of (G2) (resp. their left-hand sides); we thus get

(G3) rules:
$$\begin{array}{ll} X\xrightarrow{c}Y & \\ X\xrightarrow{c}Y'I_1I^* & X'\xrightarrow{c}Y'I_1I^*\\ & Y'\xrightarrow{c}X'\\ YI^*\bot\xrightarrow{c}XI^*\bot & Y'I^*\bot\xrightarrow{c}XI^*\bot\quad . \end{array}$$

The processes $XI_1\bot$ and $X'I_1\bot$ in the resulting Type -2 system (G3), (S2'), (V1') are obviously normed (in any reachable process, $\bot$ can only occur as the last element in the sequence), and the correctness arguments remain the same. We have thus shown the following theorem.

Theorem 4. *Bisimilarity on Type -2 systems is Σ^1_1-complete in both the normed case and the unrestricted case.*

6 Conclusion and Final Remarks

We have answered negatively the open problem stated in 1996 by Stirling [27]: "Is strong bisimilarity decidable for Type -1 and Type -2 transition graphs?". A precise borderline between decidability and undecidability has been found: for Type -1a systems with rules of the form $R \xrightarrow{a} w$ where R is a prefix-free regular language bisimilarity is decidable [21], while it is undecidable for the same class without the prefix-freeness restriction. We have also given a full characterization of the undecidability degrees of the studied problems. A summary of the results for bisimilarity checking is provided in the following table. Results without references were obtained in this paper.

	Normed Processes	**Unnormed Processes**
Type -2	Σ^1_1-complete	Σ^1_1-complete
Type -1b	Π^0_1-complete	Σ^1_1-complete
Type -1a	Π^0_1-complete	Π^0_1-complete
Type 0, and Type $1\frac{1}{2}$	decidable [28] EXPTIME-hard [16]	decidable [19, 21] EXPTIME-hard [16]
Type 2	$\in$ P [13] P-hard [1]	$\in$ 2-EXPTIME [4] PSPACE-hard [22]
Type 3	P-complete [15, 1]	P-complete [15, 1]

We note that the results for Type -1b systems illustrate a significant difference between normed and unnormed processes. An open problem is the precise complexity for PDA and BPA, and decidability of bisimilarity for unrestricted regular (equational) graphs. As a side result, our paper provides an alternative and easily understandable proof of undecidability of weak bisimilarity for normed pushdown processes since the class of ε-collapsed pushdown graphs is a superclass of Type -2 systems [29] and hence (high) undecidability of strong bisimilarity for normed Type -2 graphs implies (high) undecidability of weak bisimilarity for normed pushdown processes.

Acknowledgments. The authors thank to Géraud Sénizergues for a motivating discussion (at University of Stuttgart), to the anonymous referees for their useful remarks, and acknowledge a support from the Alexander von Humboldt Foundation.

References

1. J. Balcazar, J. Gabarro, and M. Santha. Deciding bisimilarity is P-complete. *Formal Aspects of Computing*, 4(6A):638–648, 1992.
2. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *Proc. of the 8th International Conference on Concurrency Theory (CONCUR'97)*, vol. 1243 of *LNCS*, pages 135–150. Springer-Verlag, 1997.
3. O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on infinite structures. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, chapter 9, pages 545–623. Elsevier Science, 2001.
4. O. Burkart, D. Caucal, and B. Steffen. An elementary decision procedure for arbitrary context-free processes. In *Proc. of the 20th International Symposium on Mathematical Foundations of Computer Science (MFCS'95)*, vol. 969 of *LNCS*, pages 423–433. Springer-Verlag, 1995.
5. O. Burkart, D. Caucal, and B. Steffen. Bisimulation collapse and the process taxonomy. In *Proc. of the 7th International Conference on Concurrency Theory (CONCUR'96)*, vol. 1119 of *LNCS*, pages 247–262. Springer-Verlag, 1996.
6. T. Cachat. Uniform solution of parity games on prefix-recognizable graphs. *Electronic Notes in Theoretical Computer Science*, 68(6), 2002.
7. D. Caucal. On the regular structure of prefix rewriting. *Theoretical Computer Science*, 106(1):61–86, 1992.
8. D. Caucal. Bisimulation of context-free grammars and of pushdown automata. In *Modal Logic and Process Algebra*, vol. 53 of *CSLI Lectures Notes*, pages 85–106. University of Chicago Press, 1995.
9. D. Caucal. On infinite transition graphs having a decidable monadic theory. In *Proc. of the 23th International Colloquium on Automata, Languages and Programming (ICALP'96)*, vol. 1099, pages 194–205. Springer-Verlag, 1996.
10. J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *Proc. of the 12th International Conference on Computer Aided Verification (CAV'00)*, vol. 1855 of *LNCS*, pages 232–247. Springer-Verlag, 2000.
11. D. Harel. Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness. *Journal of the ACM (JACM)*, 33(1): 224–248, 1986.
12. M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the Association for Computing Machinery*, 32(1):137–161, 1985.
13. Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theoretical Computer Science*, 158 (1–2):143–159, 1996.
14. P. Jančar and J. Srba. Highly undecidable questions for process algebras. In *Proc. of the 3rd IFIP International Conference on Theoretical Computer Science (TCS'04)*, Exploring New Frontiers of Theoretical Informatics, pages 507–520. Kluwer Academic Publishers, 2004.
15. P.C. Kanellakis and S.A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.
16. A. Kučera and R. Mayr. On the complexity of semantic equivalences for pushdown automata and BPA. In *Proc. of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS'02)*, vol. 2420 of *LNCS*, pages 433–445. Springer-Verlag, 2002.

17. R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.
18. K. Ruohonen. Reversible machines and post's correspondence problem for biprefix morphisms. *Elektronische Informationsverarbeitung und Kybernetik*, 21(12):579–595, 1985.
19. G. Sénizergues. Decidability of bisimulation equivalence for equational graphs of finite out-degree. In *Proc. of the 39th Annual Symposium on Foundations of Computer Science(FOCS'98)*, pages 120–129. IEEE Computer Society, 1998.
20. G. Sénizergues. L(A)=L(B)? Decidability results from complete formal systems. *Theoretical Computer Science*, 251(1–2):1–166, 2001.
21. G. Senizergues. The bisimulation problem for equational graphs of finite out-degree. *SIAM Journal on Computing*, 34(5):1025–1106, 2005.
22. J. Srba. Strong bisimilarity and regularity of basic process algebra is PSPACE-hard. In *Proc. of the 29th International Colloquium on Automata, Languages and Programming (ICALP'02)*, vol. 2380 of *LNCS*, pages 716–727. Springer-Verlag, 2002.
23. J. Srba. Undecidability of weak bisimilarity for pushdown processes. In *Proc. of the 13th International Conference on Concurrency Theory (CONCUR'02)*, vol. 2421 of *LNCS*, pages 579–593. Springer-Verlag, 2002.
24. J. Srba. Completeness results for undecidable bisimilarity problems. In *Proc. of the 5th International Workshop on Verification of Infinite-State Systems (INFINITY'03)*, vol. 98 of *ENTCS*, pages 5–19. Elsevier Science Publishers, 2004.
25. J. Srba. *Roadmap of Infinite results*, vol. 2: Formal Models and Semantics. World Scientific Publishing Co., 2004. Updated version can be downloaded from the author's home-page.
26. C. Stirling. Local model checking games. In *Proc. of the 6th International Conference on Concurrency Theory (CONCUR'95)*, vol. 962 of *LNCS*, pages 1–11. Springer-Verlag, 1995.
27. C. Stirling. Decidability of bisimulation equivalence for normed pushdown processes. In *Proc. of the 7th International Conference on Concurrency Theory (CONCUR'96)*, vol. 1119 of *LNCS*, pages 217–232. Springer-Verlag, 1996.
28. C. Stirling. Decidability of bisimulation equivalence for normed pushdown processes. *Theoretical Computer Science*, 195(2):113–131, 1998.
29. C. Stirling. Decidability of bisimulation equivalence for pushdown processes. Research Report EDI-INF-RR-0005, School of Informatics, Edinburgh University, January 2000. The latest version is downloadable from the author's home-page.
30. C. Stirling. Bisimulation and language equivalence. In *Logic for Concurrency and Synchronisation*, vol. 18 of *Trends in Logic*, pages 269–284. Kluwer Academic Publishers, 2003.
31. W. Thomas. On the Ehrenfeucht-Fraïssé game in theoretical computer science (extended abstract). In *Proc. of the 4th International Joint Conference CAAP/FASE, Theory and Practice of Software Development (TAPSOFT'93)*, vol. 668 of *LNCS*, pages 559–568. Springer-Verlag, 1993.