On Decidability of LTL Model Checking for Weakly Extended Process Rewrite Systems

by

Laura Bozzelli
Mojmír Křetínský
Vojtěch Řehák
Jan Strejček

FI MU Report Series FIMU-RS-2006-05

On Decidability of LTL Model Checking for Weakly Extended Process Rewrite Systems[*]

Laura Bozzelli
Dipartimento di Matematica e Apllicazion
Università degli Studi di Napoli "Federico II"
Via Cintia, 80126 Napoli, Italy
`laura.bozzelli@dma.unina.it`

Mojmír Křetínský Vojtěch Řehák Jan Strejček
Faculty of Informatics, Masaryk University
Botanická 68a, 60200 Brno, Czech Republic
`{kretinsky,rehak,strejcek}@fi.muni.cz`

November 2, 2006

Abstract

We establish a decidability boundary of the model checking problem for infinite-state systems defined by *Process Rewrite Systems* (PRS) or *weakly extended Process Rewrite Systems* (wPRS), and properties described by basic fragments of action-based *Linear Temporal Logic* (LTL). It is known that the problem for general LTL properties is decidable for Petri nets and for pushdown processes, while it is undecidable for PA processes. As our main result, we show that the problem is decidable for wPRS if we consider properties defined by formulae with only modalities *strict eventually* and *strict always*. Moreover, we show that the problem remains undecidable for PA processes even with respect to the LTL fragment with the only modality *until* or the fragment with modalities *next* and *infinitely often*.

[*]This is a full version of the paper [BKŘS06] accepted for FSTTCS 2006.

1 Introduction

Automatic verification of current software systems often needs to model them as infinite-state systems. One of the most powerful formalisms for description of infinite-state systems (except formalisms with Turing power for which nearly all interesting verification problems are undecidable) is called *Process Rewrite Systems* (PRS) [May00]. The PRS framework, based on term rewriting, subsumes many formalisms studied in the context of formal verification, e.g. *Petri nets* (PN), *pushdown processes* (PDA), and process algebras like BPA, BPP, or PA. PRS can be adopted as a formal model for programs with recursive procedures and restricted forms of dynamic creation and synchronization of concurrent processes. A substantial advantage of PRS is that some important verification problems are decidable for the whole PRS class. In particular, Mayr [May00] proved that the *reachability problem* (whether a given state is reachable) and the *reachable property problem* (whether there is a reachable state where some given actions are enabled and some given actions are disabled) are decidable for PRS.

In [KŘS04b], we have presented *weakly extended PRS* (wPRS), where a finite-state control unit with self-loops as the only loops is added to the standard PRS formalism (addition of a general finite-state control unit makes PRS Turing powerful). This control unit enriches PRS by abilities to model a bounded number of arbitrary communication events and global variables whose values are changed only a bounded number of times during any computation. We have proved that the reachability problem remains decidable for wPRS [KŘS04a] and that the problem called *reachability Hennessy–Milner property* (whether there is a reachable state satisfying a given Hennessy–Milner formula) is decidable for wPRS as well [KŘS05]. The hierarchy of all PRS and wPRS classes is depicted in Figure 1.

Concerning the model checking problem, a broad overview of (un)decidability results for subclasses of PRS and various temporal logics can be found in [May98]. Here we focus exclusively on (future) *Linear Temporal Logic* (LTL). It is known that LTL model checking of PDA is EXPTIME-complete [BEM97]. LTL model checking of PN is also decidable, but at least as hard as the reachability problem for PN [Esp94] (the reachability problem is EXPSPACE-hard [May84, Lip76] and no primitive recursive upper bound is known). If we consider only infinite runs, then the problem for PN is EXPSPACE-complete [Hab97, May98].

Conversely, LTL model checking is undecidable for all classes subsuming PA [BH96, May98]. So far, there are only two positive results for these classes. Bouajjani and Habermehl [BH96] have identified a fragment called *simple PLTL$_\Box$* for which model checking of infinite runs is decidable for PA (strictly speaking, simple PLTL$_\Box$ is not a fragment of LTL as it can express also some non-regular properties, while LTL cannot). Only recently, we have demonstrated that model checking of infinite runs is decidable for PRS and the fragment of LTL capturing exactly fairness properties [Boz05].

Our contribution: This paper completely locates the decidability boundary of the model checking problem for all subclasses of PRS (and wPRS) and all *basic LTL fragments*, where a basic LTL fragment is a set of all formulae containing only a given subset of standard modalities. The boundary is depicted in Figure 2. To locate the boundary, we show the following results.

1. We introduce a new LTL fragment $\mathcal{A}$ and prove that every formula of the basic fragment LTL($\mathsf{F_s}, \mathsf{G_s}$) (i.e. the fragment with modalities *strict eventually* and *strict always* only) can be effectively translated into $\mathcal{A}$. As LTL($\mathsf{F_s}, \mathsf{G_s}$) is closed under negation, we can also translate LTL($\mathsf{F_s}, \mathsf{G_s}$) formulae into negated formulae of $\mathcal{A}$.

2. We show that model checking (of both finite and infinite runs) of wPRS against negated formulae of $\mathcal{A}$ is decidable. The proof employs our results presented in [Boz05, KŘS04a, KŘS05] to reduce the problem to LTL model checking for PDA and PN. Thus we get decidability of model checking for wPRS against LTL($\mathsf{F_s}, \mathsf{G_s}$). Note that LTL($\mathsf{F_s}, \mathsf{G_s}$) is strictly more expressive than the *Lamport logic* (i.e. the basic fragment with modalities *eventually* and *always*), which is again strictly more expressive than the mentioned fragment of fairness properties and also than the *regular* part of simple PLTL$_\Box$.

3. We demonstrate that the model checking problem remains undecidable for PA even if we consider the basic fragment with modality *until* or the basic fragment with modalities *next* and *infinitely often* (which is strictly less expressive than the one with *next* and *eventually*).

The paper is organized as follows. The following section recalls basic definitions. Sections 3, 4, and 5 correspond, respectively, to the three results listed above. The last section discuss other potential applications of our results and it contains an open question driving our future research.

2 Preliminaries

2.1 PRS and its extensions

Let *Const* $= \{X, \dots\}$ be a set of *process constants*. The set of *process terms* t is defined by the abstract syntax $t ::= \varepsilon \mid X \mid t.t \mid t\|t$, where ε is the *empty term*, $X \in$ *Const*, and '.' and '$\|$' mean *sequential* and *parallel compositions*, respectively. We always work with equivalence classes of terms modulo commutativity and associativity of '$\|$', associativity of '.', and neutrality of ε, i.e. $\varepsilon.t = t.\varepsilon = t\|\varepsilon = t$. We distinguish four *classes of process terms* as:

1 – terms consisting of a single process constant, 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 without any restrictions, e.g. $(X.(Y\|Z))\|W$.

Let $M = \{o, p, q, \dots\}$ be a set of *control states*, $\leq$ be a partial ordering on this set, and *Act* $= \{a, b, c, \dots\}$ be a set of *actions*. Let $\alpha, \beta \in \{1, S, P, G\}$ be classes of process terms such that $\alpha \subseteq \beta$. An (α, β)-*wPRS* (*weakly extended process rewrite system*) Δ is a tuple (R, p_0, X_0), where

- R is a finite set of *rewrite rules* of the form $(p, t_1) \overset{a}{\hookrightarrow} (q, t_2)$, where $t_1 \in \alpha$, $t_1 \neq \varepsilon$, $t_2 \in \beta$, $a \in$ *Act*, and $p, q \in M$ are control states satisfying $p \leq q$,
- the pair $(p_0, X_0) \in M \times$ *Const* forms the distinguished *initial state*.

By *Act*(Δ), *Const*(Δ), and $M(\Delta)$ we denote the sets of actions, process constants, and control states occurring in the rewrite rules or the initial state of Δ, respectively.

An (α, β)-wPRS $\Delta = (R, p_0, X_0)$ induces a labelled transition system, whose states are pairs (p, t) such that $p \in M(\Delta)$ is a control state and $t \in \beta$ is a process term over *Const*(Δ). The transition relation $\longrightarrow_\Delta$ is the least relation satisfying the following inference rules:

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

Sometimes we write $\longrightarrow$ instead of $\longrightarrow_\Delta$ if Δ is clear from the context. The transition relation can be extended to finite words over *Act* in a standard way. To shorten our notation we write pt in lieu of (p, t). A state pt is called *terminal*, written $pt \not\longrightarrow_\Delta$, if there

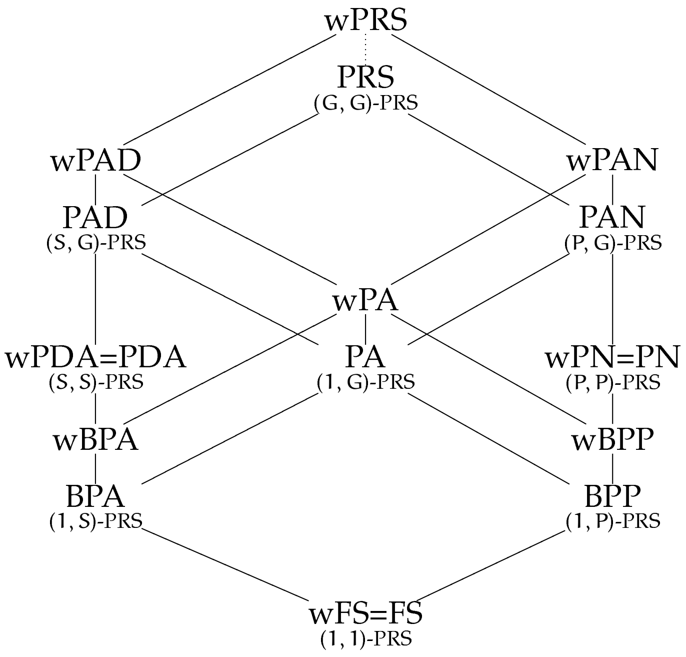

Figure 1: The hierarchy of PRS and wPRS subclasses.

is no state $p't'$ and action a such that $pt \xrightarrow{a}_{\Delta} p't'$. In this paper we always consider only systems where the initial state is not terminal. A (finite or infinite) sequence

$$\sigma = p_1t_1 \xrightarrow{a_1}_{\Delta} p_2t_2 \xrightarrow{a_2}_{\Delta} \dots \xrightarrow{a_n}_{\Delta} p_{n+1}t_{n+1} \left(\xrightarrow{a_{n+1}}_{\Delta} \dots \right)$$

is called *derivation over the word* $u = a_1a_2 \dots a_n(a_{n+1} \dots)$ *in* Δ. Finite derivations are also denoted as $p_1t_1 \xrightarrow{u}_{\Delta} p_{n+1}t_{n+1}$, infinite as $p_1t_1 \xrightarrow{u}_{\Delta}$. A derivation in Δ is called a *run of* Δ if it starts in the initial state p_0X_0 and it is either infinite, or its last state is terminal. Further, $L(\Delta)$ denotes the set of words u such that there is a run of Δ over u.

An (α, β)-wPRS Δ where $M(\Delta)$ is a singleton is called (α, β)*-PRS* (*process rewrite system*) [May00]. In such systems we omit the single control state from rules and states.

Some classes of (α, β)-PRS correspond to widely known models, namely *finite-state systems* (FS), *basic process algebras* (BPA), *basic parallel processes* (BPP), *process algebras* (PA), *pushdown processes* (PDA), and *Petri nets* (PN). The other classes have been named as PAD, PAN, and PRS. The relations between (α, β)-PRS and the mentioned formalisms and names are indicated in Figure 1. Instead of (α, β)-wPRS we juxtapose the prefix 'w-' with the acronym corresponding to the (α, β)-PRS class. For example, we use wBPA rather than $(1, S)$-wPRS. Figure 1 shows the expressiveness hierarchy of all considered classes, where expressive power of a class is measured by the set of transition systems that are definable (up to the strong bisimulation equivalence [Mil89]) by the class. This hierarchy is strict, with a potential exception concerning the classes wPRS and PRS, where the strictness is just our conjecture. For details see [KŘS04b, KŘS04a].

For technical reasons, we define a normal form of wPRS systems. A rewrite rule is *parallel* or *sequential* if it has one of the following forms:

$$\begin{array}{rl} \textbf{Parallel rules:} & pX_1\|X_2\|\ldots\|X_n \overset{a}{\hookrightarrow} qY_1\|Y_2\|\ldots\|Y_m \\ \textbf{Sequential rules:} & pX \overset{a}{\hookrightarrow} qY.Z \quad pX.Y \overset{a}{\hookrightarrow} qZ \quad pX \overset{a}{\hookrightarrow} qY \quad pX \overset{a}{\hookrightarrow} q\varepsilon \end{array}$$

where $X, Y, X_i, Y_j, Z \in \mathit{Const}$, $p, q \in M$, $n > 0$, $m \geq 0$, and $a \in \mathit{Act}$. A rule is called *trivial* if it is both parallel and sequential (i.e. it has the form $pX \overset{a}{\hookrightarrow} qY$ or $pX \overset{a}{\hookrightarrow} q\varepsilon$). A wPRS Δ is in *normal form* if it has only parallel and sequential rewrite rules.

2.2 Linear Temporal Logic (LTL) and studied problems

The syntax of *Linear Temporal Logic* (LTL) [Pnu77] is defined as follows

$$\varphi ::= \mathit{tt} \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi\,\mathsf{U}\,\varphi,$$

where a ranges over *Act*, X is called *next*, and U is called *until*. The logic is interpreted over infinite as well as nonempty finite words of actions. Given a word $u = u(0)u(1)u(2)\ldots \in \mathit{Act}^*\cup\mathit{Act}^\omega$, $|u|$ denotes the length of the word (we set $|u| = \infty$ if u is infinite). For all $0 \leq i < |u|$, by u_i we denote the i^{th} suffix of u, i.e. $u_i = u(i)u(i+1)\ldots$.

The semantics of LTL formulae is defined inductively as follows:

$$\begin{array}{lll} u \models \mathit{tt} & & \\ u \models a & \text{iff} & u(0) = a \\ u \models \neg\varphi & \text{iff} & u \not\models \varphi \\ u \models \varphi_1 \wedge \varphi_2 & \text{iff} & u \models \varphi_1 \text{ and } u \models \varphi_2 \\ u \models \mathsf{X}\varphi & \text{iff} & |u| > 1 \text{ and } u_1 \models \varphi \\ u \models \varphi_1\,\mathsf{U}\,\varphi_2 & \text{iff} & \exists 0 \leq i < |u| \,.\, (\, u_i \models \varphi_2 \text{ and } \forall\, 0 \leq j < i \,.\, u_j \models \varphi_1\,) \end{array}$$

We say that a nonempty word u *satisfies* φ whenever $u \models \varphi$. Given a set of words L, we write $L \models \varphi$ if $u \models \varphi$ holds for all $u \in L$. We say that a derivation (or run) σ over a word u satisfies φ, written $\sigma \models \varphi$, whenever $u \models \varphi$.

Moreover, we define the following modalities: $\mathsf{F}\varphi$ (*eventually*) standing for $\mathit{tt}\,\mathsf{U}\,\varphi$, $\mathsf{G}\varphi$ (*always*) standing for $\neg\mathsf{F}\neg\varphi$, $\mathsf{F_s}\varphi$ (*strict eventually*) standing for $\mathsf{XF}\varphi$, $\mathsf{G_s}\varphi$ (*strict always*) standing for $\neg\mathsf{F_s}\neg\varphi$, $\overset{\infty}{\mathsf{F}}\varphi$ (*infinitely often*) standing for $\mathsf{GF}\varphi$, $\overset{\infty}{\mathsf{G}}\varphi$ (*almost always*) standing for $\neg\overset{\infty}{\mathsf{F}}\neg\varphi$. Note that $\mathsf{F}\varphi$ is equivalent to $\varphi \vee \mathsf{F_s}\varphi$ but $\mathsf{F_s}\varphi$ cannot be expressed

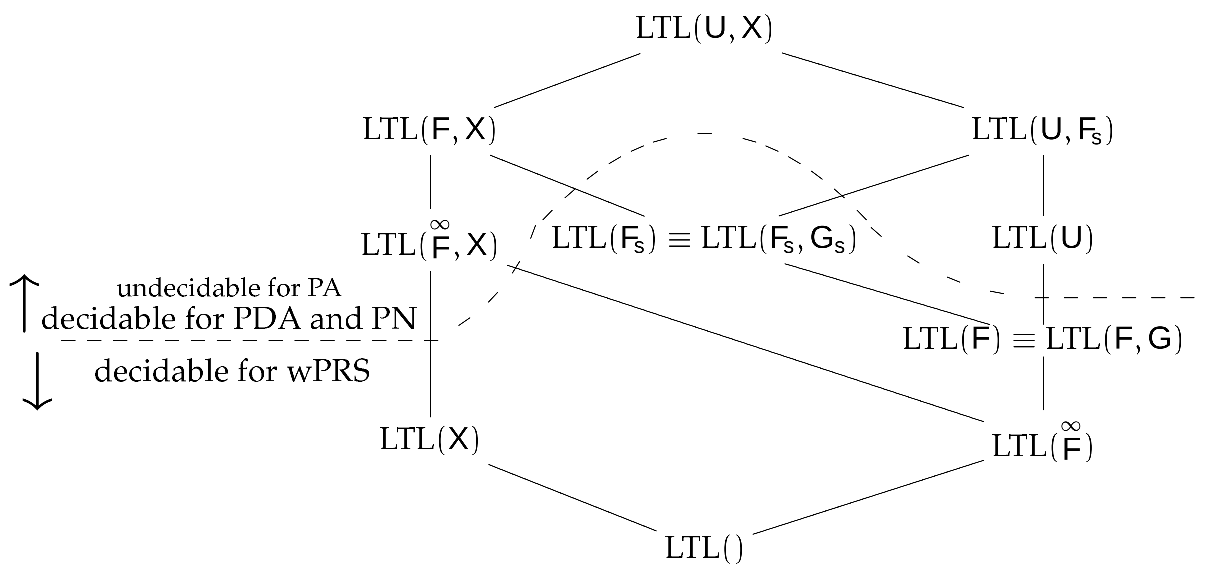

Figure 2: The hierarchy of basic fragments with model checking decidability boundary.

with F as the only modality. Thus $\mathsf{F_s}$ is "stronger" than F. The relation between $\mathsf{G_s}$ and G is similar.

For a set $\{O_1, \ldots, O_n\}$ of modalities, $\mathrm{LTL}(O_1, \ldots, O_n)$ denotes the LTL fragment containing all formulae with modalities $O_1, \ldots, O_n$ only. Such a fragment is called *basic*. Figure 2 shows an expressiveness hierarchy of all studied basic LTL fragments. Indeed, every basic LTL fragment using standard[1] future modalities is equivalent to one of the fragments in the hierarchy, where equivalence between fragments means that every formula of one fragment can be effectively translated into a semantically equivalent formula of the other fragment and vice versa. For example, $\mathrm{LTL}(\mathsf{F_s}, \mathsf{G_s}) \equiv \mathrm{LTL}(\mathsf{F_s})$. Further, the hierarchy is strict. For detailed information about expressiveness of future LTL modalities and LTL fragments we refer to [Str04].

Let $\mathcal{F}$ be an LTL fragment and $\mathcal{C}$ be a class of wPRS systems. The *model checking problem* for $\mathcal{F}$ and $\mathcal{C}$ is to decide whether a given formula $\varphi \in \mathcal{F}$ and a given system $\Delta \in \mathcal{C}$ satisfies $L(\Delta) \models \varphi$. We also mention the problem called *model checking of infinite runs*, where $L(\Delta) \cap \mathit{Act}^\omega \models \varphi$ is examined.

[1]By standard modalities we mean the ones defined in this paper and also other commonly used modalities like *strict until*, *release*, *weak until*, etc. However, it is well possible that one can define a new modality such that there is a basic fragment not equivalent to any of the fragments in the hierarchy.

3 Fragment $\mathcal{A}$ and translation of LTL($\mathsf{F_s}, \mathsf{G_s}$) into $\mathcal{A}$

The $\mathcal{A}$ fragment consists of finite disjunctions of *α-formulae* defined as follows.

Recall that LTL() denotes the fragment of formulae without any modality, i.e. boolean combinations of actions. In the following we use $\varphi_1 \mathsf{U}_+ \varphi_2$ to abbreviate $\varphi_1 \wedge \mathsf{X}(\varphi_1 \mathsf{U} \varphi_2)$. Let $\delta = \theta_1 O_1 \theta_2 O_2 \ldots \theta_n O_n \theta_{n+1}$, where $n > 0$, each $\theta_i \in$ LTL(), O_n is '$\wedge \mathsf{G_s}$', and, for each $i < n$, O_i is either 'U' or 'U_+' or '$\wedge \mathsf{X}$'. Further, let $\mathcal{B} \subseteq$ LTL() be a finite set. An α-formula is defined as

$$\alpha(\delta, \mathcal{B}) = \big(\theta_1 O_1 (\theta_2 O_2 \ldots (\theta_n O_n \theta_{n+1}) \ldots)\big) \wedge \bigwedge_{\psi \in \mathcal{B}} \mathsf{G_s F_s} \psi$$

Hence, a word u satisfies $\alpha(\delta, \mathcal{B})$ iff u can be written as $u_1.u_2. \cdots .u_{n+1}$, where

- each u_i consists only of actions satisfying θ_i and
  - $|u_i| \geq 0$ if $i = n+1$ or O_i is 'U',
  - $|u_i| > 0$ if O_i is 'U_+',
  - $|u_i| = 1$ if O_i is '$\wedge \mathsf{X}$' or '$\wedge \mathsf{G_s}$',
- and u_{n+1} satisfies $\mathsf{G_s F_s} \psi$ for every $\psi \in \mathcal{B}$.

Proof of the following lemma is a simple exercise.

Lemma 3.1. *A conjunction of α-formulae can be effectively converted into an equivalent disjunction of α-formulae.*

Theorem 3.2. *Every LTL($\mathsf{F_s}, \mathsf{G_s}$) formula can be translated into an equivalent disjunction of α-formulae.*

Proof. As $\mathsf{F_s}$ and $\mathsf{G_s}$ are dual modalities, we can assume that every LTL($\mathsf{F_s}, \mathsf{G_s}$) formula contains negations only in front of actions. Given an LTL($\mathsf{F_s}, \mathsf{G_s}$) formula φ, we construct a finite set A_φ of α-formulae such that φ is equivalent to disjunction of formulae in A_φ. Although our proof looks like induction on structure of φ, it is in fact induction on the length of φ. Thus, if $\varphi \notin$ LTL(), then we assume that for every LTL($\mathsf{F_s}, \mathsf{G_s}$) formula φ' shorter than φ we can construct the corresponding set $A_{\varphi'}$. In this proof, p represents a formula of LTL(). The structure of φ fits into one of the following cases.

•p **Case** p**:** In this case, φ is eqvivalent to $p \wedge \mathsf{G_s} tt$. Hence $A_\varphi = \{\alpha(p \wedge \mathsf{G_s} tt, \emptyset)\}$.

- $\bullet\vee$ **Case** $\varphi_1 \vee \varphi_2$**:** Due to induction hypothesis, we can assume that we have sets A_{φ_1} and A_{φ_2}. Clearly, $A_\varphi = A_{\varphi_1} \cup A_{\varphi_2}$.
- $\bullet\wedge$ **Case** $\varphi_1 \wedge \varphi_2$**:** Due to Lemma 3.1, the set A_φ can be constructed from the sets A_{φ_1} and A_{φ_2}.
- $\bullet\mathsf{F_s}$ **Case** $\mathsf{F_s}\varphi_1$**:** As $\mathsf{F_s}(\alpha_1 \vee \alpha_2) \equiv (\mathsf{F_s}\alpha_1) \vee (\mathsf{F_s}\alpha_2)$ and $\mathsf{F_s}(\alpha \wedge \mathsf{G_s}\mathsf{F_s}\phi) \equiv (\mathsf{F_s}\alpha) \wedge (\mathsf{G_s}\mathsf{F_s}\phi)$, we set $A_\varphi = \{\alpha(tt\,\mathsf{U}_+\,\delta, \mathcal{B}) \mid \alpha(\delta, \mathcal{B}) \in A_{\varphi_1}\}$.
- $\bullet\mathsf{G_s}$ **Case** $\mathsf{G_s}\varphi_1$**:** This case is divided into the following subcases according to the structure of φ_1.
  - $\circ p$ **Case** $\mathsf{G_s}p$**:** As $\mathsf{G_s}p$ is equivalent to $tt \wedge \mathsf{G_s}p$, we set $A_\varphi = \{\alpha(tt \wedge \mathsf{G_s}p, \emptyset)\}$.
  - $\circ\wedge$ **Case** $\mathsf{G_s}(\varphi_2 \wedge \varphi_3)$**:** As $\mathsf{G_s}(\varphi_2 \wedge \varphi_3) \equiv (\mathsf{G_s}\varphi_2) \wedge (\mathsf{G_s}\varphi_3)$, the set A_φ can be constructed from $A_{\mathsf{G_s}\varphi_2}$ and $A_{\mathsf{G_s}\varphi_3}$ using Lemma 3.1. Note that $A_{\mathsf{G_s}\varphi_2}$ and $A_{\mathsf{G_s}\varphi_3}$ can be constructed because $\mathsf{G_s}\varphi_2$ and $\mathsf{G_s}\varphi_3$ are shorter than $\mathsf{G_s}(\varphi_2 \wedge \varphi_3)$.
  - $\circ\mathsf{F_s}$ **Case** $\mathsf{G_s}\mathsf{F_s}\varphi_2$**:** This case is again divided into the following subcases.
    - $-p$ **Case** $\mathsf{G_s}\mathsf{F_s}p$**:** As $p \in \mathrm{LTL}()$, we directly set $A_\varphi = \{\alpha(tt \wedge \mathsf{G_s}tt, \{p\})\}$.
    - $-\vee$ **Case** $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \vee \varphi_4)$**:** As $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \vee \varphi_4) \equiv (\mathsf{G_s}\mathsf{F_s}\varphi_3) \vee (\mathsf{G_s}\mathsf{F_s}\varphi_4)$, we set $A_\varphi = A_{\mathsf{G_s}\mathsf{F_s}\varphi_3} \cup A_{\mathsf{G_s}\mathsf{F_s}\varphi_4}$.
    - $-\wedge$ **Case** $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \varphi_4)$**:** This case is also divided into subcases depending on the formulae φ_3 and φ_4.
      - $*p$ **Case** $\mathsf{G_s}\mathsf{F_s}(p_3 \wedge p_4)$**:** As $p_3 \wedge p_4 \in \mathrm{LTL}()$, this subcase has already been covered by Case $\mathsf{G_s}\mathsf{F_s}p$.
      - $*\vee$ **Case** $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge (\varphi_5 \vee \varphi_6))$**:** As $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge (\varphi_5 \vee \varphi_6)) \equiv \mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \varphi_5) \vee \mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \varphi_6)$, we set $A_\varphi = A_{\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \varphi_5)} \cup A_{\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \varphi_6)}$.
      - $*\mathsf{F_s}$ **Case** $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \mathsf{F_s}\varphi_5)$**:** As $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \mathsf{F_s}\varphi_5) \equiv (\mathsf{G_s}\mathsf{F_s}\varphi_3) \wedge (\mathsf{G_s}\mathsf{F_s}\varphi_5)$, the set A_φ can be constructed from $A_{\mathsf{G_s}\mathsf{F_s}\varphi_3}$ and $A_{\mathsf{G_s}\mathsf{F_s}\varphi_5}$ using Lemma 3.1.
      - $*\mathsf{G_s}$ **Case** $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \mathsf{G_s}\varphi_5)$: As $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \mathsf{G_s}\varphi_5) \equiv (\mathsf{G_s}\mathsf{F_s}\varphi_3) \wedge (\mathsf{G_s}\mathsf{F_s}\mathsf{G_s}\varphi_5)$, the set A_φ can be constructed from $A_{\mathsf{G_s}\mathsf{F_s}\varphi_3}$ and $A_{\mathsf{G_s}\mathsf{F_s}\mathsf{G_s}\varphi_5}$ using Lemma 3.1.
    - $-\mathsf{F_s}$ **Case** $\mathsf{G_s}\mathsf{F_s}\mathsf{F_s}\varphi_3$**:** As $\mathsf{G_s}\mathsf{F_s}\mathsf{F_s}\varphi_3 \equiv \mathsf{G_s}\mathsf{F_s}\varphi_3$, we set $A_\varphi = A_{\mathsf{G_s}\mathsf{F_s}\varphi_1}$.
    - $-\mathsf{G_s}$ **Case** $\mathsf{G_s}\mathsf{F_s}\mathsf{G_s}\varphi_3$**:** A word u satisfies $\mathsf{G_s}\mathsf{F_s}\mathsf{G_s}\varphi_3$ iff $|u| = 1$ or u is an infinite word satisfying $\mathsf{F_s}\mathsf{G_s}\varphi_3$. Note that $\mathsf{G_s}\neg tt$ is satisifed only by finite words

of length one. Further, a word u satisfies $(\mathsf{F_s}tt) \wedge (\mathsf{G_s F_s}tt)$ iff u is infinite. Thus, $\mathsf{G_s F_s G_s}\varphi_3 \equiv (\mathsf{G_s}\neg tt) \vee \varphi'$ where $\varphi' = (\mathsf{F_s}tt) \wedge (\mathsf{G_s F_s}tt) \wedge (\mathsf{F_s G_s}\varphi_3)$. Hence, $A_\varphi = A_{\mathsf{G_s}\neg tt} \cup A_{\varphi'}$ where $A_{\varphi'}$ is constructed from $A_{\mathsf{F_s}tt}$, $A_{\mathsf{G_s F_s}tt}$, and $A_{\mathsf{F_s G_s}\varphi_3}$ using Lemma 3.1.

$\circ\vee$ **Case** $\mathsf{G_s}(\varphi_2 \vee \varphi_3)$: According to the structure of φ_2 and φ_3, there are the following subcases.

$\star p$ **Case** $\mathsf{G_s}(p_2 \vee p_3)$: As $p_2 \vee p_3 \in$ LTL(), this subcase has already been covered by Case $\mathsf{G_s}p$.

$\star\wedge$ **Case** $\mathsf{G_s}(\varphi_2 \vee (\varphi_4 \wedge \varphi_5))$: As $\mathsf{G_s}(\varphi_2 \vee (\varphi_4 \wedge \varphi_5)) \equiv \mathsf{G_s}(\varphi_2 \vee \varphi_4) \wedge \mathsf{G_s}(\varphi_2 \vee \varphi_5)$, the set A_φ can be constructed from $A_{\mathsf{G_s}(\varphi_2 \vee \varphi_4)}$ and $A_{\mathsf{G_s}(\varphi_2 \vee \varphi_5)}$ using Lemma 3.1.

$\star\mathsf{F_s}$ **Case** $\mathsf{G_s}(\varphi_2 \vee \mathsf{F_s}\varphi_4)$: It holds that $\mathsf{G_s}(\varphi_2 \vee \mathsf{F_s}\varphi_4) \equiv (\mathsf{G_s}\varphi_2) \vee \mathsf{F_s}(\varphi_4 \wedge \varphi_2 \wedge \mathsf{G_s}\varphi_2) \vee \mathsf{G_s F_s}\varphi_4$. Therefore, the set A_φ can be constructed as $A_{\mathsf{G_s}\varphi_2} \cup \{\alpha(tt\,\mathsf{U_+}\,\delta, \mathcal{B}) \mid \alpha(\delta, \mathcal{B}) \in A_{\varphi_4 \wedge \varphi_2 \wedge \mathsf{G_s}\varphi_2}\} \cup A_{\mathsf{G_s F_s}\varphi_4}$, where $A_{\varphi_4 \wedge \varphi_2 \wedge \mathsf{G_s}\varphi_2}$ is composed form A_{φ_4}, A_{φ_2}, and $A_{\mathsf{G_s}\varphi_2}$ due to Lemma 3.1.

$\star\mathsf{G_s}$ **Case** $\mathsf{G_s}(\varphi_2 \vee \mathsf{G_s}\varphi_4)$: There are only the following two subcases (the others fit to some of the previous cases).

(i) **Case** $\mathsf{G_s}(\bigvee_{\varphi' \in G} \mathsf{G_s}\varphi')$: It holds that $\mathsf{G_s}(\bigvee_{\varphi' \in G} \mathsf{G_s}\varphi') \equiv (\mathsf{G_s}\neg tt) \vee \bigvee_{\varphi' \in G}(\mathsf{X G_s}\varphi')$. Therefore, the set A_φ can be constructed as $A_{\mathsf{G_s}\neg tt} \cup \bigcup_{\varphi' \in G}\{\alpha(tt \wedge \mathsf{X}\delta, \mathcal{B}) \mid \alpha(\delta, \mathcal{B}) \in A_{\mathsf{G_s}\varphi'}\}$.

(ii) **Case** $\mathsf{G_s}(p_2 \vee \bigvee_{\varphi_1 \in G} \mathsf{G_s}\varphi_1)$: As $\mathsf{G_s}(p_2 \vee \bigvee_{\varphi' \in G} \mathsf{G_s}\varphi') \equiv (\mathsf{G_s}p_2) \vee \bigvee_{\varphi' \in G}(\mathsf{X}(p_2\,\mathsf{U}\,\mathsf{G_s}\varphi'))$. Therefore, the set A_φ can be constructed as $A_{\mathsf{G_s}p_2} \cup \bigcup_{\varphi' \in G}\{\alpha(tt \wedge \mathsf{X}p_2\,\mathsf{U}\,\delta, \mathcal{B}) \mid \alpha(\delta, \mathcal{B}) \in A_{\mathsf{G_s}\varphi'}\}$.

$\circ\mathsf{G_s}$ **Case** $\mathsf{G_s}(\mathsf{G_s}\varphi_2)$: As $\mathsf{G_s}(\mathsf{G_s}\varphi_2) \equiv (\mathsf{G_s}\neg tt) \vee (\mathsf{X G_s}\varphi_2)$, the set A_φ can be constructed as $A_{\mathsf{G_s}\neg tt} \cup \{\alpha(tt \wedge \mathsf{X}\delta, \mathcal{B}) \mid \alpha(\delta, \mathcal{B}) \in A_{\mathsf{G_s}\varphi_2}\}$.

□

4 Model checking of wPRS against negated $\mathcal{A}$

This section is devoted to decidability of the model checking problem for wPRS and negated formulae of the $\mathcal{A}$ fragment. In fact, we prove decidability of the dual problem, i.e. whether a given wPRS system has a run satisfying a given formula of $\mathcal{A}$. Finite and infinite runs are treated separately.

Theorem 4.1. *The problem whether a given wPRS system has a finite run satisfying a given α-formula is decidable.*

Proof. Let Δ be a wPRS system and $\alpha(\delta, \mathcal{B})$ be an α-formula. Note that a formula $\mathsf{G_s F_s}\psi$ is valid on a finite nonempty word if and only if the length of the word is 1. Therefore, if $\mathcal{B} \neq \emptyset$ then it is easy to check whether there is a finite run of Δ satisfying $\alpha(\delta, \mathcal{B})$. In what follows we assume $\mathcal{B} = \emptyset$.

Let $\delta = \theta_1 O_1 \theta_2 O_2 \ldots \theta_n O_n \theta_{n+1}$. We construct a wPRS system Δ' with control states $M(\Delta) \times \{1, 2, \ldots, n+1\}$ in the following way.

- For any $1 \leq i \leq n$ and every rule $pt_1 \overset{a}{\hookrightarrow} qt_2$ of Δ such that a satisfies θ_i, we add to Δ' the rule $(p,i)t_1 \overset{a}{\hookrightarrow} (q, i+1)t_2$ and if O_i is U or U_+ then also the rule $(p,i)t_1 \overset{a}{\hookrightarrow} (q,i)t_2$.

- For every $p \in M(\Delta)$, $X \in \mathit{Const}(\Delta)$, and for any $1 \leq i \leq n$ such that $O_i = \mathsf{U}$, we add to Δ' the rule $(p,i)X \overset{e}{\hookrightarrow} (p, i+1)X$, where e is an arbitrary action.

- For every rule $pt_1 \overset{a}{\hookrightarrow} qt_2$ of Δ such that a satisfies θ_{n+1}, we add to Δ' the rule $(p, n+1)t_1 \overset{a}{\hookrightarrow} (q, n+1)t_2$.

- For every rule $pt_1 \overset{a}{\hookrightarrow} qt_2$ of Δ we add to Δ' the rule $(p, n+1)t_1 \overset{a}{\hookrightarrow} (p, n+1)t_1$.

Let p_0X_0 be the initial state of Δ. There is a finite run $p_0X_0 \xrightarrow{u}_\Delta qt$ satisfying $\alpha(\delta, \emptyset)$ if and only if there is a finite run $(p_0, 1)X_0 \xrightarrow{v}_{\Delta'} (q, n+1)t$. Hence, we need to decide whether there exists a state of the form $(q, n+1)t$ that is terminal and reachable from $(p_0, 1)X_0$. To do that, for every $p \in M(\Delta)$ we add to Δ' the rule $(p, n+1)Z \overset{\mathit{end}}{\hookrightarrow} (p, n+1)\varepsilon$, where $\mathit{end} \notin \mathit{Act}(\Delta)$ is a fresh action and $Z \notin \mathit{Const}(\Delta)$ is a fresh process constant. Now, it holds that Δ has a finite run satisfying $\alpha(\delta, \emptyset)$ if and only if there exists a state of Δ', which is reachable from $(p_0, 1)(X_0 \| Z)$ and the only enabled action in this state is *end*. This last condition on the state can be expressed by formula $\varphi = \langle \mathit{end} \rangle \mathit{tt} \wedge \bigwedge_{a \in \mathit{Act}(\Delta)} \neg \langle a \rangle \mathit{tt}$ of the Hennessy–Milner logic. As reachability of a state satisfying a given Hennessy–Milner formula is decidable for wPRS (see [KŘS05] for details), we are done. $\square$

The problem for infinite runs is more complicated. In order to solve it, we introduce more terminology and notation. First we define β-*formulae* and regular languages called γ-*languages*. Let $w = a_1 O_1 a_2 O_2 \ldots a_n O_n$, where $n \geq 0$, $a_1, \ldots, a_n \in \mathit{Act}$ are pairwise distinct actions and each O_i is either 'U_+' or '$\wedge \mathsf{X}$'. Further, let $B \subseteq \mathit{Act} \smallsetminus \{a_1, \ldots, a_n\}$

be a nonempty finite set of actions and $C \subseteq B$. A β-formula $\beta(w, B, C)$ and γ-language $\gamma(w, C)$ are defined as

$$\beta(w, B, C) = \big(a_1 O_1 (a_2 O_2 \ldots (a_n O_n \mathsf{G} \bigvee_{b \in B} b) \ldots)\big) \wedge \bigwedge_{b \in C} \mathsf{GF} b \wedge \bigwedge_{b \in B \smallsetminus C} (\mathsf{F} b \wedge \neg \mathsf{GF} b)$$

$$\gamma(w, C) = a_1^{o_1}.a_2^{o_2}. \cdots .a_n^{o_n}.L,$$

$$\text{where } o_i = \begin{cases} + & \text{if } O_i = \mathsf{U}_+ \\ 1 & \text{if } O_i = \wedge \mathsf{X} \end{cases} \quad \text{and } L = \begin{cases} \{\varepsilon\} & \text{if } C = \emptyset \\ \bigcap_{b \in C} C^*.b.C^* & \text{otherwise} \end{cases}$$

Roughly speaking, a β-formula is a more restrictive version of an α-formula and in context of β-formulae we consider infinite words only. Contrary to δ of an α-formula, w of a β-formula employs actions rather than LTL() formulae. While a tail of an infinite word satisfying an α-formula is specified by θ_{n+1}, in the definition of β-formulae we use a set B containing exactly all the actions of the tail and its subset C of exactly all actions occurring infinitely many times in the tail.

Remark 4.2. *Note that an infinite word satisfies a formula $\beta(w, B, C)$ if and only if it can be divided into a prefix $u \in \gamma(w, B)$ and a suffix $v \in C^\omega$ such that v contains infinitely many occurrences of every $c \in C$.*

Let w, B, C be defined as above. We say that a finite derivation σ over a word u *satisfies* $\gamma(w, C)$ if and only if $u \in \gamma(w, C)$. We write $(w', B') \sqsubseteq (w, B)$ whenever $B' \subseteq B$ and $w' = a_{i_1} O_{i_1} a_{i_2} O_{i_2} \ldots a_{i_k} O_{i_k}$ for some $1 \le i_1 < i_2 < \ldots < i_k \le n$. Moreover, we write $(w', B', C') \sqsubseteq (w, B, C)$ whenever $(w', B') \sqsubseteq (w, B)$, B' is nonempty, and $C' \subseteq C \cap B'$.

Remark 4.3. *If u is an infinite word satisfying $\beta(w, B, C)$ and v is an infinite* subword *of u (i.e. it arises from u by omitting some letters), then there is exactly one triple $(w', B', C') \sqsubseteq (w, B, C)$ such that $v \models \beta(w', B', C')$. Further, for each finite subword v of u, there is exactly one pair (w', B') such that $(w', B') \sqsubseteq (w, B)$ and $v \in \gamma(w', B')$.*

Given a PRS in normal form, by *tri*(Δ), *par*(Δ), and *seq*(Δ) we denote the system Δ restricted to trivial, parallel, and sequential rules, respectively. A derivation in *tri*(Δ) is called a *trivial* derivation in Δ. In the following we write simply *tri*, *par*, *seq* as Δ is always clearly determined by the context.

Definition 4.4. *Let Δ be a PRS in normal form and $\beta(w, B, C)$ be a β-formula. The PRS Δ is in* flat (w, B, C)-form *if and only if for each $X, Y \in$ Const(Δ), each $(w', B', C') \sqsubseteq (w, B, C)$, and each $B'' \subseteq B$, the following conditions hold:*

1. *If there is a finite derivation* $X \xrightarrow{u} Y$ *satisfying* $\gamma(w', B'')$*, then there is also a finite derivation* $X \xrightarrow{v}_{tri} Y$ *satisfying* $\gamma(w', B'')$*.*

2. *If there is a term* t *and a finite derivation* $X \xrightarrow{u} t$ *satisfying* $\gamma(w', B'')$*, then there is also a constant* Z *and a finite derivation* $X \xrightarrow{v}_{tri} Z$ *satisfying* $\gamma(w', B'')$*.*

3. *If* $w' = \varepsilon$ *and there is an infinite derivation* $X \xrightarrow{u}$ *satisfying* $\beta(w', B', C')$*, then there is also an infinite derivation* $X \xrightarrow{v}_{tri}$ *satisfying* $\beta(w', B', C')$*.*

4. *If there is an infinite derivation* $X \xrightarrow{u}_{par}$ *satisfying* $\beta(w', B', C')$*, then there is also an infinite derivation* $X \xrightarrow{v}_{tri}$ *satisfying* $\beta(w', B', C')$*;*

5. *If there is an infinite derivation* $X \xrightarrow{u}_{seq}$ *satisfying* $\beta(w', B', C')$*, then there is also an infinite derivation* $X \xrightarrow{v}_{tri}$ *satisfying* $\beta(w', B', C')$*.*

Intuitively, the system is in flat (w, B, C)-form if for every derivation of one of the listed types there is an "equivalent" trivial derivation. All conditions of the definition can be checked due to the following lemma, [Boz05], and decidability of LTL model checking for PDA and PN. Lemma 4.6 says that every PRS in normal form can be transformed into an "equivalent" flat system. Finally, the Lemma 4.9 says that if a PRS system in flat (w, B, C)-form has an infinite derivation satisfying $\beta(w, B, C)$, then it has also a trivial infinite derivation satisfying $\beta(w, B, C)$. Note that it is easy to check whether such a trivial derivation exists.

Lemma 4.5. *Given a* γ*-language* $\gamma(w, C)$*, a PRS system* Δ*, and constants* X, Y*, the following problems are decidable:*
(i) Is there any derivation $X \xrightarrow{u} Y$ *satisfying* $\gamma(w, C)$*?*
(ii) Is there any derivation $X \xrightarrow{u} t$ *such that* t *is a term and* $u \in \gamma(w, C)$*?*

Proof. The problems can be reduced to the reachability problem for wPRS (i.e. to decide whether given states p_1t_1, p_2t_2 of a given wPRS system Δ' satisfy $p_1t_1 \xrightarrow{v}_{\Delta'} p_2t_2$ for some v), which is known to be decidable [KŘS04a].

(i) Let $w = a_1O_1 \dots a_nO_n$. We construct a wPRS Δ' with the set of control states $\{1, 2, \dots n\} \cup 2^C$. The set of rewrite rules is defined as follows. We use $(n+1)$ as another name for the control state $\emptyset$.

  - For every $1 \le i \le n$ and every rule $t_1 \overset{a_i}{\hookrightarrow} t_2$ of Δ, we add to Δ' the rule $it_1 \overset{a_i}{\hookrightarrow} (i+1)t_2$ and if $O_i = U_+$ then also the rule $it_1 \overset{a_i}{\hookrightarrow} it_2$.

- For every $b \in C$, every $D \subseteq C$, and every rule $t_1 \overset{b}{\hookrightarrow} t_2$ of Δ, we add to Δ' the rule $Dt_1 \overset{b}{\hookrightarrow} (D \cup \{b\})t_2$.

Obviously, a word $u \in Act^*$ satisfies $1X \xrightarrow{u}_{\Delta'} CY$ if and only if it satisfies both $X \xrightarrow{u}_{\Delta} Y$ and $u \in \gamma(w, C)$. As we can decide whether $1X \xrightarrow{u}_{\Delta'} CY$ holds for some u, we can decide Problem (i).

(ii) We construct a wPRS Δ' as in the previous case. Moreover, for every $Z \in Const(\Delta)$ we add to Δ' the rule $CZ \overset{e}{\hookrightarrow} C\varepsilon$. It is easy to see that if a word $u \in \gamma(w, C)$ satisfies $X \xrightarrow{u}_{\Delta} t$ for some t, then $1X \xrightarrow{ue^m}_{\Delta'} C\varepsilon$ holds for some $m \geq 0$. Conversely, if $1X \xrightarrow{v}_{\Delta'} C\varepsilon$ holds for some v, then some prefix u of v satisfies both $u \in \gamma(w, C)$ and $X \xrightarrow{u}_{\Delta} t$ for some t. As we can decide whether $1X \xrightarrow{v}_{\Delta'} C\varepsilon$ holds for some v, we can decide Problem (ii).

□

The proof of the following lemma contains the algorithmic core of this section.

Lemma 4.6. *Let Δ be a PRS in normal form and $\beta(w, B, C)$ be a β-formula. One can construct a PRS Δ' in flat (w, B, C)-form such that for each $(w', B', C') \sqsubseteq (w, B, C)$ and each $X \in Const(\Delta)$, Δ' is equivalent to Δ with respect to the existence of an infinite derivation starting from X and satisfying $\beta(w', B', C')$.*

Proof. In order to obtain Δ', we describe an algorithm extending Δ with trivial rewrite rules in accordance with Conditions 1–5 of Definition 4.4.

All conditions of Definition 4.4 can be checked for each $X, Y \in Const(\Delta)$, each $(w', B', C') \sqsubseteq (w, B, C)$, and each $B'' \subseteq B$. For Conditions 1 and 2, this follows from Lemma 4.5. The problem whether there is an infinite derivation $X \xrightarrow{u}$ satisfying $\beta(\varepsilon, B', C')$ is a special case of the *fairness problem*, which is decidable due to [Boz05]. Finally, Conditions 4 and 5 can be checked due to decidability of LTL model checking for PDA and PN. If there is a non-satisfied condition, we add some trivial rules forming the missing derivation.

Let us assume that Condition 3 (resp., 4 or 5) is not satisfied, i.e. there exists an infinite derivation $X \xrightarrow{u}$ (resp. $X \xrightarrow{u}_{par}$ or $X \xrightarrow{u}_{seq}$) satisfying $\beta(w', B', C')$ for some $(w', B', C') \sqsubseteq (w, B, C)$ and violating the condition. Remark 4.2 implies that C' is nonempty and there is a finite derivation $X \xrightarrow{v}_{\Delta} t$ satisfying $\gamma(w', B')$. Hence, there exists an ordering of $B' = \{b_1, b_2, \ldots, b_m\}$ such that

(*) for each $1 \leq j \leq m$, there is a finite derivation in Δ starting from X and satisfying $\gamma(w', \{b_1, \dots, b_j\})$.

Such an ordering can be effectively computed using Lemma 4.5. Further, let $w' = a_1O_1a_2O_2 \dots a_nO_n$ and let $C' = \{c_1, c_2, \dots, c_k\}$. Then, we add the trivial rule $Z_{i-1} \overset{a_i}{\hookrightarrow} Z_i$ for each $1 \leq i \leq n$, the trivial rule $Z_{n+j-1} \overset{b_j}{\hookrightarrow} Z_{n+j}$ for each $1 \leq j \leq m$, and the trivial rule $Z_{n+m+j-1} \overset{c_j}{\hookrightarrow} Z_{n+m+j}$ for each $1 \leq j \leq k$, where $Z_0 = X$, $Z_1, \dots, Z_{n+m+k-1}$ are fresh process constants, and $Z_{n+m+k} = Z_{n+m}$. These added rules form an infinite derivation using only trivial rules, starting from X, and satisfying $\beta(w', B', C')$.

Similarly, if there are X, Y, and $\gamma(w', B'')$ with $w' = a_1O_1a_2O_2 \dots a_nO_n$ such that Condition 1 or 2 of Definition 4.4 is violated, then we first compute an ordering $\{b_1, \dots, b_m\}$ of B'' satisfying (*), and then we add the trivial rule $Z_{i-1} \overset{a_i}{\hookrightarrow} Z_i$ for each $1 \leq i \leq n$, and the trivial rule $Z_{n+j-1} \overset{b_j}{\hookrightarrow} Z_{n+j}$ for each $1 \leq j \leq m$, where $Z_0 = X$ and $Z_1, \dots, Z_{n+m}$ are fresh process constants (with exception of Z_{n+m} which is Y in the case of Condition 1). The added trivial rules generate derivation $X \xrightarrow{a_1 \dots a_n b_1 \dots b_m} Z_{n+m}$ satisfying $\gamma(w', B'')$.

Let Δ'' be the PRS Δ extended with the new rules. The condition (*) ensures that, for each $X \in \mathit{Const}(\Delta)$ and each $(w', B', C') \sqsubseteq (w, B, C)$, Δ'' is equivalent to Δ with respect to the existence of an infinite derivation starting from X and satisfying $\beta(w', B', C')$. If Δ'' is not in flat (w, B, C)-form, then the algorithm repeats the procedure described above on the system Δ'' with the difference that X and Y range over the constants of the original system Δ. The algorithm eventually terminates as the number of iterations is bounded by the number of pairs of process constants X, Y of Δ, times the number of triples (w', B', C') satisfying $(w', B', C') \sqsubseteq (w, B, C)$, and times the number of subsets $B'' \subseteq B$. We claim that the resulting PRS Δ' is in flat (w, B, C)-form. For the process constants of the original system Δ, by construction Δ' satisfies all conditions of Definition 4.4. For the added constants, it is sufficient to observe that any derivation in Δ' starting from such a constant either is trivial or has a trivial prefix leading to a constant of Δ. Hence, Δ' is the desired PRS system. □

Definition 4.7 (Subderivation). *Let Δ be a PRS in normal form and σ_1 be a (finite or infinite) derivation $s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \dots$, where $s_1 \xrightarrow{a_1} s_2$ has the form $X \xrightarrow{a_1} Y.Z$ and, for each $i \geq 2$, if s_i is not the last state of the derivation, then it has the form $s_i = t_i.Z$ with $t_i \neq \varepsilon$. Then σ_1 is called a* subderivation *of a derivation σ if σ has a suffix σ' satisfying the following:*

1. *every transition step in σ' is of the form $s_i \| t' \xrightarrow{a_i} s_{i+1} \| t'$ or $s_i \| t' \xrightarrow{b} s_i \| t''$, where $t' \xrightarrow{b} t''$,*

2. *in σ', if we replace every step of the form $s_i \| t' \xrightarrow{a_i} s_{i+1} \| t'$ by step $s_i \xrightarrow{a_i} s_{i+1}$ and we skip every step of the form $s_i \| t' \xrightarrow{b} s_i \| t''$, we get precisely σ_1.*

Further, if σ_1 and σ are finite, the last term of σ_1 is a process constant, and σ is a prefix of a derivation σ', then σ_1 is also a subderivation *of σ'.*

Remark 4.8. *Let Δ be a PRS in normal form and σ be a derivation of Δ having a suffix σ' of the form $\sigma' = X \| t \xrightarrow{a} (Y.Z) \| t \xrightarrow{u}$. Then, there is a subderivation of σ whose first transition step $X \xrightarrow{a} Y.Z$ corresponds to the first transition step of σ'.*

Intuitively, the subderivation captures the behaviour of the subterm Y.Z since its emergence until it is possibly reduced to a term without any sequential composition. Due to the normal form of Δ, the subterm Y.Z behaves undependently on the rest of the term (as long as it contains a sequential composition).

Lemma 4.9. *Let Δ be a PRS in flat (w, B, C)-form. Then, the following condition holds for each $X \in Const(\Delta)$ and each $(w', B', C') \sqsubseteq (w, B, C)$:*
If there is an infinite derivation $X \xrightarrow{u}$ satisfying $\beta(w', B', C')$, then there is also an infinite derivation $X \xrightarrow{v}_{tri}$ satisfying $\beta(w', B', C')$.

A sketch of the proof. Given an infinite derivation σ satisfying a formula $\beta(\sigma) = \beta(w', B', C')$ where $(w', B', C') \sqsubseteq (w, B, C)$, by *trivial equivalent* of σ we mean an infinite trivial derivation starting in the same term as σ and satisfying $\beta(\sigma)$. Similarly, given a finite derivation σ satisfying some $\gamma(\sigma) = \gamma(w', B')$ where $(w', B') \sqsubseteq (w, B)$, by *trivial equivalent* of σ we mean a finite trivial derivation σ' such that σ' starts in the same term as σ, it satisfies $\gamma(\sigma)$, and if the last term of σ is a process constant, then the last term of σ' is the same process constant.

The lemma is proven by contradiction. We assume that there exist some infinite derivations violating the condition of the lemma. Let σ be one of these derivations such that the number of transition steps of σ generated by sequential non-trivial rules with actions $a \notin B$ is minimal (note that this number is always finite as we consider derivations satisfying $\beta(w', B', C')$ for some $(w', B', C') \sqsubseteq (w, B, C)$). First, we prove that every subderivation of σ has a trivial equivalent. Then we replace all subderivations of σ by the corresponding trivial equivalents. This step is technically nontrivial because

σ may have infinitely many subderivations. By the replacement we obtain an infinite derivation σ' satisfying $\beta(\sigma)$ and starting in the same process constant as σ. Moreover, σ' has no subderivations and hence it does not contain any sequential operator. Flat (w, B, C)-form of Δ (Condition 4) implies that σ' has a trivial equivalent. This is also a trivial equivalent of σ which means that σ does not violate the condition of our lemma.

Proof. In this proof, by β-formula we always mean a formula of the form $\beta(w', B', C')$ where $(w', B', C') \sqsubseteq (w, B, C)$. We also consider only infinite derivations satisfying some of these β-formulae. Remark 4.3 implies that such an infinite derivation σ satisfies exactly one β-formula. We denote this β-formula by $\beta(\sigma)$. Further, by $\mathit{SEQ}(\sigma)$ we denote the number of transition steps $t_i \xrightarrow{a} t_{i+1}$ of σ generated by a sequential non-trivial rule and such that $a \notin B$. Note that $\mathit{SEQ}(\sigma)$ is always finite due to the restrictions on considered infinite derivations. Given an infinite derivation σ, by its *trivial equivalent* we mean an infinite trivial derivation starting in the same term as σ and satisfying $\beta(\sigma)$.

Similarly, we consider only finite derivations satisfying some $\gamma(w', B')$ where $(w', B') \sqsubseteq (w, B)$. Remark 4.3 implies that such a finite derivation σ satisfies exactly one γ-language, which is denoted by $\gamma(\sigma)$. Given a finite derivation σ, by its *trivial equivalent* we mean a finite trivial derivation σ' such that σ' starts in the same term as σ, it satisfies $\gamma(\sigma)$, and if the last term of σ is a process constant, then the last term of σ' is the same process constant.

Using the introduced terminology, the lemma says that every infinite derivation starting in a process constant has a trivial equivalent. For the sake of contradiction, we assume that the lemma does not hold. Let Σ be the set of infinite derivations violating the lemma and let $k = \min\{\mathit{SEQ}(\sigma) \mid \sigma \in \Sigma\}$.

First of all, we prove two claims.

Claim 1. Let σ be an infinite derivation satisfying $\mathit{SEQ}(\sigma) \leq k$. Then every subderivation of σ has a trivial equivalent.

Proof of the claim: For finite subderivations, the existence of trivial equivalents follows directly from the flat (w, B, C)-form of Δ (Conditions 1 and 2). Let σ_1 be an infinite subderivation of σ. It has the form $\sigma_1 = X \xrightarrow{a}_{\mathit{seq}} Y.Z \xrightarrow{b_1} t_1.Z \xrightarrow{b_2} t_2.Z \xrightarrow{b_3} \ldots$ where $t_1, t_2, \ldots$ are nonempty terms. There are two cases:

- If $a \in B$, then $\beta(\sigma_1)$ has the form $\beta(\varepsilon, B', C')$. Hence, σ_1 has a trivial equivalent due to the flat (w, B, C)-form of Δ (Condition 3).

- If $a \notin B$, then the first step $X \xrightarrow{a}_{seq} Y.Z$ of σ_1 is counted in $SEQ(\sigma_1)$ and the corresponding step $X\|t' \xrightarrow{a}_{seq} Y.Z\|t'$ of σ is counted in $SEQ(\sigma)$. Hence, $0 < SEQ(\sigma)$. Let σ_2 be the derivation $\sigma_2 = Y \xrightarrow{b_1} t_1 \xrightarrow{b_2} t_2 \xrightarrow{b_3} \ldots$. As $SEQ(\sigma_2) < SEQ(\sigma_1) \leq k$, the definition of k implies that σ_2 has a trivial equivalent $\sigma_2' = Y \xrightarrow{c_1}_{tri} Y_1 \xrightarrow{c_2}_{tri} Y_2 \xrightarrow{c_3}_{tri}$. Further, as σ_2' satisfies $\beta(\sigma_2)$, the derivation $\sigma_1' = X \xrightarrow{a}_{seq} Y.Z \xrightarrow{c_1}_{tri} Y_1.Z \xrightarrow{c_2}_{tri} Y_2.Z \xrightarrow{c_3}_{tri} \ldots$ satisfies $\beta(\sigma_1)$. Moreover, the flat (w, B, C)-form of Δ (Condition 5) implies that σ_1' has a trivial equivalent. Obviously, it is also a trivial equivalent of σ_1. □

Claim 2. Let σ be an infinite derivation such that $SEQ(\sigma) \leq k$, it starts in a parallel term p, and it satisfies a formula $\beta(w', B', C')$. Then there is an infinite derivation $p \xrightarrow{u}_{par} p' \xrightarrow{v}$ such that p' is a parallel term, $u \in \gamma(w', B')$, and v satisfies $\beta(\varepsilon, C', C')$.

Proof of the claim: Remark 4.2 implies that σ can be written as $p \xrightarrow{u_1} t \xrightarrow{u_2}$ where $p \xrightarrow{u_1} t$ is the *minimal* prefix of σ satisfying $\gamma(w', B')$ and such that $t \xrightarrow{u_2}$ satisfies $\beta(\varepsilon, C', C')$. Let $\widetilde{SEQ}(\sigma)$ denote the number of transition steps in the prefix $p \xrightarrow{u_1} t$ generated by sequential non-trivial rules (note that $\widetilde{SEQ}(\sigma) \geq SEQ(\sigma)$). We prove the claim by induction on $\widetilde{SEQ}(\sigma)$. The base case $\widetilde{SEQ}(\sigma) = 0$ is obvious. Now, assume that $\widetilde{SEQ}(\sigma) > 0$. Since p is parallel term and Δ is in normal form, the first transition step of $p \xrightarrow{u_1} t$ counted in $\widetilde{SEQ}(\sigma)$ has the form $Y\|p' \xrightarrow{a} (W.Z)\|p'$ and it corresponds to the first transition step $Y \xrightarrow{a} W.Z$ of a subderivation σ_1. In σ, we replace the subderivation σ_1 with its trivial equivalent (whose existence is guaranteed by Claim 1) and we obtain a new derivation σ'' starting from p, satisfying $\beta(\sigma)$ and such that $\widetilde{SEQ}(\sigma'') < \widetilde{SEQ}(\sigma)$. Hence, the second claim directly follows from the induction hypothesis. In the following, we describe the replacement of such a subderivation.

Let $\sigma_1 = Y \xrightarrow{u}$ and $\sigma_1' = Y \xrightarrow{v}_{tri}$ be its trivial equivalent. Let $\beta(\sigma_1) = \beta(c_1 O_1 c_2 O_2 \ldots c_n O_n, B'', C'')$. Then $u, v \in c_1^+ c_2^+ \ldots c_n^+ . B^\omega$. Recall that $c_1, c_2, \ldots, c_n$ are pairwise distinct and $B \subseteq Act \smallsetminus \{c_1, \ldots, c_n\}$. Intuitively, for every $1 \leq i \leq n$, we replace the first transition step of σ_1 labelled with c_i by the sequence of transition steps of σ_1' labelled with c_i, and then we cancel the other transition steps of σ_1 labelled with c_i.[2]

[2]By replacement of a transition step $s_1 \xrightarrow{a} s_2$ of σ_1 by a sequence $Y_1 \xrightarrow{v'}_{tri} Y_2$ of transition steps of σ_1' we mean that the corresponding transition step $s_1\|t' \xrightarrow{a} s_2\|t'$ of σ is replaced by $Y_1\|t' \xrightarrow{v'}_{tri} Y_2\|t'$, and all immediately succeeding steps $s_2\|t'' \xrightarrow{b} s_2\|t'''$ of σ are replaced by $Y_2\|t'' \xrightarrow{b} Y_2\|t'''$. Further, by cancellation of a transition step $s_1 \xrightarrow{c_i} s_2$ of σ_1 we mean that the corresponding transition step $s_1\|t' \xrightarrow{c_i} s_2\|t'$ of σ is replaced by $Y_2\|t'$, where Y_2 is the last process constant of σ_1' such that a transition under c_i leads to Y_2, and all immediately succeeding steps $s_2\|t'' \xrightarrow{b} s_2\|t'''$ of σ are replaced by $Y_2\|t'' \xrightarrow{b} Y_2\|t'''$.

Further, the first transition step of σ_1 labelled with an action of B is replaced with the minimal prefix of the remaining part of σ'_1 satisfying $\gamma(\varepsilon, B'')$. Finally, the remaining transition steps of σ_1 are orderly replaced with the remaining transition steps of σ'_1. The case when σ_1 and its trivial equivalent σ'_1 are finite is similar.

It is easy to see that the described replacement operation preserves the fulfillment of $\beta(\sigma)$ and the obtained derivation σ'' satisfies $\widetilde{SEQ}(\sigma'') < \widetilde{SEQ}(\sigma)$. □

With this claim, we can easily reach a contradiction. Let $\sigma = X \xrightarrow{u}$ be an infinite derivation such that $SEQ(\sigma) = k$ and it has no trivial equivalent. Further, let $\beta(\sigma) = (w', B', C')$. Note that C' is nonempty. Claim 2 says that there is a derivation $X \xrightarrow{u_1}_{par} p_1 \xrightarrow{v_1}$ where p_1 is a parallel term, $u_1 \in \gamma(w', B')$, and v_1 satisfies $\beta(\varepsilon, C', C')$. Applying the second claim on the suffix $p_1 \xrightarrow{v_1}$, we get a derivation $p_1 \xrightarrow{u_2}_{par} p_2 \xrightarrow{v_2}$ where p_2 is a parallel term, $u_2 \in \gamma(\varepsilon, C')$, and v_2 satisfies $\beta(\varepsilon, C', C')$. Iterating this argument, we get a sequence $(p_i \xrightarrow{u_{i+1}}_{par} p_{i+1})_{i \in \mathbb{N}}$ of derivations satisfying $\gamma(\varepsilon, C')$. These derivations are nonempty as C' is nonempty. Let us consider the derivation

$$\sigma' = X \xrightarrow{u_1}_{par} p_1 \xrightarrow{u_2}_{par} p_2 \xrightarrow{u_3}_{par} p_3 \xrightarrow{u_4}_{par} \ldots$$

Flat (w, B, C)-form of Δ (Condition 4) implies that σ' has a trivial equivalent. However, this is also a trivial equivalent of σ as both σ, σ' start with X and σ' satisfies $\beta(\sigma)$. This is a contradiction. □

Theorem 4.10. *The problem whether a given PRS Δ in normal form has an infinite run satisfying a given formula $\beta(w, B, C)$ is decidable.*

Proof. Due to Lemmata 4.6 and 4.9, the problem can be reduced to the problem whether there is an infinite derivation $X \xrightarrow{v}_{tri}$ satisfying $\beta(w, B, C)$. This problem corresponds to LTL model checking of finite-state systems, which is decidable. □

The following three theorems show that Theorem 4.10 holds even for wPRS and α-formulae.

Theorem 4.11. *The problem whether a given PRS Δ in normal form has an infinite run satisfying a given α-formula is decidable.*

Proof. Let Δ be a PRS in normal form and $\alpha(\theta_1 O_1 \ldots \theta_n O_n \xi, \mathcal{B})$ be an α-formula. For every θ_i and every rule $t_1 \overset{b}{\hookrightarrow} t_2$ such that b satisfies θ_i, we add a rule $t_1 \overset{a_i}{\hookrightarrow} t_2$, where a_i is a fresh action corresponding to θ_i. Similarly, for every $\psi \in \mathcal{B} \cup \{\xi\}$ and every rule

$t_1 \overset{b}{\hookrightarrow} t_2$ such that b satisfies $\psi \wedge \xi$, we add a rule $t_1 \overset{a_\psi}{\hookrightarrow} t_2$, where a_ψ is a fresh action. Let Δ' be the resulting PRS system. Note that Δ' is also in normal form. Obviously, Δ has an infinite run satisfying the original α-formula if and only if Δ' has an infinite run satisfying $\alpha(a_1 O_1 \ldots a_n O_n (a_\xi \vee \bigvee_{b \in C} b), C)$, where $C = \{a_\psi \mid \psi \in \mathcal{B}\}$. It is an easy exercise to show that this new α-formula can be effectively transformed into a disjunction of β-formulae which is equivalent with respect to infinite words. Hence, the problem is decidable due to Theorem 4.10. $\square$

Theorem 4.12. *The problem whether a given PRS Δ has an infinite run satisfying a given α-formula is decidable.*

Proof. Let Δ be a PRS, $\alpha(\delta, \mathcal{B})$ be an α-formula, and $e \notin \mathit{Act}(\Delta)$ be a fresh action. First we describe our modification of the standard algorithm [May00] that transforms Δ into a PRS in normal form.

If Δ is not in normal form, then there exists a rule r which is neither parallel nor sequential; r has one of the following forms:

1. $r = t \overset{a}{\hookrightarrow} t_1 \| t_2$ (resp., $r = t_1 \| t_2 \overset{a}{\hookrightarrow} t$) where t or t_1 or t_2 is not a parallel term. Let $Z_1, Z_2, Z \notin \mathit{Const}(\Delta)$ be fresh process constants. We replace r with the rules $t \overset{e}{\hookrightarrow} Z$, $Z \overset{a}{\hookrightarrow} Z_1 \| Z_2$, $Z_1 \overset{e}{\hookrightarrow} t_1$, and $Z_2 \overset{e}{\hookrightarrow} t_2$ (resp., $t_1 \overset{e}{\hookrightarrow} Z_1$, $t_2 \overset{e}{\hookrightarrow} Z_2$, $Z_1 \| Z_2 \overset{a}{\hookrightarrow} Z$, and $Z \overset{e}{\hookrightarrow} t$).

2. $r = t \overset{a}{\hookrightarrow} t_1.(t_2 \| t_3)$ (resp., $r = t_1.(t_2 \| t_3) \overset{a}{\hookrightarrow} t$). Let $Z \notin \mathit{Const}(\Delta)$ be a fresh process constant. We modify Δ in two steps. First, we replace $t_2 \| t_3$ by Z in left-hand and right-hand sides of all rules of Δ. Then, we add the rules $Z \overset{e}{\hookrightarrow} t_2 \| t_3$ and $t_2 \| t_3 \overset{e}{\hookrightarrow} Z$.

3. $r = t_1 \overset{a}{\hookrightarrow} t_2.X$ (resp., $r = t_2.X \overset{a}{\hookrightarrow} t_1$) where t_1 or t_2 is not a process constant. Let $Z_1, Z_2 \notin \mathit{Const}(\Delta)$ be fresh process constants. We replace r with the rules $t_1 \overset{e}{\hookrightarrow} Z_1$, $Z_1 \overset{a}{\hookrightarrow} Z_2.X$, and $Z_2 \overset{e}{\hookrightarrow} t_2$ (resp., $t_2 \overset{e}{\hookrightarrow} Z_2$, $Z_2.X \overset{a}{\hookrightarrow} Z_1$, and $Z_1 \overset{e}{\hookrightarrow} t_1$).

After a finite number of applications of this procedure (with the same action e), we obtain a PRS Δ' in normal form.

We define a formula $\alpha(\delta', \mathcal{B}')$, where $\mathcal{B}' = \mathcal{B} \cup \{\bigvee_{a \in \mathit{Act}(\Delta)} a\}$ and δ' arises from $\delta = \theta_1 O_1 \ldots \theta_n O_n \xi$ by the following substitution for every $1 \leq i \leq n$.

- If O_i is U, then replace the pair $\theta_i \,\mathsf{U}$ by the pair $(e \vee \theta_i) \,\mathsf{U}$.
- If O_i is U_+, then replace the pair $\theta_i \,\mathsf{U}_+$ by the sequence $(e \vee \theta_i) \,\mathsf{U}\, \theta_i \,\mathsf{U}_+$.

- If O_i is $\wedge \mathsf{X}$, then replace the pair $\theta_i \wedge \mathsf{X}$ by the sequence $e \,\mathsf{U}\, \theta_i \wedge \mathsf{X}$.
- $\theta_n O_n = \theta_n \wedge \mathsf{G_s}$ is replaced by the sequence $e \,\mathsf{U}\, \theta_n \wedge \mathsf{G_s}$.
- ξ is replaced by $(\xi \vee e)$.

Let us note that the construction of $\mathcal{B}'$ ensures that any word with a suffix e^ω does not satisfy $\alpha(\delta', \mathcal{B}')$. Observe that $u' \models \alpha(\delta', \mathcal{B}')$ if and only if $u \models \alpha(\delta, \mathcal{B})$, where u is obtained from u' by eliminating all occurrences of action e.

Clearly, Δ has an infinite run satisfying $\alpha(\delta, \mathcal{B})$ if and only if Δ' has an infinite run satisfying $\alpha(\delta', \mathcal{B}')$. As Δ' is in normal form, we can now apply Theorem 4.11. □

Theorem 4.13. *The problem whether a given wPRS system has an infinite run satisfying a given α-formula is decidable.*

Proof. Let Δ be a wPRS with initial state p_0X_0 and $\alpha(\delta, \mathcal{B})$ be an α-formula. We construct a PRS Δ' with initial state X_0 which can simulate Δ. We also define set of formulae recognizing correct simulations.

The system Δ' is very similar to Δ. We only change actions of rules to hold information about control states in the rules and then we remove all control states. More precisely, for every rule of the form $pt_1 \overset{a}{\hookrightarrow} pt_2$ of Δ we add to Δ' the rule $t_1 \overset{a_{[p]}}{\hookrightarrow} t_2$, and for every rule of the form $pt_1 \overset{a}{\hookrightarrow} qt_2$ of Δ we add to Δ' the rule $t_1 \overset{a_{[p<q]}}{\hookrightarrow} t_2$.

Further, we modify the formula $\alpha(\delta, \mathcal{B})$ such that every occurrence of each action a is replaced by $\bigvee_{q \in M(\Delta)} (a_{[q]} \vee \bigvee_{p<q} a_{[p<q]})$. Let $\alpha(\delta', \mathcal{B}')$ be the resulting formula.

Moreover, for every sequence $p_1 < p_2 < \ldots < p_k$ of control states of $M(\Delta)$ such that $p_1 = p_0$, we define an α-formula

$$\varphi_{[p_1<p_2<\ldots<p_k]} = \alpha(\theta_{[p_1]} \,\mathsf{U}\, \theta_{[p_1<p_2]} \wedge \mathsf{X}\, \theta_{[p_2]} \,\mathsf{U}\, \theta_{[p_2<p_3]} \wedge \mathsf{X} \,\ldots\, \theta_{[p_{k-1}<p_k]} \wedge \mathsf{G_s}\theta_{[p_k]}, \emptyset)$$

where $\theta_{[p_i]} = \bigvee_{a \in Act(\Delta)} a_{[p_i]}$ and $\theta_{[p_i<p_j]} = \bigvee_{a \in Act(\Delta)} a_{[p_i<p_j]}$.

It is easy to see that there is an infinite run of Δ satisfying $\alpha(\delta, \mathcal{B})$ if and only if there is an infinite run of Δ' satisfying $\alpha(\delta', \mathcal{B}')$ and $\varphi_{[p_1<p_2<\ldots<p_k]}$ for some sequence $p_1 < p_2 < \ldots < p_k$. As the number of such sequences is finite and each $\varphi_{[p_1<p_2<\ldots<p_k]}$ is an α-formula, Theorem 4.12 and Lemma 3.1 imply that the considered problem is decidable. □

As LTL($\mathsf{F_s}, \mathsf{G_s}$) is closed under negation, Theorems 3.2, 4.1, and 4.13 give us the following.

Corollary 4.14. *The model checking problem for wPRS and LTL($\mathsf{F_s}, \mathsf{G_s}$) is decidable.*

This problem is EXPSPACE-hard due to EXPSPACE-hardness of the model checking problem for LTL(F, G) for PN [Hab97]. Our decidability proof does not provide any primitive recursive upper bound as it employs LTL model checking for PN, for which no primitive recursive upper bound is known.

5 Undecidability results

Obviously, the model checking for wPRS and LTL(X) is decidable. Hence, to show that the decidability boundary of Figure 2 is drawn correctly, we have to prove that the model checking problem is undecidable for the class PA and fragments LTL(U) and LTL($\overset{\infty}{\mathsf{F}}, \mathsf{X}$). The undecidability proofs are based on reduction from the non-halting problem for Minsky 2-counter machines, which is known to be undecidable [Min67].

We recall a definition of Minsky machines first. A *Minsky 2-counter machine*, or a *machine* for short, is a finite sequence $N = l_1\colon i_1,\ l_2\colon i_2,\ \ldots,\ l_{n-1}\colon i_{n-1},\ l_n\colon \texttt{halt}$, where $n \geq 1$, $l_1, l_2, \ldots, l_n$ are *labels*, and each i_j is an instruction for

- *increment*: $\texttt{c}_k\texttt{:=c}_k\texttt{+1; goto } l_r$, or
- *test-and-decrement*: $\texttt{if c}_k\texttt{>0 then c}_k\texttt{:=c}_k\texttt{-1; goto } l_r \texttt{ else goto } l_s$

where $k \in \{1, 2\}$ and $1 \leq r, s \leq n$.

The machine N induces a transition relation $\longrightarrow$ over *configurations* of the form (l_j, v_1, v_2), where l_j is a label of an instruction to be executed and $v_1, v_2 \geq 0$ represent current values of counters c_1 and c_2, respectively.

We say that the machine N *halts* if there are numbers $v_1, v_2 \geq 0$ such that $(l_1, 0, 0) \longrightarrow^* (l_n, v_1, v_2)$. The *non-halting problem* is to decide whether a given machine N does not halt. The problem is undecidable [Min67].

Theorem 5.1. *Model checking of PA against LTL(U) is undecidable.*

Proof. Given a machine N, we construct a PA system Δ_N with initial term $D_1 \| D_2 \| H$ and set of rules containing

- for every instruction $l_i\colon \texttt{c}_k\texttt{:=c}_k\texttt{+1; goto } l_r$, the rules

$$D_k \overset{l_i}{\hookrightarrow} S_k.D_k \qquad C_k \overset{l_i}{\hookrightarrow} S_k.C_k \qquad S_k \overset{inc_i}{\hookrightarrow} C_k.S_k$$

- for every instruction l_i : `if` c_k`>0 then` c_k`:=`c_k`-1; goto` l_r `else goto` l_s, the rules

$$D_k \overset{l_i}{\hookrightarrow} E_k \qquad E_k \overset{zero_i}{\hookrightarrow} D_k \qquad C_k \overset{l_i}{\hookrightarrow} \varepsilon \qquad S_k \overset{dec_i}{\hookrightarrow} \varepsilon$$

- the rule $H \overset{l_n}{\hookrightarrow} H$ corresponding to instruction l_n: `halt`.

Now, we define a formula ψ describing a correct step of the constructed PA when simulating machine N.

$$\begin{aligned}\psi \;=\; & \textstyle\bigwedge_{\text{each } l_i: c_k := c_k+1;\ \texttt{goto}\ l_r} \big((l_i \Longrightarrow (l_i \,\mathsf{U}\, inc_i)) \wedge (inc_i \Longrightarrow (inc_i \,\mathsf{U}\, l_r))\big) \wedge \\ & \textstyle\bigwedge_{\text{each } l_i:\texttt{if}\ c_k>0\ \texttt{then}\ c_k:=c_k-1;\ \texttt{goto}\ l_r\ \texttt{else goto}\ l_s} \big((l_i \Longrightarrow (l_i \,\mathsf{U}\, (dec_i \vee zero_i))) \\ & \qquad \wedge (dec_i \Longrightarrow (dec_i \,\mathsf{U}\, l_r)) \\ & \qquad \wedge (zero_i \Longrightarrow (zero_i \,\mathsf{U}\, l_s))\big)\end{aligned}$$

Finally, we set $\varphi = l_1 \wedge (\psi \,\mathsf{U}\, l_n)$. It is easy to see that machine N halts if and only if the system Δ_N has a run satisfying φ. In other words, the machine N does not halt if and only if $L(\Delta_N) \models \neg\varphi$. $\square$

Theorem 5.2. *Model checking of PA against LTL($\overset{\infty}{\mathsf{F}}$, X) is undecidable.*

Proof. Given a machine $N = l_1 : i_1,\ l_2 : i_2,\ \ldots,\ l_{n-1} : i_{n-1},\ l_n$: `halt`, we construct a PA system Δ_N with initial term $D_1 \| D_2 \| H$ and set of rules containing

- for every instruction l_i: c_k`:=`c_k`+1; goto` l_r, the rules

$$D_k \overset{inc_i}{\hookrightarrow} C_k.D_k \qquad C_k \overset{inc_i}{\hookrightarrow} C_k.C_k$$

- for every instruction l_i : `if` c_k`>0 then` c_k`:=`c_k`-1; goto` l_r `else goto` l_s, the rules

$$D_k \overset{zero_i}{\hookrightarrow} D_k \qquad C_k \overset{dec_i}{\hookrightarrow} \varepsilon$$

- rules corresponding to `halt` and instruction labels

$$H \overset{halt}{\hookrightarrow} H \qquad H \overset{l_i}{\hookrightarrow} H \text{ for every } 1 \leq i \leq n$$

- and the rules allowing to reset the counters

$$C_1 \overset{del_1}{\hookrightarrow} \varepsilon \qquad C_2 \overset{del_2}{\hookrightarrow} \varepsilon \qquad D_1 \overset{reset_1}{\hookrightarrow} D_1 \qquad D_2 \overset{reset_2}{\hookrightarrow} D_2$$

As in the previous proof, we define a formula ψ describing a correct step of the constructed PA when simulating machine N.

$$\begin{aligned}\psi \;=\; & \textstyle\bigwedge_{\text{each } l_i:\texttt{c}_k\texttt{:=c}_k\texttt{+1; goto } l_r} \big((l_i \Longrightarrow \mathsf{X}\mathit{inc}_i) \wedge (\mathit{inc}_i \Longrightarrow \mathsf{X}l_r)\big) \wedge \\ & \textstyle\bigwedge_{\text{each } l_i:\texttt{if c}_k\texttt{>0 then c}_k\texttt{:=c}_k\texttt{-1; goto } l_r \texttt{ else goto } l_s} \big((l_i \Longrightarrow \mathsf{X}(\mathit{dec}_i \vee \mathit{zero}_i)) \\ & \qquad\qquad \wedge (\mathit{dec}_i \Longrightarrow \mathsf{X}l_r) \\ & \qquad\qquad \wedge (\mathit{zero}_i \Longrightarrow \mathsf{X}l_s)\big) \\ & \wedge (l_n \Longrightarrow \mathsf{X}\mathit{halt})\end{aligned}$$

Moreover, we define a formula ρ describing a correct step of resetting counters and restarting the simulation.

$$\begin{aligned}\rho \;=\; & \quad (\mathit{halt} \Longrightarrow \mathsf{X}(\mathit{del}_1 \vee \mathit{reset}_1)) \;\wedge\; (\mathit{del}_1 \Longrightarrow \mathsf{X}(\mathit{del}_1 \vee \mathit{reset}_1)) \\ & \wedge\; (\mathit{reset}_1 \Longrightarrow \mathsf{X}(\mathit{del}_2 \vee \mathit{reset}_2)) \;\wedge\; (\mathit{del}_2 \Longrightarrow \mathsf{X}(\mathit{del}_2 \vee \mathit{reset}_2)) \\ & \wedge\; (\mathit{reset}_2 \Longrightarrow \mathsf{X}l_1)\end{aligned}$$

The formula $\varphi = \overset{\infty}{\mathsf{G}}(\psi \wedge \rho) \wedge \overset{\infty}{\mathsf{F}}\mathit{halt}$ says that at some point the *halt* action occurs, both counters are reset, a correct simulation is started, and whenever the simulation ends (with *halt* action), this sequence of events is performed again. Moreover, note that φ is satisfied only if the action *halt* appears infinitely many times. Hence, there is a run of Δ_N satisfying φ if and only if N halts. In other words, the machine N does not halt if and only if $L(\Delta_N) \models \neg\varphi$. $\square$

In the proofs of the previous two theorems, the PA systems constructed there have only infinite runs. This means that model checking of infinite runs remains undecidable for PA and both LTL($\overset{\infty}{\mathsf{F}}$, X) and LTL(U). However, note that model checking of *finite runs* against LTL($\overset{\infty}{\mathsf{F}}$, X) is decidable, even for wPRS. The proof is based on the observation that a nonempty finite run satisfies $\overset{\infty}{\mathsf{F}}\varphi$ if and only if the last action of the run satisfies φ. The same holds for the formula $\overset{\infty}{\mathsf{G}}\varphi$. Hence, if we restrict only to nonempty finite runs, the modalities $\overset{\infty}{\mathsf{F}}, \overset{\infty}{\mathsf{G}}$ are equivalent. The observation also implies that $\overset{\infty}{\mathsf{F}}\neg\varphi$ is equivalent to $\neg\overset{\infty}{\mathsf{F}}\varphi$, $\overset{\infty}{\mathsf{F}}(\varphi_1 \wedge \varphi_2)$ is equivalent to $(\overset{\infty}{\mathsf{F}}\varphi_1) \wedge (\overset{\infty}{\mathsf{F}}\varphi_2)$, $\overset{\infty}{\mathsf{F}}\overset{\infty}{\mathsf{F}}\varphi$ is equivalent to $\overset{\infty}{\mathsf{F}}\varphi$, and that $\overset{\infty}{\mathsf{F}}\mathsf{X}\varphi$ never holds. It is now easy to see that every LTL($\overset{\infty}{\mathsf{F}}$, X) formula can describe only a bounded prefix of a finite run (using the modality X) and the last action of the run. Thus, decidability of model checking of finite runs against LTL($\overset{\infty}{\mathsf{F}}$, X) follows from decidability of the *reachability Hennessy-Milner property* problem [KŘS05].

6 Conclusion

We have established the decidability border of model checking of wPRS classes and basic fragments of future LTL (see Figure 2) by showing that the model checking problem of wPRS against $\mathrm{LTL}(\mathsf{F_s}, \mathsf{G_s})$ is decidable, while the same problem for PA and $\mathrm{LTL}(\mathsf{U})$ or $\mathrm{LTL}(\mathsf{X}, \overset{\infty}{\mathsf{F}})$ is undecidable. So far, only two positive results on LTL model checking of PA (and classes subsuming PA) have been published: decidability of model checking of infinite runs for PRS and LTL fragment of fairness properties [Boz05] and decidability of the same problem for PA and *simple* $\mathit{PLTL}_\Box$ [BH96]. Note that the fairness fragment and the regular part of simple $\mathrm{PLTL}_\Box$ are strictly less expressive than $\mathrm{LTL}(\mathsf{F}, \mathsf{G})$ (also known as Lamport logic), which is again strictly less expressive than $\mathrm{LTL}(\mathsf{F_s}, \mathsf{G_s})$. We also emphasize that our positive result for $\mathrm{LTL}(\mathsf{F_s}, \mathsf{G_s})$ deals with both finite and infinite runs, and with wPRS rather than PRS or PA only.

It is also worth mentioning that our proof techniques differ from those used in [Boz05] and [BH96]. The decidability proof for $\mathrm{LTL}(\mathsf{F_s}, \mathsf{G_s})$ is based on the auxiliary result saying that model checking for wPRS and negated $\mathcal{A}$ fragment is decidable. In fact, this auxiliary result is very powerful. We conjecture that it also implies decidability of the model checking problem of wPRS and the common fragment of CTL and LTL called $\mathrm{LTL}^{\mathrm{det}}$ [Mai00]. Note that $\mathrm{LTL}^{\mathrm{det}}$ is semantically incomparable with $\mathrm{LTL}(\mathsf{F_s}, \mathsf{G_s})$.

Unfortunately, our results are insufficient to establish the decidability border for basic LTL fragments with both future and past modalities. Indeed, fragments $\mathrm{LTL}(\mathsf{F_s}, \mathsf{P_s})$ and $\mathrm{LTL}(\mathsf{F}, \mathsf{P})$, where $\mathsf{P}, \mathsf{P_s}$ are past counterparts of $\mathsf{F}, \mathsf{F_s}$ respectively, do not semantically coincide with any fragment of Figure 2 and decidability of the model checking problem for these two fragments and all wPRS classes subsuming PA is an open question. However, we conjecture that our technique can be adopted to answer this question positively.

Acknowledgment. Authors have been partially supported as follows: M. Křetínský by the Grant Agency of the Czech Republic, grant No. 201/06/1338, V. Řehák by the research centre "Institute for Theoretical Computer Science (ITI)", project No. 1M0545, and J. Strejček by the Academy of Sciences of the Czech Republic, grant No. 1ET408050503. The paper has been written during L. Bozzelli's postdoc stay in LSV, CNRS & ENS Cachan and J. Strejček's postdoc stay in LaBRI, Univeristé Bordeaux 1.

References

[BEM97] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *Proc. of CONCUR'97*, volume

1243 of *LNCS*, pages 135–150, 1997.

[BH96] A. Bouajjani and P. Habermehl. Constrained properties, semilinear systems, and Petri nets. In *Proc. of CONCUR'96*, volume 1119 of *LNCS*, pages 481–497. Springer–Verlag, 1996.

[BKŘS06] L. Bozzelli, M. Křetínský, V. Řehák, and J. Strejček. On Decidability of LTL Model Checking for Process Rewrite Systems. In *Proceedings of FSTTCS 2006*, LNCS. Springer, 2006. To appear, Dec.2006.

[Boz05] L. Bozzelli. Model checking for process rewrite systems and a class of action-based regular properties. In *Proc. of VMCAI'05*, volume 3385 of *LNCS*, pages 282–297. Springer, 2005.

[Esp94] J. Esparza. On the decidability of model checking for several mu-calculi and Petri nets. In *CAAP*, volume 787 of *LNCS*, pages 115–129. Springer, 1994.

[Hab97] P. Habermehl. On the complexity of the linear-time μ-calculus for Petri nets. In *Proceedings of ICATPN'97*, volume 1248 of *LNCS*, pages 102–116. Springer–Verlag, 1997.

[KŘS04a] M. Křetínský, V. Řehák, and J. Strejček. Extended process rewrite systems: Expressiveness and reachability. In *Proceedings of CONCUR'04*, volume 3170 of *LNCS*, pages 355–370. Springer, 2004.

[KŘS04b] M. Křetínský, V. Řehák, and J. Strejček. On extensions of process rewrite systems: Rewrite systems with weak finite-state unit. In *Proceedings of INFINITY'03*, volume 98 of *ENTCS*, pages 75–88. Elsevier, 2004.

[KŘS05] M. Křetínský, V. Řehák, and J. Strejček. Reachability of Hennessy-Milner properties for weakly extended PRS. In *Proceedings of FSTTCS 2005*, volume 3821 of *LNCS*, pages 213–224. Springer, 2005.

[Lip76] R. Lipton. The reachability problem is exponential-space hard. Technical Report 62, Department of Computer Science, Yale University, 1976.

[Mai00] M. Maidl. The common fragment of CTL and LTL. In *Proc. 41th Annual Symposium on Foundations of Computer Science*, pages 643–652, 2000.

[May84] E. W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM Journal on Computing*, 13(3):441–460, 1984.

[May98] R. Mayr. *Decidability and Complexity of Model Checking Problems for Infinite-State Systems*. PhD thesis, Technische Universität München, 1998.

[May00] R. Mayr. Process rewrite systems. *Information and Computation*, 156(1):264–286, 2000.

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

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

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symposium on the Foundations of Computer Science*, pages 46–57, 1977.

[Str04] J. Strejček. *Linear Temporal Logic: Expressiveness and Model Checking*. PhD thesis, Faculty of Informatics, Masaryk University in Brno, 2004.