Reachability is Decidable for Weakly Extended Process Rewrite Systems [⋆]

Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček

Faculty of Informatics, Masaryk University
Botanická 68a, CZ-602 00 Brno, Czech Republic
{kretinsky,rehak,strejcek}@fi.muni.cz

Abstract

Process Rewrite Systems (PRS) are widely accepted as a formalism for the description of infinite-state systems. It is known that the reachability problem for PRS is decidable. The problem becomes undecidable when PRS are extended with a finite-state control unit. In this paper we show that the problem remains decidable when PRS are extended with a weak (i.e. acyclic except for self-loops) finite-state control unit. We also present some applications of this decidability result.

1 Introduction

Automatic verification of current software systems often needs to model them as systems with an evolving structure and/or operating on unbounded data types, i.e. as infinite-state systems.

Infinite-state systems can be specified in a number of ways with their respective advantages and limitations. Petri nets, pushdown processes, and process algebras like BPA, BPP, or PA all serve to exemplify this. Here we employ the classes of infinite-state systems defined by term rewrite systems and called *Process Rewrite Systems (PRS)* as introduced by Mayr [2]. PRS subsume a variety of the formalisms studied in the context of formal verification (e.g. all the models mentioned above).

A PRS is a finite set of rules of the form $t \overset{a}{\hookrightarrow} t'$, where a is an action under which a subterm t can be reduced to a subterm t'. Terms are built up from an empty process ε and a set of process constants using (associative) sequential

[⋆] Some parts of this work have been reported in a conference paper [1].

"." and (associative and commutative) parallel "$\|$" operators. The semantics of PRS can be defined by *labelled transition systems (LTS)* – labelled directed graphs whose nodes (states of the system) correspond to terms modulo structural congruence induced by neutrality of ε and properties of "." and "$\|$", and whose edges correspond to individual actions (computation steps) which can be performed in a given state. The relevance of various subclasses of PRS for modelling and analysing programs is shown e.g. in [3], for automatic verification see e.g. surveys [4,5].

Mayr [2] has also shown that the reachability problem (i.e. given terms t, t': is t reducible to t'?) for PRS is decidable. Most research (with some recent exceptions, e.g. [6,3]) has been devoted to the PRS classes from the lower part of the PRS hierarchy depicted in Figure 1, especially to *pushdown processes (PDA)*, *Petri nets (PN)* and their respective subclasses. We mention the successes of PDA in modeling recursive programs (without process creation), PN in modeling dynamic creation of concurrent processes (without recursive calls), and *communicating pushdown systems (CPDS)* [7] in modeling both features. All of these formalisms subsume a notion of a *finite-state control unit (FSU)* keeping some kind of global information which is accessible to the redexes (the components that can be reduced) of a PRS term – hence an FSU can regulate rewriting. On the other hand, using an FSU to extend the PRS rewriting mechanism is very powerful since the reachability problem becomes undecidable for a *state-extended* version of *PA processes (sePA)* [8], and CPDS as well.

This paper deals with a hierarchy of PRS classes and their respective extensions of two types: *weakly extended PRS (wPRS)* [9,1] classes (i.e. PRS systems equipped with weak FSU inspired by weak automata [10]) and state-extended PRS classes [11]. In this paper we omit the extension introduced in [12] under the name *PRS with finite constraint unit (fcPRS)*. However, decidability of the reachability problem for wPRS carries over to fcPRS as every fcPRS system can be translated into an isomorphic wPRS system in a straightforward way. For more information about fcPRS and its expressiveness the reader is referred to [12,9]. The classes in the hierarchy (depicted in Figure 1) are related by their expressive power with respect to (strong) bisimulation equivalence. As the main contribution of the paper we show that the reachability problem remains decidable for the very expressive class of wPRS. This result determines the decidability borderline of the reachability problem in the mentioned hierarchy: the problem is decidable for all classes except the sePA class and its superclasses. Moreover, the result has several applications. In this paper, two of them are discussed in more detail, namely

- decidability of some safety properties over wPRS and
- semi-decidability of weak trace non-equivalence for wPRS.

The outline of the paper is as follows: Section 2 recalls syntax and semantics of PRS. Extended versions of PRS are defined in Section 3. This section also presents the hierarchy reflecting relative expressiveness of PRS classes and their extended versions with respect to bisimulation equivalence. In Section 4 we show that the reachability problem is decidable for weakly extended PRS. Section 5 is devoted to the applications of our decidability result. The last section summarises our results.

Related work: In the context of reachability analysis one can see at least two approaches: (i) abstraction (approximate) analysis techniques on 'stronger' models such as sePA and its superclasses with undecidable reachability, e.g. see a recent work [7], and (ii) precise techniques for computing the set of states that are reachable from a given regular set of states, e.g. [13,14,6]. In the latter approach, the sets are represented symbolically and various term structural equivalences are considered. The papers dealing with this approach usually work with the classes PA or PAD rather than with general PRS systems.

2 Preliminaries

A *labelled transition system (LTS)* $\mathcal{L}$ is a tuple $(S, Act, \longrightarrow, \alpha_0)$, where S is a set of *states* or *processes*, Act is a set of *atomic actions* or *labels*, $\longrightarrow \subseteq S \times Act \times S$ is a *transition relation* (written $\alpha \xrightarrow{a} \beta$ instead of $(\alpha, a, \beta) \in \longrightarrow$), and $\alpha_0 \in S$ is a distinguished *initial state.*

We use the natural generalizations $\alpha \xrightarrow{\sigma} \beta$ for finite sequence of actions $\sigma \in Act^*$. Next, $\alpha \longrightarrow \beta$ means $\alpha \xrightarrow{a} \beta$ for some $a \in Act$, and $\alpha \longrightarrow^* \beta$ means $\alpha \xrightarrow{\sigma} \beta$ for some $\sigma \in Act^*$. A state β is *reachable from a state* α if $\alpha \longrightarrow^* \beta$. Further, β is *reachable* if it is reachable from the initial state.

A binary relation R on a set of states S is a *bisimulation* [15] iff for each $(\alpha, \beta) \in R$ the following conditions hold:

- $\forall \alpha' \in S, a \in Act : \alpha \xrightarrow{a} \alpha' \Longrightarrow (\exists \beta' \in S : \beta \xrightarrow{a} \beta' \wedge (\alpha', \beta') \in R)$
- $\forall \beta' \in S, a \in Act : \beta \xrightarrow{a} \beta' \Longrightarrow (\exists \alpha' \in S : \alpha \xrightarrow{a} \alpha' \wedge (\alpha', \beta') \in R)$

Bisimulation equivalence (or *bisimilarity*) on an LTS is the union of all bisimulations (i.e. the largest bisimulation).

Let $Const = \{X, \ldots\}$ be a set of *process constants.* The set $\mathcal{T}$ of *process terms* (ranged over by $t, \ldots$) is defined by the abstract syntax $t = \varepsilon \mid X \mid t_1.t_2 \mid t_1 \| t_2$, where ε is the empty term, $X \in Const$ is a process constant (used as an atomic process), '$\|$' and '.' mean parallel and sequential compositions respectively.

The set $\mathit{Const}(t)$ is the set of all constants occurring in the process term t. We always work with equivalence classes of terms modulo commutativity and associativity of '$\|$', associativity of '.', and neutrality of ε, i.e. $\varepsilon.t = t = t.\varepsilon$ and $t\|\varepsilon = t$.

We distinguish four *classes of process terms* as:

1 – terms consisting of a single process constant only, in particular $\varepsilon \notin 1$,
S – *sequential* terms - terms without parallel composition, e.g. $X.Y.Z$,
P – *parallel* terms - terms without sequential composition, e.g. $X\|Y\|Z$,
G – *general* terms - terms with arbitrarily nested sequential and parallel compositions, e.g. $(X.(Y\|Z))\|W$.

Definition 1 *Let $Act = \{a, b, \cdots\}$ be a set of* atomic actions*, $\alpha, \beta \in \{1, S, P, G\}$ such that $\alpha \subseteq \beta$. An (α, β)-*PRS (process rewrite system) *Δ is defined as a pair (R, t_0), where*

- *R is a finite set of* rewrite rules *of the form $t_1 \overset{a}{\hookrightarrow} t_2$, where $t_1 \in \alpha$, $t_1 \neq \varepsilon$, $t_2 \in \beta$ are process terms and $a \in Act$ is an atomic action,*
- *$t_0 \in \beta$ is the* initial state*.*

Given a PRS Δ we define $\mathit{Const}(\Delta)$ as the set of all constants occurring in the rewrite rules of Δ or in its initial state, and $\mathit{Act}(\Delta)$ as the set of all actions occurring in the rewrite rules of Δ. We usually write $(t_1 \overset{a}{\hookrightarrow} t_2) \in \Delta$ instead of $(t_1 \overset{a}{\hookrightarrow} t_2) \in R$ where $\Delta = (R, t_0)$.

The semantics of Δ is given by the LTS $(S, \mathit{Act}(\Delta), \longrightarrow_\Delta, t_0)$, where $S = \{t \in \beta \mid \mathit{Const}(t) \subseteq \mathit{Const}(\Delta)\}$ is the set of states, t_0 is the initial state and $\longrightarrow_\Delta$ is the least relation satisfying the inference rules[1]

$$\frac{(t_1 \overset{a}{\hookrightarrow} t_2) \in \Delta}{t_1 \xrightarrow{a}_\Delta t_2} \qquad \frac{t_1 \xrightarrow{a}_\Delta t_1'}{t_1\|t_2 \xrightarrow{a}_\Delta t_1'\|t_2} \qquad \frac{t_1 \xrightarrow{a}_\Delta t_1'}{t_1.t_2 \xrightarrow{a}_\Delta t_1'.t_2}$$

If no confusion arises, we sometimes speak about a "process rewrite system" meaning the "labelled transition system generated by a process rewrite system".

Some classes of (α, β)-PRS correspond to widely known models: the $(1, 1)$-PRS class corresponds to finite state systems (FS), $(1, P)$-PRS is the class of basic parallel processes (BPP), the $(1, S)$-PRS class is known as basic process algebras (BPA), $(1, G)$-PRS is the process algebra (PA) class, the (S, S)-PRS

[1] Note that parallel composition is commutative and, thus, the inference rule for the parallel composition also holds with t_1 and t_2 exchanged.

class corresponds to pushdown processes (PDA, see [16] for justification), and (P,P)-PRS is the class of Petri nets (PN). The other classes (S,G)-PRS, (P,G)-PRS, and (G,G)-PRS were introduced (and named as PAD, PAN, and PRS) by Mayr [2]. The correspondence between (α,β)-PRS classes and the acronyms just mentioned can be seen in Figure 1 as well.

3 Extended PRS

In this section we recall the definitions of two extensions of process rewrite systems, namely *state-extended PRS (sePRS)* [11] and *weakly extended PRS (wPRS)* [9].

sePRS State-extended PRS corresponds to PRS extended with a finite-state control without any other restrictions. The well-known example of this extension is the state-extended BPA class (also known as pushdown processes).

wPRS The notion of weakness employed in the wPRS formalism corresponds to that of weak automata [10] in automata theory. The behaviour of a weak state control is acyclic except for self-loops, i.e. the control states are ordered and non-increasing during every sequence of transitions. As the control is finite, its state can be changed only finitely many times during every sequence of transitions.

We first define the syntax of sePRS and wPRS systems and then we give a semantics for both these PRS extensions.

Definition 2 *Let* $Act = \{a, b, \cdots\}$ *be a set of* atomic actions, $\alpha, \beta \in \{1, S, P, G\}$ *such that* $\alpha \subseteq \beta$. *An* (α,β)-sePRS Δ *is a tuple* (M, R, m_0, t_0), *where*

- M *is a finite set of the* control states,
- R *is a finite set of* rewrite rules *of the form* $(m, t_1) \overset{a}{\hookrightarrow} (n, t_2)$, *where* $t_1 \in \alpha$, $t_1 \neq \varepsilon$, $t_2 \in \beta$, $m, n \in M$, *and* $a \in Act$,
- *a pair* $(m_0, t_0) \in M \times \beta$ *forms a distinguished* initial state *of the system.*

An (α,β)-wPRS Δ *is a tuple* $(M, \leq, R, m_0, t_0)$, *where the set of states* $(M, \leq)$ *is partially ordered and each rewrite rule* $(m, t_1) \overset{a}{\hookrightarrow} (n, t_2)$ *in* R *satisfies* $n \leq m$; *the other symbols have the same meaning as above.*

To shorten our notation we write mt instead of (m,t). As in the PRS case, instead of $(mt_1 \overset{a}{\hookrightarrow} nt_2) \in R$ where $\Delta = (M, R, m_0, t_0)$ (or $\Delta = (M, \leq, R, m_0, t_0)$), we usually write $(mt_1 \overset{a}{\hookrightarrow} nt_2) \in \Delta$. The meaning of $Const(\Delta)$ (process constants used in rewrite rules or in t_0) and $Act(\Delta)$ (actions occurring in rewrite

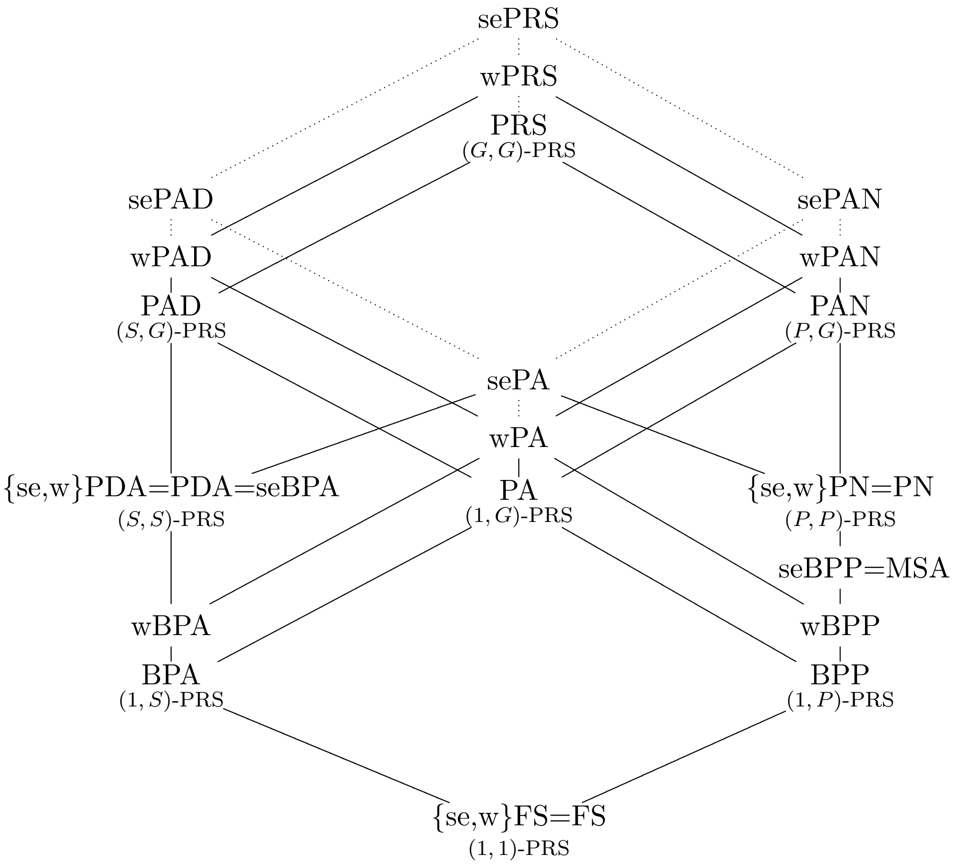

Fig. 1. The hierarchy of classes defined by (extended) rewrite formalisms.

rules) for a given extended PRS Δ is also the same as in the PRS case.

The semantics of an extended (α, β)-PRS system Δ is given by the corresponding labelled transition system $(S, Act(\Delta), \longrightarrow_\Delta, m_0t_0)$, where

$$S = M \times \{t \in \beta \mid Const(t) \subseteq Const(\Delta)\}$$

and the relation $\longrightarrow_\Delta$ is defined as the least relation satisfying the inference rules

$$\frac{(mt_1 \overset{a}{\hookrightarrow} nt_2) \in \Delta}{mt_1 \xrightarrow{a}_\Delta nt_2}, \qquad \frac{mt_1 \xrightarrow{a}_\Delta nt_1'}{m(t_1 \| t_2) \xrightarrow{a}_\Delta n(t_1' \| t_2)}, \qquad \frac{mt_1 \xrightarrow{a}_\Delta nt_1'}{m(t_1.t_2) \xrightarrow{a}_\Delta n(t_1'.t_2)},$$

where $t_1, t_2, t_1' \in \mathcal{T}$ and $m, n \in M$.

Instead of $(1,S)$-sePRS, $(1,S)$-wPRS, ... we use a more natural notation seBPA, wBPA, etc. The class seBPP is also known as *multiset automata (MSA)* or *parallel pushdown automata (PPDA)*, see [17].

Figure 1 describes the hierarchy of PRS classes and their extended counterparts with respect to bisimulation equivalence. More precisely, the classes of (extended) PRS systems are here interpreted as the sets of their underlying labelled transition systems. The depicted hierarchy is then the upward oriented Hasse diagram of a partial order relation '$\subseteq$' between these sets of labelled transition systems modulo bisimulation equivalence. In other words, a line connecting X and Y with Y placed higher than X means that every transition system definable in X can be (up to bisimulation equivalence) defined in Y while the reverse does not hold – we write $X \subsetneq Y$. The dotted lines represent the facts $X \subseteq Y$, where the relation $X \subsetneq Y$ is only our conjecture.

Some observations (even up to isomorphism) are immediate, for example

(1) the classes FS, PDA, and PN coincide with their extended analogues,
(2) if X $\subseteq$ Y then seX $\subseteq$ seY and wX $\subseteq$ wY, and
(3) (α, β)-PRS $\subseteq$ (α, β)-wPRS $\subseteq$ (α, β)-sePRS for all (α, β)-PRS.

All the relations represented in the hierarchy have been proven in [1,18,2,12,9] (see [1] for more detailed comments).

The two lines leaving sePA down to the left and down to the right deserve further comments. The classes (S, S)-PRS and seBPA on the left-hand side collapse (up to isomorphism) due to Caucal [16]. The situation on the right-hand side is different due to the relations MSA $\subsetneq$ PN and PN $\subsetneq$ sePA established in [18] and [1] respectively. (Strictness of the last relation follows from incomparability of the classes PDA and PN.)

4 Reachability for wPRS is decidable

In this section we show that for a given wPRS Δ and its states rt_1, st_2 it is decidable whether st_2 is reachable from rt_1, i.e. whether $rt_1 \longrightarrow^*_\Delta st_2$. This is called the *reachability problem* for wPRS.

Our proof exhibits a similar structure to the proof of decidability of the reachability problem for PRS [2]; first we reduce the general problem to the reachability problem for wPRS with rules containing at most one occurrence of a sequential or parallel operator, and then we solve this subproblem using the fact that the reachability problems for both PN and PDA are decidable [19,20]. The latter part of our proof is based on a new idea of *passive steps* presented later.

To get just a sketch of the entire proof we suggest to read the definitions and statements (skipping their technical proofs). Several of them are preceded by comments that provide some intuition. As the labels on rewrite rules are not relevant here, we omit them in this section.

Definition 3 *Let Δ be a wPRS. A rewrite rule in Δ is* parallel *or* sequential *if it has one of the following forms:*

$$\text{parallel rules: } pX \hookrightarrow q(Y\|Z) \quad p(X\|Y) \hookrightarrow qZ \quad pX \hookrightarrow qY \quad pX \hookrightarrow q\varepsilon,$$

$$\text{sequential rules: } pX \hookrightarrow q(Y.Z) \quad p(X.Y) \hookrightarrow qZ \quad pX \hookrightarrow qY \quad pX \hookrightarrow q\varepsilon,$$

where X, Y, Z are process constants and p, q are control states. A rule is trivial *if it is both parallel and sequential (i.e. it has the form $pX \hookrightarrow qY$ or $pX \hookrightarrow q\varepsilon$). A wPRS Δ is in* normal form *if every rewrite rule in Δ is either parallel or sequential.*

Lemma 4 *Given a wPRS Δ with terms t_1 and t_2, we can effectively construct a wPRS Δ' in normal form over the same control states, along with terms t'_1 and t'_2, such that $rt_1 \longrightarrow^*_\Delta st_2$ iff $rt'_1 \longrightarrow^*_{\Delta'} st'_2$.*

PROOF. In this proof we assume that the sequential composition is left-associative. It means that the term $X.Y.Z$ is considered as $(X.Y).Z$, hence its proper subterms are X, Y, Z, and $X.Y$, but not $Y.Z$. However, the term $Y\|Z$ is a subterm of $X.(Y\|Z)$.

Let $size(t)$ denote the number of sequential and parallel operators in a term t. We put $size(pt \hookrightarrow qt') = size(t) + size(t')$. Given any wPRS Δ, let k_i be the number of rules $(pt \hookrightarrow qt') \in \Delta$ that are neither parallel nor sequential and $size(pt \hookrightarrow qt') = i$. Thus, Δ is in normal form iff $k_i = 0$ for all i. In this case, let $n = 0$. Otherwise, let n be the largest i such that $k_i \neq 0$ (n exists as the set of rules is finite). We define $norm(\Delta)$ to be the pair (n, k_n).

We now describe a procedure transforming any given wPRS Δ which is not in normal form and terms t_1, t_2 into a wPRS Δ' and terms t'_1, t'_2 such that $rt_1 \longrightarrow^*_\Delta st_2 \iff rt'_1 \longrightarrow^*_{\Delta'} st'_2$ and $norm(\Delta') < norm(\Delta)$ with respect to the lexicographical ordering on norms as pairs of integers.

Let us assume that a wPRS Δ is not in normal form. Then there is a rule that is neither sequential nor parallel and has the maximal *size*. If the rule is a of the form $p(X_1.X_2) \hookrightarrow q(Y_1\|Y_2)$ or $p(Y_1\|Y_2) \hookrightarrow q(X_1.X_2)$, let t be $X_1.X_2$; otherwise, let t be a non-atomic and proper subterm of this rule. Now, replace every occurrence of the subterm t in rewrite rules of Δ and in the terms t_1, t_2 by a fresh constant X_t. Then add two rules $pX_t \hookrightarrow pt$ and $pt \hookrightarrow pX_t$ for each control state p. This yields a new wPRS Δ' and terms t'_1 and t'_2 where the constant X_t serves as an abbreviation for the term t. By the definition of *norm* we get $norm(\Delta') < norm(\Delta)$. The correctness of our transformation remains to be demonstrated, namely that

$$rt_1 \longrightarrow^*_\Delta st_2 \iff rt'_1 \longrightarrow^*_{\Delta'} st'_2.$$

The implication $\Longleftarrow$ is obvious. For the opposite direction we show that every rewriting step in Δ from pl_1 to ql_2 under the rule $(pl \hookrightarrow ql') \in \Delta$ corresponds to a sequence of several rewriting steps in Δ' leading from pl'_1 to ql'_2, where l'_1, l'_2 are equal to l_1, l_2 with all occurrences of t replaced by X_t. Let us assume the rule $pl \hookrightarrow ql'$ modifies a subterm t of pl_1, and/or a subterm t appears in ql_2 after the rule application (the other cases are trivial). If the rule modifies a subterm t of l_1 then there are two cases.

(1) Let l include the whole t. Then the corresponding rule in Δ' (with t replaced by X_t) can be applied directly on pl'_1.
(2) Let l contain a part of t only. Due to the left-associativity of a sequential operator, t is not a subterm of the right part of any sequential composition in l_1. Thus we apply the added rule $pX_t \hookrightarrow pt$ on pl'_1 first and then we apply the rule in Δ' corresponding to the rule $pl \hookrightarrow ql'$.

The situation when t appears in ql_2 after the application of the considered rule is similar. Either l' includes the whole t and then the application of the corresponding rule in Δ' results directly in ql'_2, or t is not a subterm of the right part of any sequential composition in l_2 and thus the application of the corresponding rule in Δ' is followed by an application of the added rule $qt \hookrightarrow qX_t$ reaching the state ql'_2.

By repeating this procedure we finally get a wPRS Δ'' in normal form and terms t''_1,t''_2 satisfying $rt_1 \longrightarrow^*_\Delta st_2 \iff rt''_1 \longrightarrow^*_{\Delta''} st''_2$. □

Mayr's proof for PRS now transforms the PRS Δ in normal form into the PRS Δ' in so-called *transitive normal form* satisfying $(X \hookrightarrow Y) \in \Delta'$ whenever $X \longrightarrow^*_{\Delta'} Y$. This step employs the fact that rewriting under sequential rules in a parallel environment (or vice versa) has "local effect" only. Intuitively, whenever there is a rewriting sequence

$$X \| Y \longrightarrow^*_\Delta (X_1.X_2) \| Y \longrightarrow^*_\Delta (X_1.X_2) \| Z \longrightarrow^*_\Delta X_2 \| Z$$

in a PRS in normal form, then the rewriting of each parallel component is independent in the sense that there are also rewriting sequences $X \longrightarrow^*_\Delta X_1.X_2 \longrightarrow^*_\Delta X_2$ and $Y \longrightarrow^*_\Delta Z$. This does not hold for wPRS in normal form as the rewriting in one parallel component can influence the rewriting in other parallel components via a weak control. To get this independence back we introduce the concept of *passive steps* emulating the changes of a control state produced by the environment.

Definition 5 *A finite sequence of control state pairs $PS = \{(p_i, q_i)\}_{i=1}^n$ satisfying $p_1 > q_1 \geq p_2 > q_2 \geq \cdots \geq p_n > q_n$ is called a sequence of* passive steps*, or just passive steps for short.*

Let Δ be a wPRS and PS be passive steps. By Δ^{PS} we denote the system

Δ with an added rule $pX \hookrightarrow qX$ for each (p,q) in PS and $X \in Const(\Delta)$. Further, we define Δ_{triv}, Δ_{seq}, and Δ_{par} to be the subset of trivial, sequential, and parallel rules of Δ, respectively.

Informally, $rt_1 \longrightarrow^*_{\Delta^{PS}} st_2$ means that the state rt_1 can be rewritten into the state st_2 provided a control state can be passively changed from p to q for every passive step (p,q) in PS. Please note that there is only a finite number of different sequences of passive steps for a given wPRS system.

Definition 6 *Let wPRS Δ be in normal form. If for every $X, Y \in Const(\Delta)$, control states r, s, and passive steps PS it holds that*

$rX \longrightarrow^*_{\Delta^{PS}} sY \implies rX \longrightarrow^*_{\Delta^{PS}_{triv}} sY$ *then Δ is in* flat normal form*,*

$rX \longrightarrow^*_{\Delta^{PS}_{seq}} sY \implies rX \longrightarrow^*_{\Delta^{PS}_{triv}} sY$ *then Δ is in* sequential flat normal form*,*

$rX \longrightarrow^*_{\Delta^{PS}_{par}} sY \implies rX \longrightarrow^*_{\Delta^{PS}_{triv}} sY$ *then Δ is in* parallel flat normal form*.*

The following lemma says that it is sufficient to check reachability via sequential rules and via parallel rules in order to construct a wPRS in flat normal form. This allows us to reduce the reachability problem for wPRS to the reachability problems for wPN and wPDA, i.e. to the reachability problems for PN and PDA.

Lemma 7 *If a wPRS is in both sequential and parallel flat normal form then it is in flat normal form as well.*

PROOF. We assume the contrary and derive a contradiction. Let Δ be a wPRS in sequential and parallel flat normal form. Let us choose passive steps PS and a rewriting sequence $rX \longrightarrow^*_{\Delta^{PS}} sY$ such that $rX \not\longrightarrow^*_{\Delta^{PS}_{triv}} sY$ and the number of applications of non-trivial rewrite rules applied in the sequence is minimal. As the wPRS Δ is in both sequential and parallel flat normal form, $rX \not\longrightarrow^*_{\Delta^{PS}_{seq}} sY$ and $rX \not\longrightarrow^*_{\Delta^{PS}_{par}} sY$. Hence, both sequential and parallel operators occur in the rewriting sequence. There are two following cases.

(1) Assume that a sequential operator appears first. The parallel operator is then introduced by a rule of the form $pU \hookrightarrow q(T\|S)$ applied to a state $p(U.t)$, where t is a (possibly empty) sequential term. Note that $q((T\|S).t) \longrightarrow^*_{\Delta^{PS}} sY$ and recall the fact that at most one process constant can be removed in one rewriting step. Hence, first of all the term $T\|S$ is rewritten onto some single process constant V in the rest of the sequence considered. Let o be a control state after this rewriting. Using the same rewriting steps as in the original sequence, pU can be rewritten to oV in system Δ^{PS}. Let $PS'' = PS$.

(2) Assume that a parallel operator appears first. The sequential operator

is then introduced by a rule of the form $pU \hookrightarrow q(T.S)$ applied to a state $p(U\|t)$, where t is a (possibly empty) parallel term. The rest of the sequence subsumes steps rewriting the term $T.S$ onto some single process constant V. Let o be a control state in the state where $T.S$ is rewritten to V. Contrary to the previous case, the mentioned steps can be interleaved with steps rewriting the parallel component t and possibly changing a control state. Let PS' be a sequence of control state pairs corresponding to the changes of control states caused by rewriting of the parallel component t. We merge PS' with the subsequence of PS containing only the steps employed in the considered rewriting sequence. As the result we get one sequence of passive steps denoted as PS''. Please note that the elimination of the unused steps of PS ensures that PS'' satisfies the definition of passive steps. Now, making use of the passive steps PS'' and the steps rewriting U to V in the original sequence, we construct a rewriting sequence in system $\Delta^{PS''}$ leading from pU to oV.

Thus we have obtained a rewriting sequence in $\Delta^{PS''}$ from pU to oV with fewer applications of non-trivial rewrite rules – we omit at least the first application of a non-trivial rewrite rule in the original sequence. Further, at least the first step of the new sequence is an application of a non-trivial rewrite rule. Moreover, as the number of applications of non-trivial rewrite rules used in the original sequence is minimal, we get $pU \not\longrightarrow^*_{\Delta^{PS''}_{triv}} oV$. This contradicts our initial assumptions about the choices of PS and the rewriting sequence in Δ^{PS}. □

Example 8 *Here, we illustrate a possible change of passive steps (PS to PS″) described in the second case of the proof above. Let us consider a wPRS Δ with control states $r > p > q > t > v > o > s$ and the following rewrite rules*

$$\begin{array}{ccc} rX \hookrightarrow p(U\|Z) & pU \hookrightarrow q(T.S) & v(T.S) \hookrightarrow oV \\ & qZ \hookrightarrow tY & o(V\|Y) \hookrightarrow sY \end{array}$$

as well as the following sequence in Δ^{PS} where $PS = \{(t,v)\}$

$$\begin{array}{l} r\underline{X} \longrightarrow_{\Delta^{PS}} p(\underline{U}\|Z) \longrightarrow_{\Delta^{PS}} q((T.S)\|\underline{Z}) \longrightarrow_{\Delta^{PS}} \\ \longrightarrow_{\Delta^{PS}} \underline{t}((T.S)\|Y) \overset{passive}{\longrightarrow}_{\Delta^{PS}} v((\underline{T.S})\|Y) \longrightarrow_{\Delta^{PS}} o(\underline{V\|Y}) \longrightarrow_{\Delta^{PS}} sY \end{array}$$

where redexes are underlined. The sequence of passive steps constructed due to the case 2 is $PS'' = \{(q,t),(t,v)\}$ and the constructed rewriting sequence is

$$p\underline{U} \longrightarrow_{\Delta^{PS''}} \underline{q}(T.S) \overset{passive}{\longrightarrow}_{\Delta^{PS''}} \underline{t}(T.S) \overset{passive}{\longrightarrow}_{\Delta^{PS''}} v(\underline{T.S}) \longrightarrow_{\Delta^{PS''}} oV.$$

The following lemma employs the algorithms deciding the reachability problem for PDA and PN. Recall that the classes PDA and PN coincide with the classes of wPDA and wPN, respectively.

Lemma 9 *For every wPRS Δ in normal form, terms t_1, t_2 over $Const(\Delta)$, and control states r, s of Δ, a wPRS Δ' can be constructed such that Δ' is in flat normal form and satisfies $rt_1 \longrightarrow^*_{\Delta} st_2 \iff rt_1 \longrightarrow^*_{\Delta'} st_2$.*

PROOF. To obtain Δ' we enrich Δ by trivial rewrite rules transforming the system into sequential and parallel flat normal form, which suffices thanks to Lemma 7.

Using the algorithms deciding reachability for PDA and PN, our algorithm checks if there are some control states r, s, constants $X, Y \in Const(\Delta)$, and passive steps $PS = \{(p_i, q_i)\}_{i=1}^{n}$ (satisfying $r \geq p_1$ and $q_n \geq s$ as control states pairs out of this range are of no use here) such that $rX \not\longrightarrow^*_{\Delta^{PS}_{triv}} sY$, but $rX \longrightarrow^*_{\Delta^{PS}_{seq}} sY$ or $rX \longrightarrow^*_{\Delta^{PS}_{par}} sY$ hold. We finish if the answer is negative. Otherwise we add to Δ the rules $rX \hookrightarrow p_1 Z_1$, $q_i Z_i \hookrightarrow p_{i+1} Z_{i+1}$ for $i = 1, \ldots, n-1$, and $q_n Z_n \hookrightarrow sY$, where $Z_1, \ldots, Z_n$ are fresh process constants; if $n = 0$ then we add just the rule $rX \hookrightarrow sY$. Hence, $rX \longrightarrow^*_{\Delta''^{PS}_{triv}} sY$ where Δ'' is the system Δ with the new rules.

The algorithm then repeats this procedure on the system Δ'' with one difference: the X, Y range over the constants of the original system Δ. This is sufficient as the new constants occur only in trivial rules. Thus, if the system with added rules is not in sequential or parallel flat normal form, then there is a counterexample with the constants X, Y of the original system Δ. The algorithm eventually terminates as the number of iterations is bounded by the number of pairs of states rX, sY of Δ, times the number of sequences of passive steps PS. The correctness follows from the fact that the added rules only duplicate existing rewrite sequences between states of Δ. □

Theorem 10 *The reachability problem for wPRS is decidable.*

PROOF. Let Δ be a wPRS with states rt_1, st_2. We want to decide whether $rt_1 \longrightarrow^*_{\Delta} st_2$ or not. We assume that $rt_1 \neq st_2$ (the other case is trivial).

Clearly $rt_1 \longrightarrow^*_{\Delta} st_2 \iff rX \longrightarrow^*_{\Delta''} sY$, where X, Y are fresh constants and Δ'' arises from Δ by the addition of the rules $rX \hookrightarrow rt_1$ and $st_2 \hookrightarrow sY$ (if $t_2 = \varepsilon$ then the latter rule is not correct rule; in this case we add to Δ'' a rule $pt \hookrightarrow qY$ for each rule $(pt \hookrightarrow q\varepsilon) \in \Delta$ instead). Lemma 4 and Lemma 9 successively reduce the question whether $rX \longrightarrow^*_{\Delta''} sY$ to the question whether $rX \longrightarrow^*_{\Delta'} sY$, where Δ' is in flat normal form – note that the algorithm in the proof of Lemma 4 does not change terms t_1, t_2 if they are process constants. The definition of flat normal form implies $rX \longrightarrow^*_{\Delta'} sY \iff rX \longrightarrow^*_{\Delta'_{triv}} sY$. Finally the relation $rX \longrightarrow^*_{\Delta'_{triv}} sY$ is easy to check. □

5 Applications

In this section we discuss some applications of the decidability result presented in the previous section.

5.1 Model checking some safety properties

In the context of verification, one often formulates a property expressing that *some "bad" states are not reachable.* These properties are called *safety properties.* If the number of bad states is finite, the problem can be directly solved using reachability problem; but it is usually not the case. Usually, the bad states are characterized as those satisfying some specific property, e.g. to be a deadlock state, an internal variable x is equal to zero, stack overflows, division by zero is perfomed, etc. In what follows, we solve model checking for wPRS and only such safety properties that express the bad states as those where a transition with a given label is enabled. In particular, we solve a problem whether, for given wPRS Δ and its action *bad*, there is a reachable state in which a transition with the label *bad* is enabled.

Lemma 11 *Given a wPRS Δ and $bad \in Act(\Delta)$, it is decidable whether there exists a reachable state mt such that $mt \xrightarrow{bad}_{\Delta} nt'$ for some state nt'.*

PROOF. The proof is done by reduction to the reachability problem. Let $\Delta = (M, \leq, R, m_0, t_0)$. We construct a wPRS $\Delta' = (M', \leq', R', m_0, t_0)$, where $(M', \leq')$ is $(M, \leq)$ extended with a new control state r which is the least with respect to $\leq$ and where R' arises from R by adding the following rewrite rules:

$$
\begin{array}{lll}
(1) & mt_1 \overset{bad}{\hookrightarrow} rt_1 & \text{for all } (mt_1 \overset{bad}{\hookrightarrow} nt_2) \in \Delta, \\
(2) & rX \overset{bad}{\hookrightarrow} r\varepsilon & \text{for all } X \in Const(\Delta).
\end{array}
$$

The rules of type (1) allow us to change any control state to r whenever a *bad* transition is enabled in the original system. Entering the control state r, a term can be rewritten to ε using the rules of type (2). Hence, a state mt such that $mt \xrightarrow{bad}_{\Delta} nt'$ for some state nt' is reachable in Δ if and only if the state $r\varepsilon$ is reachable in Δ'. □

Therefore, our decidability result can be seen as a contribution to an automatic verification of infinite-state systems as well.

5.2 Semi-decidability of weak trace non-equivalence

Weak trace equivalence is a familiar notion which can already be found, for instance, in [21]. It is one of the semantic equivalences with a *silent* action. These equivalences are based on a notion of observable behaviour of systems: only the interactions of the system with the environment (observer) are observable. The internal structure of the system is not considered observable and system internal activities are modelled by silent (τ) actions which can precede and/or follow any observable action. For an overview of equivalences with silent moves and more general setting with respect to various testing scenarios we refer to [22]. Here we employ a more straightforward definition of weak trace equivalence, for instance see [23].

Given a labelled transition system $(S, \mathit{Act}, \longrightarrow, \alpha_0)$ with a distinguished action $\tau \in \mathit{Act}$, we define the *weak trace set* of a state $s \in S$ as

$$\mathit{wtr}(s) = \{w \in (\mathit{Act} \smallsetminus \{\tau\})^* \mid s \overset{w}{\Longrightarrow} t \text{ for some } t \in S\},$$

where $s \overset{w}{\Longrightarrow} t$ means that there is some $w' \in \mathit{Act}^*$ such that $s \overset{w'}{\longrightarrow} t$ and w is equal to w' with its τ actions deleted. Two states of a system are said to be *weak trace equivalent* if they have the same weak trace sets. It is already known that weak trace non-equivalence is semi-decidable for Petri nets (see e.g. [24]), pushdown processes (due to [20]), and PA processes (due to [13]). Before we strengthen the result to wPRS and all its subclasses, we prove an auxiliary lemma stating that the weak trace sets are recursive.

Lemma 12 *Given a wPRS Δ, its state mt, and a word $w \in \mathit{Act}(\Delta)^*$, it is decidable whether $w \in \mathit{wtr}(mt)$ or not.*

PROOF. We show that the problem can be reduced to the reachability problem. Let $\Delta = (M, \sqsubseteq, R, m_0, t_0)$ be a wPRS, mt be its state, and $w = w(0)w(1)w(2)\ldots w(k) \in (\mathit{Act} \smallsetminus \{\tau\})^+$ be a word (the case $w = \varepsilon$ is trivial as $mt \longrightarrow_\Delta^* mt$). We construct a wPRS $\Delta' = (M', \sqsubseteq', R', (m_0, 0), t_0)$, where

- $M' = \{e\} \cup M \times \{0, 1, \ldots, k\}$,
- $\sqsubseteq'$ is defined as $e \sqsubseteq' e$ and $e \sqsubseteq' (m, i)$ for all $(m, i) \in M'$, and $(n, j) \sqsubseteq' (m, i)$ for all $(m, i), (n, j) \in M'$ satisfying $n \sqsubseteq m$ and $i \leq j$,
- R' consists of the following rules:

$$
\begin{array}{lll}
(1) & (m,i)t_1 \overset{\tau}{\hookrightarrow} (n,i)t_2 & \text{for all } 0 \le i \le k \text{ and } (mt_1 \overset{\tau}{\hookrightarrow} nt_2) \in \Delta, \\
(2) & (m,i)t_1 \overset{w(i)}{\hookrightarrow} (n,i+1)t_2 & \text{for all } 0 \le i < k \text{ and } (mt_1 \overset{w(i)}{\hookrightarrow} nt_2) \in \Delta, \\
(3) & (m,k)t_1 \overset{w(k)}{\hookrightarrow} e\varepsilon & \text{for all } (mt_1 \overset{w(k)}{\hookrightarrow} nt_2) \in \Delta, \\
(4) & eX \overset{\tau}{\hookrightarrow} e\varepsilon & \text{for all } X \in \mathit{Const}(\Delta).
\end{array}
$$

Roughly speaking the second components of control states allow us to use the rewrite rules of type (1) labelled with τ while the rules of type (2) can be used only in the order given by w. According to rules (3), the transition corresponding to the last letter of w changes the control state to e. Rules (4) then allow us to rewrite the current term to ε. Hence, one can readily confirm that $w \in \mathit{wtr}(mt)$ with respect to Δ if and only if the state $e\varepsilon$ is reachable from the state $(m,0)t$ in the system Δ'. □

Theorem 13 *Weak trace non-equivalence for wPRS is semi-decidable.*

PROOF. Let mt_1 and nt_2 be states of a wPRS Δ. A semi-decidability algorithm goes through all words $w \in (\mathit{Act}(\Delta) \smallsetminus \{\tau\})^*$ and tests whether $w \in \mathit{wtr}(mt_1) \smallsetminus \mathit{wtr}(nt_2)$ or $w \in \mathit{wtr}(nt_2) \smallsetminus \mathit{wtr}(mt_1)$. The membership of w in these sets is decidable due to the previous lemma. If the algorithm finds such a word, then two given states mt_1, nt_2 are weak trace non-equivalent. Moreover, if the states are weak trace non-equivalent, the algorithm will eventually find a witness w. Hence, the weak trace non-equivalence is semi-decidable. □

To sum up, the border of the semi-decidability is moved up to the class of wPRS in the hierarchy. We emphasize that the semi-decidability result is new for classes PAN, PAD, and PRS of the original PRS hierarchy, too. As the reachability problem is undecidable for the other classes of our refined hierarchy (i.e. sePA and its superclasses), it is easy to see that the weak trace non-equivalence is not even semi-decidable for them.

5.3 Other applications

The decidability of the reachability problem for wPRS has already been used to show decidability of several other problems, in particular

- the problem whether a wPRS system contains a reachable state satisfying a given Hennessy-Milner formula [25] and
- the model checking problem for wPRS and LTL formulae with only modalities *(strict) eventually*, *(strict) always*, and their past counterparts [26].

Our decidability result has also been applied in the area of cryptographic protocols. Hüttel and Srba [27,28] define a replicative variant of a calculus for Dolev and Yao's ping-pong protocols [29]. They show that the reachability problem for their calculus is decidable as it can be reduced to the reachability problem for wPRS. We note that this application does not employ the full power of our result, as all the systems produced by the reduction belong to wPAD class.

6 Conclusions

We have shown that an extension of the Process Rewrite System mechanism with a weak finite-state control unit (wPRS) keeps the reachability problem decidable. Some applications of this result have been discussed as well.

Some related work concerning the reachability problem on PRS classes has already been mentioned in the last paragraph of Section 1.

There are several directions for the future research, including decidability and complexity issues of various problems for new subclasses. For decidability questions we note that BPP class, its two extensions, and Petri Nets form a strict (sub)hierarchy with respect to bisimulation:

$$\mathrm{BPP} \subsetneq \mathrm{wBPP} \subsetneq \mathrm{seBPP} \subsetneq \mathrm{PN}$$

We recall that bisimulation equivalence is decidable (even PSPACE-complete) for BPP processes (see [30] together with [31]) and undecidable for seBPP (as proven in [17] using Jančar's result for PN [32]). It remains open for wBPP class and this decidability borderline is a subject of our further research.

Acknowledgments. We would like to thank Jiří Srba for valuable comments, suggestions, and pointers. The authors have been supported as follows: M. Křetínský by Ministry of Education of the Czech Republic, project No. MSM0021622419, and by the Czech Science Foundation, grant No. 201/06/1389. V. Řehák by the research centre "Institute for Theoretical Computer Science (ITI)", project No. 1M0545, and by the Czech Science Foundation, grant No. 201/08/P459. J. Strejček by the Academy of Sciences of the Czech Republic, grant No. 1ET408050503, and by the Czech Science Foundation, grant No. 201/08/P375.

References

[1] M. Křetínský, V. Řehák, J. Strejček, Extended process rewrite systems: Expressiveness and reachability, in: Proceedings of CONCUR'04, Vol. 3170 of Lecture Notes in Computer Science, Springer, 2004, pp. 355–370.

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

[3] J. Esparza, Grammars as processes, in: Formal and Natural Computing, Vol. 2300 of Lecture Notes in Computer Science, Springer, 2002, pp. 277–297.

[4] O. Burkart, D. Caucal, F. Moller, B. Steffen, Verification on infinite structures, in: Handbook of Process Algebra, Elsevier, 2001, pp. 545–623.

[5] J. Srba, Roadmap of infinite results, EATCS Bulletin (78) (2002) 163–175, `http://www.brics.dk/~srba/roadmap/`.

[6] A. Bouajjani, T. Touili, Reachability Analysis of Process Rewrite Systems, in: Proc. of FST&TCS-2003, Vol. 2914 of Lecture Notes in Computer Science, Springer, 2003, pp. 74–87.

[7] A. Bouajjani, J. Esparza, T. Touili, A generic approach to the static analysis of concurrent programs with procedures, International Journal on Foundations of Computer Science 14 (4) (2003) 551–582.

[8] A. Bouajjani, R. Echahed, P. Habermehl, On the verification problem of nonregular properties for nonregular processes, in: Proc. of LICS'95, IEEE, 1995, pp. 123–133.

[9] M. Křetínský, V. Řehák, J. Strejček, On extensions of process rewrite systems: Rewrite systems with weak finite-state unit, in: Proceedings of the 5th International Workshop on Verification of Infinite-State Systems (INFINITY'03), Vol. 98 of Electronic Notes in Theoretical Computer Science, Elsevier, 2004, pp. 75–88.

[10] D. Muller, A. Saoudi, P. Schupp, Alternating automata, the weak monadic theory of trees and its complexity, Theoretical Computer Science 97 (1–2) (1992) 233–244.

[11] P. Jančar, A. Kučera, R. Mayr, Deciding bisimulation-like equivalences with finite-state processes, Theoretical Computer Science 258 (2001) 409–433.

[12] J. Strejček, Rewrite systems with constraints, in: Proc. of EXPRESS'01, Electronic Notes in Theoretical Computer Science 52 (2002).

[13] D. Lugiez, P. Schnoebelen, The regular viewpoint on PA-processes, in: Proc. of CONCUR'98, Vol. 1466 of Lecture Notes in Computer Science, Springer, 1998, pp. 50–66.

[14] J. Esparza, A. Podelski, Efficient algorithms for pre* and post* on interprocedural parallel flow graphs, in: Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POLP'00), ACM Press, 2000, pp. 1–11.

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

[16] D. Caucal, On the regular structure of prefix rewriting, Theoretical Computer Science 106 (1992) 61–86.

[17] F. Moller, Infinite results, in: Proc. of CONCUR'96, Vol. 1119 of Lecture Notes in Computer Science, Springer, 1996, pp. 195–216.

[18] Y. Hirshfeld, F. Moller, Pushdown automata, multiset automata, and Petri nets, Theoretical Computer Science 256 (1–2) (2001) 3–21. URL `http://www.elsevier.nl../22/abstract.html; http://www.elsevier.nl/gej-ng/10/41/16/198/21/22/article.pdf`

[19] E. W. Mayr, An algorithm for the general Petri net reachability problem, in: Proc. of 13th Symp. on Theory of Computing, ACM, 1981, pp. 238–246.

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

[21] C. A. R. Hoare, Communicating Sequential Processes, in: On the construction of programs – an advanced course, Cambrigde University Press, 1980, pp. 229–254.

[22] R. J. van Glabbeek, The Linear time – Branching Time Spectrum II, in: Proc. of CONCUR'93, Vol. 715 of Lecture Notes in Computer Science, Springer, 1993, pp. 66–81.

[23] P. Jančar, J. Esparza, F. Moller, Petri nets and regular behaviours, Journal of Computer and System Sciences 59 (3) (1999) 476–503.

[24] P. Jančar, High Undecidability of Weak Bisimilarity for Petri Nets, in: Proc. of TAPSOFT, Vol. 915 of Lecture Notes in Computer Science, Springer, 1995, pp. 349–363.

[25] M. Křetínský, V. Řehák, J. Strejček, Reachability of Hennessy-Milner properties for weakly extended PRS, in: R. Ramanujam, S. Sen (Eds.), FSTTCS 2005, Vol. 3821 of Lecture Notes in Computer Science, Springer-Verlag, 2005, pp. 213–224.

[26] L. Bozzelli, M. Křetínský, V. Řehák, J. Strejček, On decidability of LTL model checking for process rewrite systems, Acta Informatica 46 (1) (2009) 1–28.

[27] H. Hüttel, J. Srba, Recursion vs. replication in simple cryptographic protocols, in: Proceedings of SOFSEM 2005: Theory and Practice of Computer Science, Vol. 3381 of Lecture Notes in Computer Science, Springer, 2005, pp. 178–187.

[28] H. Hüttel, J. Srba, Decidability issues for extended ping-pong protocols, Journal of Automated Reasoning 36 (1–2) (2006) 125–147.

[29] D. Dolev, A. Yao, On the security of public key protocols, IEEE Transactions on Information Theory 29 (2) (1983) 198–208.

[30] P. Jančar, Strong bisimilarity on basic parallel processes is PSPACE-complete, in: Proc. of 18th IEEE Symposium on Logic in Computer Science (LICS'03), IEEE Computer Society, 2003, pp. 218–227.

[31] J. Srba, Strong bisimilarity and regularity of basic parallel processes is PSPACE-hard, in: Proc. of STACS 2002, Vol. 2285 of Lecture Notes in Computer Science, Springer, 2002, pp. 535–546.

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