Acta Informatica (2009) 46:1–28
DOI 10.1007/s00236-008-0082-3

ORIGINAL ARTICLE

On decidability of LTL model checking for process rewrite systems

Laura Bozzelli · Mojmír Křetínský · Vojtěch Řehák · Jan Strejček

Received: 26 August 2007 / Revised: 23 June 2008 / Accepted: 14 October 2008 /
Published online: 11 November 2008

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) with both future and past operators. 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. We show that the problem is decidable for wPRS if we consider properties defined by LTL formulae with only modalities *strict eventually*, *strict always*, and their past counterparts. 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*.

1 Introduction

Automatic verification of current software systems often needs to model them as infinite-state systems. One of the most powerful formalisms for a finite description of infinite-state systems (except formalisms which are language equivalent to Turing machines) is called

Some of the results presented in this paper have been already published in [4,12].

L. Bozzelli
Dipartimento di Matematica e Apllicazioni, Università degli Studi di Napoli "Federico II",
Via Cintia, 80126 Naples, Italy
e-mail: laura.bozzelli@dma.unina.it

M. Křetínský · V. Řehák · J. Strejček (✉)
Faculty of Informatics, Masaryk University, Botanická 68a, 60200 Brno, Czech Republic
e-mail: strejcek@fi.muni.cz

M. Křetínský
e-mail: kretinsky@fi.muni.cz

V. Řehák
e-mail: rehak@fi.muni.cz

Process Rewrite Systems (PRS) [17]. 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 merit of PRS is that some important verification problems are decidable for the whole PRS class. In particular, Mayr [17] proved that the following problems are decidable for PRS:

– the *reachability problem*—whether a given state is reachable,
– the *reachable property problem*—whether there is a reachable state where some given actions are enabled and some given actions are disabled.

In [10], 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 language equivalent to Turing machines). This *weak* 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 [9] 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 [11]. Note that the latter problem is strictly more general than the reachable property problem. The hierarchy of all PRS and wPRS classes is depicted in Fig. 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 [16]. Here we focus exclusively on *Linear Temporal Logic* (LTL). It is known that LTL model checking of PDA is EXPTIME-complete [1]. LTL model checking of PN is also decidable, but at least as hard as the reachability problem for PN [5] (the reachability problem is EXPSPACE-hard [13,15] and no primitive recursive upper bound is known). If we consider only infinite runs, then the problem for PN is EXPSPACE-complete [8,16].

Conversely, LTL model checking is undecidable for all the classes subsuming PA [2,16]. So far, there are only two positive results for these classes. Bouajjani and Habermehl [2] have identified a fragment called *simple* $PLTL_{\Box}$ for which model checking of infinite runs is decidable for PA (strictly speaking, simple $\text{PLTL}_{\Box}$ is not a fragment of LTL as it can express also some non-regular properties, while LTL cannot). Only recently, Bozzelli [3] has demonstrated that model checking of infinite runs is decidable for PRS and the fragment of LTL capturing exactly fairness properties.

Our contribution. This paper contains several results on decidability of LTL model checking. In particular, we completely locate 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 LTL formulae containing only a given subset of standard temporal modalities and closed under boolean connectives. The boundary is depicted in Fig. 2. To locate the boundary, we demonstrate the following results.

1. We introduce a new LTL fragment $\mathcal{A}$. Then we prove that the problem whether a given wPRS has a (finite or infinite) run satisfying a given formula of $\mathcal{A}$ is decidable. The proof employs our results presented in [3,9,11] to reduce the problem to LTL model checking for PDA and PN. This result directly implies decidability of the model checking problem for wPRS and negated formulae of $\mathcal{A}$.
2. We show 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 negations of $\mathcal{A}$ formulae. This translation yields decidability of the model checking problem for wPRS and 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 define a past extension P$\mathcal{A}$ of the fragment $\mathcal{A}$. Using the result for $\mathcal{A}$, we show that the model checking problem for wPRS and negated formulae of P$\mathcal{A}$ remains decidable. Further, we prove that every formula of the basic fragment LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$) (LTL($\mathsf{F_s}, \mathsf{G_s}$) extended with the past counterparts of $\mathsf{F_s}$ and $\mathsf{G_s}$) can be effectively translated into P$\mathcal{A}$. Hence, we get decidability of the model checking problem for wPRS and LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$). We note that LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$) is strictly more expressive than LTL($\mathsf{F_s}, \mathsf{G_s}$) (for example, the formula $\mathsf{F_s}(b \wedge \mathsf{H_s}a)$ is not equivalent to any LTL($\mathsf{F_s}, \mathsf{G_s}$) formula) and semantically equivalent to First-Order Monadic Logic of Order restricted to 2 variables and without successor predicate ($\mathsf{FO}^2[<]$, see [6] for effective translations). Thus we also positively solve the model checking problem for wPRS and $\mathsf{FO}^2[<]$.
4. 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 also presents two results that are not connected to the decidability boundary.

5. We introduce a more general *pointed model checking problem* (whether all runs of a given wPRS system going through a given state satisfy a given formula in the given state). We show that this problem is decidable for wPRS and LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$).
6. Finally, we show that negated formulae of LTL$^{\text{det}}$ (the fragment known as 'the common fragment of CTL and LTL' [14]) can be effectively translated into $\mathcal{A}$. As a consequence we get that the model checking problem is decidable for wPRS and LTL$^{\text{det}}$.

Structure of the paper. The following section recalls basic definitions. Sections 3–6 correspond, respectively, to the first four items listed above. Section 5 also covers the results on the pointed model checking problem. Section 7 deals with the model checking problem for LTL$^{\text{det}}$. The last section summarizes our results and tries to give an intuitive explanation of the found decidability border location.

2 Preliminaries

2.1 PRS and weakly extended PRS

Let $Const = \{X, \ldots\}$ 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 $M = \{o, p, q, \ldots\}$ be a set of *control states*, $\leq$ be a partial ordering on this set, and $Act = \{a, b, c, \ldots\}$ 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 triple (R, p_0, t_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 $p, q \in M$ satisfy $p \leq q$,
- the pair $(p_0, t_0) \in M \times \beta$ forms the distinguished *initial state*.

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

A wPRS $\Delta = (R, p_0, t_0)$ induces a labelled transition system, whose states are pairs (p, t) such that $p \in M(\Delta)$ and t is a process term over $Const(\Delta)$. The transition relation $\longrightarrow_\Delta$ is the least relation satisfying the following inference rules:

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

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 *reachable from* a state $p't'$ if there exists a word u such that $p't' \xrightarrow{u} pt$. We say that a state is *reachable* if it is reachable from the initial state p_0t_0. Further, a state pt is called *terminal* if there is no state $p't'$ and no 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

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

is called *derivation over the word* $u = a_1a_2 \ldots a_n(a_{n+1} \ldots)$ *in* Δ. Finite derivations are also denoted as $p_1t_1 \xrightarrow{u}_\Delta p_{n+1}t_{n+1}$, infinite ones 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_0t_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*) [17]. 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 [17]. The relations between (α, β)-PRS and the mentioned formalisms and names are indicated in Fig. 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 the classes mentioned above, where expressive power of a class is measured by the set of transition systems that are definable (up to the strong bisimulation equivalence [18]) by the class. This hierarchy is strict, with a possible exception concerning the classes wPRS and PRS, where the strictness is just our conjecture. For details see [10].

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:

Parallel rules: $pX_1\|X_2\|\cdots\|X_n \overset{a}{\hookrightarrow} qY_1\|Y_2\|\cdots\|Y_m$
Sequential rules: $pX \overset{a}{\hookrightarrow} qY.Z \quad pX.Y \overset{a}{\hookrightarrow} qZ \quad pX \overset{a}{\hookrightarrow} qY \quad pX \overset{a}{\hookrightarrow} q\varepsilon$

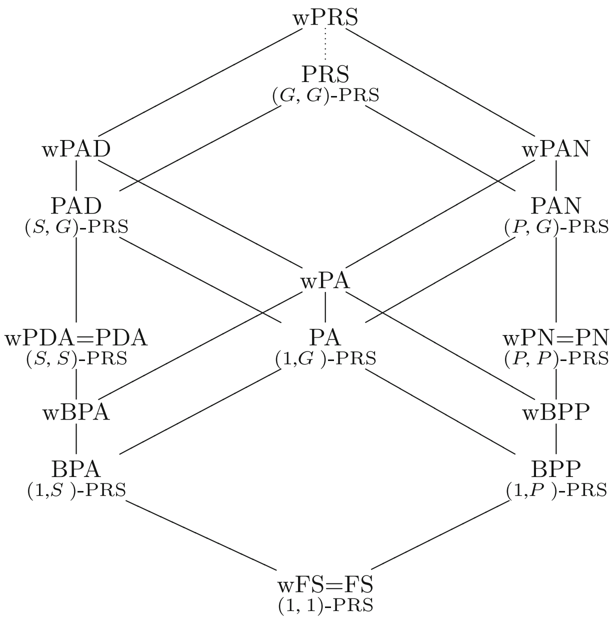

Fig. 1 The hierarchy of PRS and wPRS subclasses

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 $\Delta = (R, p_0, t_0)$ is in *normal form* if t_0 is a process constant and R contains only parallel and sequential rewrite rules.

PRS, wPRS, other extensions of PRS, and their respective subclasses are discussed in more detail in [21].

2.2 Linear temporal logic

The syntax of *Linear temporal logic* (LTL) [20] is defined as follows

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

where X and U are the future modal operators *next* and *until*, while Y and S are their past counterparts *previously* and *since*, and a ranges over Act. The logic is interpreted over infinite and nonempty finite *pointed* words of actions. Given a word $u = a_0a_1a_2\ldots \in \mathit{Act}^* \cup \mathit{Act}^\omega$, $|u|$ denotes the length of the word (we set $|u| = \infty$ if u is infinite). A *pointed word* is a pair (u, i) of a nonempty word u and a *position* $0 \leq i < |u|$ in this word.

The semantics of LTL formulae is defined inductively as follows:

$$\begin{array}{lll}
(u,i) \models tt & & \\
(u,i) \models a & \text{iff} & u = a_0a_1a_2\ldots \text{ and } a_i = a \\
(u,i) \models \neg\varphi & \text{iff} & (u,i) \not\models \varphi \\
(u,i) \models \varphi_1 \wedge \varphi_2 & \text{iff} & (u,i) \models \varphi_1 \text{ and } (u,i) \models \varphi_2 \\
(u,i) \models \mathsf{X}\varphi & \text{iff} & i+1 < |u| \text{ and } (u,i+1) \models \varphi \\
(u,i) \models \varphi_1\,\mathsf{U}\,\varphi_2 & \text{iff} & \exists k.\ (i \leq k < |u| \ \wedge \ (u,k) \models \varphi_2 \ \wedge \\
 & & \quad \wedge\ \forall j.\ (i \leq j < k \ \Rightarrow \ (u,j) \models \varphi_1))
\end{array}$$

$$
\begin{array}{lll}
(u,i) \models \mathsf{Y}\varphi & \text{iff} & 0 < i \text{ and } (u, i-1) \models \varphi \\
(u,i) \models \varphi_1 \,\mathsf{S}\, \varphi_2 & \text{iff} & \exists k.\ (0 \le k \le i \ \wedge\ (u,k) \models \varphi_2\ \wedge \\
& & \quad \wedge\ \forall j.\ (k < j \le i \ \Rightarrow\ (u,j) \models \varphi_1))
\end{array}
$$

We say that (u, i) *satisfies* φ whenever $(u, i) \models \varphi$. Further, a nonempty word u *satisfies* φ, written $u \models \varphi$, whenever $(u, 0) \models \varphi$. Given a set L of words, we write $L \models \varphi$ if $u \models \varphi$ holds for all $u \in L$. Finally, we say that a run σ of a wPRS Δ over a word u satisfies φ, written $\sigma \models \varphi$, whenever $u \models \varphi$.

Formulae φ, ψ are *(initially) equivalent*, written $\varphi \equiv_i \psi$, iff, for all words u, it holds that $u \models \varphi \iff u \models \psi$. Formulae φ, ψ are *globally equivalent*, written $\varphi \equiv \psi$, iff, for all pointed words (u, i), it holds that $(u, i) \models \varphi \iff (u, i) \models \psi$. Clearly, if two formulae are globally equivalent then they are also initially equivalent. Moreover, two formulae without past modalities are globally equivalent if and only if they are initially equivalent. Therefore we do not distinguish between initial and global equivalence when we talk about formulae without past.

The following table defines some derived future operators and their past counterparts.

Future modality		Meaning	Past modality		Meaning
$\mathsf{F}\varphi$	*Eventually*	$tt\,\mathsf{U}\,\varphi$	$\mathsf{P}\varphi$	*Eventually in the past*	$tt\,\mathsf{S}\,\varphi$
$\mathsf{G}\varphi$	*Always*	$\neg\mathsf{F}\neg\varphi$	$\mathsf{H}\varphi$	*Always in the past*	$\neg\mathsf{P}\neg\varphi$
$\mathsf{F_s}\varphi$	*Strict eventually*	$\mathsf{XF}\varphi$	$\mathsf{P_s}\varphi$	*Eventually in the strict past*	$\mathsf{YP}\varphi$
$\mathsf{G_s}\varphi$	*Strict always*	$\neg\mathsf{F_s}\neg\varphi$	$\mathsf{H_s}\varphi$	*Always in the strict past*	$\neg\mathsf{P_s}\neg\varphi$
$\overset{\infty}{\mathsf{F}}\varphi$	*Infinitely often*	$\mathsf{GF}\varphi$	$\mathsf{I}\varphi$	*Initially*	$\mathsf{HP}\varphi$

Given a set $\{O_1, \ldots, O_n\}$ of modalities, LTL$(O_1, \ldots, O_n)$ denotes the LTL fragment (closed under boolean connectives) containing all formulae with modalities $O_1, \ldots, O_n$ only. Such a fragment is called *basic* if either it contains future operators only, or for each included future operator, it contains its past counterpart and vice versa. For example, the fragment LTL(F, S) is not basic.

Figure 2 shows an expressiveness hierarchy of all studied basic LTL fragments. Indeed, every basic LTL fragment using standard[1] 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 an initially equivalent formula of the other fragment and vice versa. In particular, LTL$(\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s})$ is equivalent to LTL$(\mathsf{F_s}, \mathsf{P_s})$.[2] We also mind the result of [7] stating that each LTL formula can be converted into one which employs future operators only, i.e., LTL$(\mathsf{U}, \mathsf{X}) \equiv_i$ LTL$(\mathsf{U}, \mathsf{S}, \mathsf{X}, \mathsf{Y})$. The hierarchy is also strict: a solid line between two fragments indicates that every formula of the lower fragment is initially equivalent to some formula of the upper fragment, but the opposite relation does not hold. We refer to [22] for details about the expressiveness of LTL fragments.

[1] By standard modalities we mean the ones defined here 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.

[2] As $\mathsf{F_s}$, $\mathsf{G_s}$ and $\mathsf{P_s}$, $\mathsf{H_s}$ are pairs of dual operators, the fragments LTL$(\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s})$ and LTL$(\mathsf{F_s}, \mathsf{P_s})$ are in fact equivalent even with respect to the global equivalence.

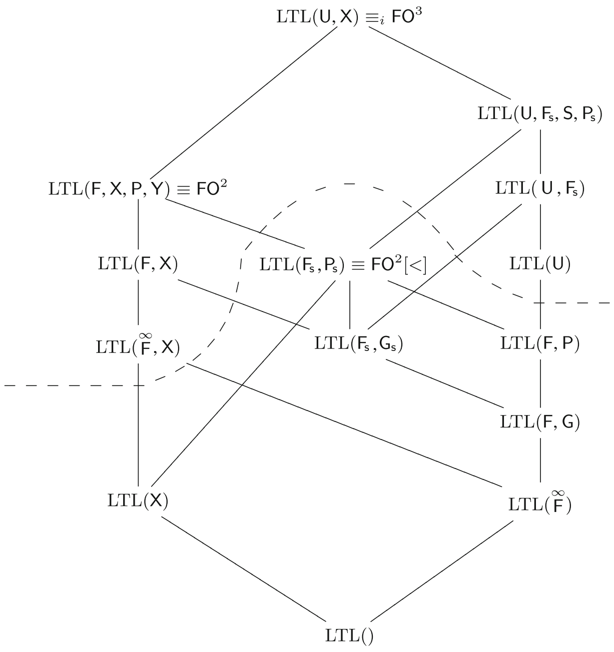

Fig. 2 The hierarchy of basic LTL fragments with respect to the initial equivalence. The *dashed line* shows the decidability boundary of the model checking problem for wPRS: the problem is decidable for all the fragments below the line, while it is undecidable for all the fragments above the line (even if we consider PA systems only)

2.3 Studied problems

Let $\mathcal{F}$ be an LTL fragment and $\mathcal{C}$ be a class of wPRS systems. This paper deals with the following three verification problems.

1. The *model checking problem* for $\mathcal{F}$ and $\mathcal{C}$ is to decide, for any given formula $\varphi \in \mathcal{F}$ and any given system $\Delta \in \mathcal{C}$, whether $L(\Delta) \models \varphi$ holds.
2. We also consider the problem called *model checking of infinite runs*, where $L(\Delta) \cap Act^{\omega} \models \varphi$ is examined.
3. The *pointed model checking problem* for $\mathcal{F}$ and wPRS is to decide whether a given formula $\varphi \in \mathcal{F}$, a given wPRS system Δ, and a given nonterminal state pt of Δ satisfy $L(pt, \Delta) \models \varphi$, where $L(pt, \Delta)$ is the set of all pointed words (u, i) such that Δ has a run $p_0t_0 \xrightarrow{a_0} p_1t_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{i-1}} p_it_i \xrightarrow{a_i} \ldots$ satisfying $u = a_0a_1a_2\ldots$ and $pt = p_it_i$.

3 Model checking for negated $\mathcal{A}$

This section starts with the definition of the LTL fragment $\mathcal{A}$. The rest of the section is devoted to decidability of the model checking problem for wPRS and negated formulae of this fragment.

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 $\varphi_1 \wedge \mathsf{X}(\varphi_1 \mathsf{U} \varphi_2)$.

Definition 1 Let $\delta = \theta_1 O_1 \theta_2 O_2 \dots \theta_n O_n \theta_{n+1}$, where $n > 0$, each $\theta_i \in \mathrm{LTL}()$, 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 $\mathcal{B} \subseteq \mathrm{LTL}()$ be a finite set. An *α-formula* is defined as

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

The fragment $\mathcal{A}$ consists of all finite disjunctions of α-formulae.

Hence, a word u satisfies $\alpha(\delta, \mathcal{B})$ iff u can be written as a concatenation $u_1.u_2.\cdots.u_{n+1}$, where each word 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 '$\wedge\, \mathsf{G_s}$',
- u_{n+1} satisfies $\mathsf{G_s F_s} \psi$ for every $\psi \in \mathcal{B}$.

In the following we use the fact that finite disjunctions of α-formulae are closed under conjunction.

Lemma 1 *A conjunction of α-formulae can be effectively converted into an equivalent disjunction of α-formulae.*

The proof is a straightforward but quite technical exercise, see [21] for some hints. To support an intuition, we provide an example of a conjunction of two simple α-formulae and an equivalent disjunction.

Example 1 A conjunction $\alpha(\theta_1 \,\mathsf{U}\, \theta_2 \wedge \mathsf{G_s}\theta_3, \mathcal{B}) \wedge \alpha(\theta'_1 \,\mathsf{U}\, \theta'_2 \wedge \mathsf{G_s}\theta'_3, \mathcal{B}')$ is equivalent to the following disjunction.

$$\begin{aligned}
&\alpha((\theta_1 \wedge \theta'_1) \,\mathsf{U}\, (\theta_2 \wedge \theta'_2) \quad \wedge \mathsf{G_s}(\theta_3 \wedge \theta'_3), \mathcal{B} \cup \mathcal{B}') \\
&\quad \vee \alpha((\theta_1 \wedge \theta'_1) \,\mathsf{U}\, (\theta_2 \wedge \theta'_1) \wedge \mathsf{X}(\theta_3 \wedge \theta'_1) \,\mathsf{U}\, (\theta_3 \wedge \theta'_2) \wedge \mathsf{G_s}(\theta_3 \wedge \theta'_3), \mathcal{B} \cup \mathcal{B}') \\
&\quad \vee \alpha((\theta_1 \wedge \theta'_1) \,\mathsf{U}\, (\theta_1 \wedge \theta'_2) \wedge \mathsf{X}(\theta_1 \wedge \theta'_3) \,\mathsf{U}\, (\theta_2 \wedge \theta'_3) \wedge \mathsf{G_s}(\theta_3 \wedge \theta'_3), \mathcal{B} \cup \mathcal{B}')
\end{aligned}$$

In order to show that the model checking problem for wPRS and negated formulae of $\mathcal{A}$ is decidable, 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 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 satisfied by 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 $\delta = \theta_1 O_1 \theta_2 O_2 \dots \theta_n O_n \theta_{n+1}$. We construct a wPRS system Δ' with control states $M(\Delta) \times \{1, 2, \dots, n + 1\}$ and the following four types of transition rules.

1. For any $1 \leq i \leq n$ and every rule $pt_1 \overset{a}{\hookrightarrow} qt_2$ of Δ such that an action a satisfies θ_i, we add the rule $(p, i)t_1 \overset{a}{\hookrightarrow} (q, i + 1)t_2$ to Δ'. Moreover, if O_i is U or U_+ then we also add the rule $(p, i)t_1 \overset{a}{\hookrightarrow} (q, i)t_2$.
2. Let e be a fresh action. For every $p \in M(\Delta)$, $X \in \mathit{Const}(\Delta)$, and for all i, $1 \leq i \leq n$, such that $O_i = \mathsf{U}$, we add the rule $(p, i)X \overset{e}{\hookrightarrow} (p, i + 1)X$ to Δ'.

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

Loosely speaking, the rules of type 1–3 allow Δ' to simulate all the runs of Δ which satisfy $\alpha(\delta, \emptyset)$. The rules of type 4 assure that a state $(p, n+1)t$ of Δ' is terminal if and only if the state pt of Δ is terminal.

Let $p_0 t_0$ be the initial state of Δ. There is a finite run $p_0 t_0 \xrightarrow{u}_{\Delta} qt$ satisfying $\alpha(\delta, \emptyset)$ if and only if there is a finite run $(p_0, 1)t_0 \xrightarrow{v}_{\Delta'} (q, n+1)t$. Hence, we need to decide whether there exists a state of the form $(q, n+1)t$ that is terminal and reachable from $(p_0, 1)t_0$. To that end, for every $p \in M(\Delta)$ we add to Δ' the rule $(p, n+1)Z \overset{end}{\hookrightarrow} (p, n+1)\varepsilon$, where $end \notin Act(\Delta)$ is a fresh action and $Z \notin Const(\Delta)$ 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 $(p_0, 1)(t_0 \| Z)$ and the only enabled action in this state is *end*. This last condition on the state can be expressed by formula $\varphi = \langle end \rangle tt \wedge \bigwedge_{a \in Act(\Delta)} \neg \langle a \rangle tt$ of the Hennessy–Milner logic. As reachability of a state satisfying a given Hennessy–Milner formula is decidable for wPRS (see [11] for details), we are done. □

The problem for infinite runs is more complicated. In order to solve it, we introduce more terminology and notation. At 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 $n \geq 0$, $a_1, \ldots, a_n \in Act$ are pairwise distinct actions and each O_i is either 'U_+' or '$\wedge \mathsf{X}$'. Further, let $B \subseteq Act \smallsetminus \{a_1, \ldots, a_n\}$ 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

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

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

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

Roughly speaking, a β-formula is a more restrictive version of an α-formula and in the 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 those actions occurring infinitely many times in the tail.

Remark 1 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 B, C, and $w = a_1 O_1 a_2 O_2 \ldots a_n O_n$ 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 $1 \leq i_1 < i_2 < \ldots < i_k \leq n$. 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 2 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 $tri(\Delta)$, $par(\Delta)$, and $seq(\Delta)$ we denote the system Δ restricted to trivial, parallel, and sequential rules, respectively. A derivation in $tri(\Delta)$ is called a *trivial* derivation in Δ. In what follows we write simply *tri*, *par*, *seq* as Δ is always clearly determined by the context.

Definition 2 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 $X, Y \in Const(\Delta)$, 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, results of [3], and decidability of LTL model checking for PDA and PN. Lemma 3 says that every PRS in normal form can be transformed into an "equivalent" flat system. Finally, Lemma 4 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 2 *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 two 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 [9].

(i) Let $w = a_1O_1 \dots a_nO_n$. We construct a wPRS Δ' with the set of control states $\{1, 2, \dots, n\} \cup 2^C$. Intuitively, control states $1, 2, \dots, n$ are used to check that the actions $a_1, a_2, \dots, a_n$ appear in the right order and quantity due to w, while the other actions are not allowed. After that, the control states in 2^C are used to check that every action in C appears at least once. The set of rewrite rules is defined as follows. For the sake of compactness, we use $(n + 1)$ as another name for the control state $\emptyset$.

- For every $1 \leq i \leq n$ 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 $O_i = \mathsf{U}_+$ 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, for some v, $1X \xrightarrow{v}_{\Delta'} C\varepsilon$ holds, we can decide Problem (ii). □

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

Lemma 3 *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' has an infinite derivation starting from X and satisfying $\beta(w', B', C')$ if and only if Δ has 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 2.

All the conditions of Definition 2 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 2. 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 [3]. Finally, Conditions 4 and 5 can be checked due to decidability of LTL model checking for PDA [1] and PN [5]. If there is a non-satisfied condition, we add some trivial rules forming the missing derivation.

Let us assume that Condition 3 (or 4 or 5, respectively) is not satisfied, i.e., there exists an infinite derivation $X \xrightarrow{u}$ (or $X \xrightarrow{u}_{par}$ or $X \xrightarrow{u}_{seq}$, respectively) satisfying $\beta(w', B', C')$ for some $(w', B', C') \sqsubseteq (w, B, C)$ and violating the condition. Remark 1 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 $\gamma(w', \{b_1, \ldots, b_j\})$.

We can effectively select such an ordering out of all orderings of B' using Lemma 2. Further, let $w' = a_1 O_1 a_2 O_2 \ldots a_n O_n$ and let $C' = \{c_1, c_2, \ldots, c_k\}$. 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 $Z_0 = X, Z_1, \ldots, Z_{n+m+k-1}$ 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 $w' = a_1 O_1 a_2 O_2 \ldots a_n O_n$ such that Condition 1 or 2 of Definition 2 is violated, then we first compute an ordering $\{b_1, \ldots, b_m\}$ 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 $Z_1, \ldots, Z_{n+m}$ 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 $X \xrightarrow{a_1 \ldots a_n b_1 \ldots b_m} Z_{n+m}$ 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 $(w', B', C') \sqsubseteq (w, B, C)$, the system 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 $B'' \subseteq B$. Let Δ' be the resulting PRS. We claim that 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 2. 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 3 (Subderivation) Let Δ be a PRS in normal form and σ_1 be a (finite or infinite) derivation $s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$, 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 the 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 3 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 independently on the rest of the term (as long as it contains a sequential composition).

Lemma 4 *Let Δ be a PRS in flat (w, B, C)-form. Then, for each $X \in \mathit{Const}(\Delta)$ and each $(w', B', C') \sqsubseteq (w, B, C)$, the following condition holds*:
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 with 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 with 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 with 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 a 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 2 implies that such an infinite derivation σ satisfies exactly one β-formula. We denote this β-formula by $\beta(\sigma)$. Further, by $SEQ(\sigma)$ 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 $SEQ(\sigma)$ 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 with 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 2 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 with 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 with a process constant has a trivial equivalent. For the sake of contradiction, we assume that the lemma does not hold. Let Σ be the nonempty set of infinite derivations violating the lemma and let $k = \min\{SEQ(\sigma) \mid \sigma \in \Sigma\}$.

First of all, we prove two claims.

Claim 1 Let σ be an infinite derivation satisfying $SEQ(\sigma) \leq k$. 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 $\sigma_1 = X \xrightarrow{a}_{seq} Y.Z \xrightarrow{b_1} t_1.Z \xrightarrow{b_2} t_2.Z \xrightarrow{b_3} \ldots$ 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 with 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 1 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 $\widetilde{SEQ}(\sigma) \geq SEQ(\sigma)$ as in $SEQ(\sigma)$ we do not count transition steps labelled with actions of B). 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 with 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 $B \subseteq Act \setminus \{c_1, \ldots, c_n\}$. 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.[3] 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 fulfilment of $\beta(\sigma)$ and the obtained derivation σ'' satisfies $\widetilde{SEQ}(\sigma'') < \widetilde{SEQ}(\sigma)$. □

With this claim, we can easily derive 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 this 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

[3] 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 $Y_2 \| t'' \xrightarrow{b} Y_2 \| t'''$.

consider the derivation

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

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 2 *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 Lemmas 3 and 4, 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 2 holds even for wPRS and α-formulae.

Theorem 3 *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 2. □

Theorem 4 *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 $e \notin Act(\Delta)$ be a fresh action. First of all, we describe our modification of the standard algorithm [17] that transforms Δ into a PRS in normal form.

Let t_0 be the initial state of Δ. If t_0 is not a process constant, we replace it by a fresh process constant X_0 and we add a rewrite rule $X_0 \overset{a}{\hookrightarrow} t$ for each action a and each term t such that $t_0 \xrightarrow{a}_\Delta t$. Note that the number of added rules is always finite.

If Δ is still 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 $Z_1, Z_2, Z \notin Const(\Delta)$ 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 $Z \notin Const(\Delta)$ 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 $Z_1, Z_2 \notin Const(\Delta)$ 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 $\mathcal{B}' = \mathcal{B} \cup \{\bigvee_{a \in Act(\Delta)} a\}$ and δ' arises from $\delta = \theta_1 O_1 \dots \theta_n O_n \xi$ by the following substitution for every i, $1 \leq i \leq n$.

- 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 3. □

Theorem 5 *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 the initial state $p_0 t_0$ and $\alpha(\delta, \mathcal{B})$ be an α-formula. We construct a PRS Δ' with the initial state t_0 which can simulate Δ. We also define a set of formulae recognizing correct simulations.

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

Further, we modify the formula $\alpha(\delta, \mathcal{B})$ in such a way 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 nonempty subset $\{p_1, p_2, \dots, p_k\} \subseteq M(\Delta)$ of control states satisfying $p_1 < p_2 < \dots < p_k$ and $p_1 = p_0$, we define an α-formula

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

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 $\varphi_{[p_1<p_2<\dots<p_k]}$ for some control states $p_1, p_2, \dots, p_k$ such that $p_1 < p_2 < \dots < p_k$ and $p_1 = p_0$. As the number of such sequences is finite and each $\varphi_{[p_1<p_2<\dots<p_k]}$ is an α-formula, Theorem 4 and Lemma 1 imply that the considered problem is decidable. □

Theorems 1 and 5 imply the following corollary.

Corollary 1 *The model checking problem for wPRS and negated formulae of $\mathcal{A}$ is decidable.*

4 Model checking for LTL($\mathsf{F_s}, \mathsf{G_s}$)

This section focuses on the fragment LTL($\mathsf{F_s}, \mathsf{G_s}$): we show that formulae of this fragment can be translated into $\mathcal{A}$ and thus the model checking problem for LTL($\mathsf{F_s}, \mathsf{G_s}$) and wPRS is decidable.

Theorem 6 *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 the disjunction of formulae in A_φ. Although our proof looks like by induction on the structure of φ, it is in fact by 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 equivalent to $p \wedge \mathsf{G_s}tt$. Hence $A_\varphi = \{\alpha(p \wedge \mathsf{G_s}tt, \emptyset)\}$.
- •$\vee$ *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, $A_\varphi = A_{\varphi_1} \cup A_{\varphi_2}$.
- •$\wedge$ *Case* $\varphi_1 \wedge \varphi_2$ Due to Lemma 1, the set A_φ can be constructed from the sets A_{φ_1} and A_{φ_2}.
- •$\mathsf{F_s}$ *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 $A_\varphi = \{\alpha(tt\, \mathsf{U}_+\, \delta, \mathcal{B}) \mid \alpha(\delta, \mathcal{B}) \in A_{\varphi_1}\}$.
- •$\mathsf{G_s}$ *Case* $\mathsf{G_s}\varphi_1$ This case is divided into the following subcases according to the structure of φ_1.
  - ∘p *Case* $\mathsf{G_s}p$ As $\mathsf{G_s}p$ is equivalent to $tt \wedge \mathsf{G_s}p$, we set $A_\varphi = \{\alpha(tt \wedge \mathsf{G_s}p, \emptyset)\}$.
  - ∘$\wedge$ *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 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 $\mathsf{G_s}(\varphi_2 \wedge \varphi_3)$.
  - ∘$\mathsf{F_s}$ *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 $p \in$ 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 $p_3 \wedge p_4 \in$ 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 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 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 satisfied only by finite words of length one. Further, a word u satisfies $(\mathsf{F_s}tt) \wedge (\mathsf{G_s}\mathsf{F_s}tt)$ iff u is infinite. Thus, $\mathsf{G_s}\mathsf{F_s}\mathsf{G_s}\varphi_3 \equiv (\mathsf{G_s}\neg tt) \vee \varphi'$ where $\varphi' = (\mathsf{F_s}tt) \wedge (\mathsf{G_s}\mathsf{F_s}tt) \wedge (\mathsf{F_s}\mathsf{G_s}\varphi_3)$.

Hence, $A_\varphi = A_{\mathsf{G_s}\neg tt} \cup A_{\varphi'}$ where $A_{\varphi'}$ is constructed from $A_{\mathsf{F_s}tt}$, $A_{\mathsf{G_s}\mathsf{F_s}tt}$, and $A_{\mathsf{F_s}\mathsf{G_s}\varphi_3}$ using Lemma 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.

$-p$ *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 $\mathsf{G_s}\,p$.

$-\wedge$ *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 1.

$-\mathsf{F_s}$ *Case* $\mathsf{G_s}(\varphi_2 \vee \mathsf{F_s}\varphi_4)$ It holds that $\mathsf{G_s}(\varphi_2 \vee \mathsf{F_s}\varphi_4) \equiv (\mathsf{G_s}\varphi_2) \vee \mathsf{F_s}(\varphi_4 \wedge \varphi_2 \wedge \mathsf{G_s}\varphi_2) \vee \mathsf{G_s}\mathsf{F_s}\varphi_4$. Therefore, the set A_φ can be constructed as $A_{\mathsf{G_s}\varphi_2} \cup \{\alpha(tt\,\mathsf{U}_+\,\delta, \mathcal{B}) \mid \alpha(\delta,\mathcal{B}) \in A_{\varphi_4\wedge\varphi_2\wedge\mathsf{G_s}\varphi_2}\} \cup A_{\mathsf{G_s}\mathsf{F_s}\varphi_4}$, where $A_{\varphi_4\wedge\varphi_2\wedge\mathsf{G_s}\varphi_2}$ is constructed from A_{φ_4}, A_{φ_2}, and $A_{\mathsf{G_s}\varphi_2}$ due to Lemma 1.

$-\mathsf{G_s}$ *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 $\mathsf{G_s}(\bigvee_{\varphi'\in G} \mathsf{G_s}\varphi') \equiv (\mathsf{G_s}\neg tt) \vee \bigvee_{\varphi'\in G}(\mathsf{X}\mathsf{G_s}\varphi')$. 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 $\mathsf{G_s}(p_2 \vee \bigvee_{\varphi'\in G} \mathsf{G_s}\varphi') \equiv (\mathsf{G_s}\,p_2) \vee \bigvee_{\varphi'\in G}(\mathsf{X}(p_2\,\mathsf{U}\,\mathsf{G_s}\varphi'))$, 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 $\mathsf{G_s}(\mathsf{G_s}\varphi_2) \equiv (\mathsf{G_s}\neg tt) \vee (\mathsf{X}\mathsf{G_s}\varphi_2)$, the set A_φ can be constructed as $A_{\mathsf{G_s}\neg tt} \cup \{\alpha(tt \wedge \mathsf{X}\delta, \mathcal{B}) \mid \alpha(\delta,\mathcal{B}) \in A_{\mathsf{G_s}\varphi_2}\}$. □

As LTL($\mathsf{F_s}$, $\mathsf{G_s}$) is closed under negation, Theorem 6 and Corollary 1 give us the following.

Corollary 2 *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) and PN [8]. Our decidability proof does not provide any primitive recursive upper bound as it employs reachability for PN (for example, it is used in a decision procedure for reachability for wPRS [9]), for which no primitive recursive upper bound is known.

5 Model checking for LTL($\mathsf{F_s}$, $\mathsf{G_s}$, $\mathsf{P_s}$, $\mathsf{H_s}$)

This section extends the results of the previous two sections to handle past modalities *eventually in the strict past* and *always in the strict past* as well.

We start with a past extension of α-formulae called Pα-formulae. Intuitively, a Pα-formula is a conjunction of an α-formula and a past version of the α-formula.

A formal definition of a Pα-formula makes use of $\varphi_1\,\mathsf{S}_+\,\varphi_2$ to abbreviate $\varphi_1 \wedge \mathsf{Y}(\varphi_1\,\mathsf{S}\,\varphi_2)$.

Definition 4 Let $\eta = \iota_1 P_1 \iota_2 P_2 \ldots \iota_m P_m \iota_{m+1}$, where $m > 0$, each $\iota_j \in$ LTL(), P_m is '$\wedge\,\mathsf{H_s}$', and, for each $j < m$, P_j is either 'S' or 'S_+' or '$\wedge\,\mathsf{Y}$'. Further, let $\alpha(\delta,\mathcal{B})$ be an α-formula. Then a Pα*-formula* is defined as

$$\mathrm{P}\alpha(\eta,\delta,\mathcal{B}) = (\iota_1 P_1(\iota_2 P_2 \ldots (\iota_m P_m \iota_{m+1})\ldots)) \wedge \alpha(\delta,\mathcal{B})\,.$$

The fragment P$\mathcal{A}$ consists of all finite disjunctions of Pα-formulae.

Note that the definition of a Pα-formula does not contain any past counterpart of $\wedge_{\psi\in\mathcal{B}}\mathsf{G_S}\mathsf{F_S}\psi$ as every history is finite.

Therefore, a pointed word (u, k), where $u = a_0a_1a_2\ldots$, satisfies P$\alpha(\eta, \delta, \mathcal{B})$ if and only if $a_0a_1\ldots a_k$ can be written as a concatenation $v_{m+1}.v_m.\cdots.v_2.v_1$, where each word v_i consists only of actions satisfying ι_i and

- $|v_i| \geq 0$ if $i = m + 1$ or P_i is 'S',
- $|v_i| > 0$ if P_i is 'S_+',
- $|v_i| = 1$ if P_i is '$\wedge\,\mathsf{Y}$' or '$\wedge\,\mathsf{H_S}$'.

The following lemma says that the fragment P$\mathcal{A}$ is 'semantically closed' under conjunction and application of some temporal operators. As in the case of Lemma 1, the proof is intuitively clear but some parts are quite technical. We refer to [21] for some hints.

Lemma 5 *Let φ be a* Pα*-formula and $p \in LTL()$. Formulae $\mathsf{X}\varphi$, $\mathsf{Y}\varphi$, $p\,\mathsf{U}\,\varphi$, $p\,\mathsf{S}\,\varphi$, $\mathsf{F_S}\varphi$, $\mathsf{P_S}\varphi$, and also any conjunction of* Pα*-formulae can be effectively converted into a globally equivalent disjunction of* Pα*-formulae.*

The next step is to show that we can decide whether a given wPRS system has a run satisfying a given Pα-formula. The proof utilizes Corollary 1.

Theorem 7 *The problem whether a given wPRS system has a run satisfying a given* Pα*-formula is decidable.*

Proof A run over a nonempty (finite or infinite) word $u = a_0a_1a_2\ldots$ satisfies a formula φ iff $(u, 0) \models \varphi$. Moreover, $(u, 0) \models$ P$\alpha(\eta, \delta, \mathcal{B})$ iff $(a_0, 0) \models \eta$ and $(u, 0) \models \alpha(\delta, \mathcal{B})$. Let $\eta = \iota_1P_1\iota_2P_2\ldots\iota_mP_m\iota_{m+1}$. It follows from the semantics of LTL that $(a_0, 0) \models \eta$ if and only if $(a_0, 0) \models \iota_m$ and $P_i = \mathsf{S}$ for all $i < m$. Therefore, the problem is to check whether $P_i = \mathsf{S}$ for all $i < m$ and whether the given wPRS system has a run satisfying $\iota_m \wedge \alpha(\delta, \mathcal{B})$. As $\iota_m \wedge \alpha(\delta, \mathcal{B})$ can be easily translated into a disjunction of α-formulae, Corollary 1 finishes the proof. □

It remains to show that every LTL($\mathsf{F_S}, \mathsf{G_S}, \mathsf{P_S}, \mathsf{H_S}$) formula can be translated into a P$\mathcal{A}$ formula. The proof uses the same approach as the one of Theorem 6: it proceeds by a thorough analysis of the structure of a translated formula. The full proof is in Appendix A.

Theorem 8 *Every LTL($\mathsf{F_S}, \mathsf{G_S}, \mathsf{P_S}, \mathsf{H_S}$) formula φ can be translated into a globally equivalent disjunction of* Pα*-formulae.*

As LTL($\mathsf{F_S}, \mathsf{G_S}, \mathsf{P_S}, \mathsf{H_S}$) is closed under negation, Theorems 8 and 7 give us the following.

Corollary 3 *The model checking problem for wPRS and LTL($\mathsf{F_S}, \mathsf{G_S}, \mathsf{P_S}, \mathsf{H_S}$) is decidable.*

Moreover, we can show that the pointed model checking problem is decidable for wPRS and LTL($\mathsf{F_S}, \mathsf{G_S}, \mathsf{P_S}, \mathsf{H_S}$) as well. Again, we solve the dual problem for Pα-formulae.

Theorem 9 *Let Δ be a wPRS and pt be a reachable nonterminal state of Δ. The problem whether $L(pt, \Delta)$ contains a pointed word (u, i) satisfying a given* Pα*-formula is decidable.*

Proof Let $\Delta = (R, p_0, t_0)$ be a wPRS and pt be a reachable nonterminal state of Δ. We construct a wPRS $\Delta' = (R', p_0, t_0.X)$ where $X \notin Const(\Delta)$ is a fresh process constant,

$$R' = R \cup \{(p(t.X) \overset{a}{\hookrightarrow} pX_a), (pX_a \overset{f}{\hookrightarrow} pY_a), (pY_a \overset{a}{\hookrightarrow} p't') \mid pt \xrightarrow{a} p't'\},$$

$f \notin Act(\Delta)$ is a fresh action, and $X_a, Y_a \notin Const(\Delta)$ are fresh process constants for each $a \in Act(\Delta)$.

Let $u = a_0a_1a_2\ldots$ be a word. It is easy to see that (u, i) is in $L(pt, \Delta)$ iff $a_0a_1\ldots a_{i-1}a_i.f.a_i.a_{i+1}\ldots$ is in $L(\Delta')$. Hence, for any given Pα-formula $\varphi = \mathrm{P}\alpha(\eta, \delta, \mathcal{B})$ we construct a Pα-formula $\varphi' = \mathrm{P}\alpha(\eta, tt \wedge \mathsf{X}f \wedge \mathsf{X}\delta, \mathcal{B})$. We get that

$$L(pt, \Delta) \models \mathrm{P}\alpha(\eta, \delta, \mathcal{B}) \iff L(\Delta') \models \mathsf{F}(\mathrm{P}\alpha(\eta, tt \wedge \mathsf{X}f \wedge \mathsf{X}\delta, \mathcal{B}))$$

and due to Lemma 5 and Theorem 7 the proof is done. □

As LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$) is closed under negation and Theorem 8 works with global equivalence, Theorem 9 gives us the following.

Corollary 4 *The pointed model checking problem is decidable for wPRS and LTL*($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$).

6 Undecidability results

Obviously, the model checking for wPRS and LTL(X) is decidable. Hence, to show that the decidability boundary of Fig. 2 is drawn correctly, we have to prove that the model checking problem is undecidable for wPRS and the fragments LTL(U) and LTL($\overset{\infty}{\mathsf{F}}, X$). In fact, we show that the problem is undecidable even for the subclass of PA systems and the mentioned LTL fragments. The undecidability proofs are based on reductions from the non-halting problem for Minsky 2-counter machines, which is known to be undecidable [19].

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

– *increment*: $\mathtt{c}_k$`:=`$\mathtt{c}_k$`+1; goto` l_r, or
– *test-and-decrement*: `if` $\mathtt{c}_k$`>0 then` $\mathtt{c}_k$`:=`$\mathtt{c}_k$`-1; goto` l_r `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 $\mathtt{c}_1$ and $\mathtt{c}_2$, respectively.

We say that the machine N *halts* if $(l_1, 0, 0) \longrightarrow^* (l_n, v_1, v_2)$ for some numbers $v_1, v_2 \geq 0$, where $\longrightarrow^*$ denotes the reflexive and transitive closure of $\longrightarrow$. The *non-halting problem* is to decide whether a given machine N does not halt. The problem is undecidable [19].

Theorem 10 *Model checking of PA against LTL*(U) *is undecidable.*

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

– for every instruction l_i : $\mathtt{c}_k$`:=`$\mathtt{c}_k$`+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` $\mathtt{c}_k$`>0 then` $\mathtt{c}_k$`:=`$\mathtt{c}_k$`-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 the instruction l_n : `halt`.

Now, we define a formula ψ describing a correct step of the constructed PA system Δ_N when simulating the machine N. The formula ψ is the following conjunction:

$$\begin{array}{ll}\bigwedge_{\text{each } l_i:\texttt{c}_k\texttt{:=c}_k\texttt{+1; goto } l_r} ((l_i \Longrightarrow (l_i \,\mathsf{U}\, inc_i)) \wedge (inc_i \Longrightarrow (inc_i \,\mathsf{U}\, l_r))) \wedge & \\ \bigwedge_{\text{each } l_i:\texttt{if c}_k\texttt{>0 then c}_k\texttt{:=c}_k\texttt{-1; goto } l_r \texttt{ else goto } l_s} & ((l_i \Longrightarrow (l_i \,\mathsf{U}\, (dec_i \vee zero_i))) \\ & \wedge\, (dec_i \Longrightarrow (dec_i \,\mathsf{U}\, l_r)) \\ & \wedge (zero_i \Longrightarrow (zero_i \,\mathsf{U}\, l_s)))\end{array}$$

Finally, we set $\varphi = l_1 \wedge (\psi \,\mathsf{U}\, l_n)$. It is easy to see that the 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 $L(\Delta_N) \models \neg\varphi$. □

Theorem 11 *Model checking of PA against LTL($\overset{\infty}{\mathsf{F}}, \mathsf{X}$) 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 state $D_1 \| D_2 \| H$ and set of rules containing

- for every instruction l_i : $\texttt{c}_k\texttt{:=c}_k\texttt{+1; goto }l_r$, 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 : $\texttt{if c}_k\texttt{>0 then c}_k\texttt{:=c}_k\texttt{-1; goto }l_r\texttt{ else goto }l_s$, 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

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

- 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 system Δ_N when simulating the machine N. The formula ψ is the following conjunction:

$$\begin{array}{ll}\bigwedge_{\text{each } l_i:\texttt{c}_k\texttt{:=c}_k\texttt{+1; goto } l_r} ((l_i \Longrightarrow \mathsf{X} inc_i) \wedge (inc_i \Longrightarrow \mathsf{X} l_r)) \wedge & \\ \bigwedge_{\text{each } l_i:\texttt{if c}_k\texttt{>0 then c}_k\texttt{:=c}_k\texttt{-1; goto } l_r \texttt{ else goto } l_s} & ((l_i \Longrightarrow \mathsf{X}(dec_i \vee zero_i)) \\ & \wedge\, (dec_i \Longrightarrow \mathsf{X} l_r) \\ & \wedge (zero_i \Longrightarrow \mathsf{X} l_s)) \\ \wedge (l_n \Longrightarrow \mathsf{X} halt) & \end{array}$$

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

$$\begin{array}{rcll}\rho & = & (halt \Longrightarrow \mathsf{X}(del_1 \vee reset_1)) & \wedge\ (del_1 \Longrightarrow \mathsf{X}(del_1 \vee reset_1)) \\ & \wedge & (reset_1 \Longrightarrow \mathsf{X}(del_2 \vee reset_2)) & \wedge\ (del_2 \Longrightarrow \mathsf{X}(del_2 \vee reset_2)) \\ & \wedge & (reset_2 \Longrightarrow \mathsf{X} l_1) & \end{array}$$

The formula $\varphi = \overset{\infty}{\mathsf{G}}(\psi \wedge \rho) \wedge \overset{\infty}{\mathsf{F}} halt$ 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 $L(\Delta_N) \models \neg\varphi$. □

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(U) and LTL($\overset{\infty}{\mathsf{F}}, \mathsf{X}$).

It can be easily shown that model checking of finite runs for PA and LTL(U) is undecidable as well. To that end, it suffices to modify the construction in the proof of Theorem 10 by adding a rule $X \overset{e}{\hookrightarrow} \varepsilon$ for every $X \in \{H, C_1, D_1, S_1, C_2, D_2, S_2\}$.

In contrast, model checking of *finite runs* for LTL($\overset{\infty}{\mathsf{F}}, \mathsf{X}$) 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($\overset{\infty}{\mathsf{F}}, \mathsf{X}$) 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 for LTL($\overset{\infty}{\mathsf{F}}, \mathsf{X}$) follows from decidability of the *reachability Hennessy–Milner property* problem [11].

7 Model checking for LTL$^{\text{det}}$

This section deals with the LTL$^{\text{det}}$ fragment also known as 'the common fragment of CTL and LTL' [14]. Using our results of Sect. 3 we show that the model checking problem for wPRS and this fragment is decidable. A definition of LTL$^{\text{det}}$ employs a binary modality *weak until*, denoted with W, with the meaning $\varphi \,\mathsf{W}\, \psi \;\equiv\; \mathsf{G}\varphi \,\vee\, \varphi \,\mathsf{U}\, \psi$.

Definition 5 Let $Act = \{a, b, \ldots\}$ be a countably infinite set of atomic actions. The syntax of *LTL^{det} formula* is defined as follows.

$$\begin{aligned}\varphi ::= p \mid \varphi_1 \wedge \varphi_2 \mid (p \wedge \varphi_1) \vee (\neg p \wedge \varphi_2) \mid \mathsf{X}\varphi_1 \mid \\ (p \wedge \varphi_1)\,\mathsf{U}\,(\neg p \wedge \varphi_2) \mid (p \wedge \varphi_1)\,\mathsf{W}\,(\neg p \wedge \varphi_2),\end{aligned}$$

where p ranges over LTL().

Note that LTL$^{\text{det}}$ is not closed under application of negation. To prove the decidability of model checking for wPRS an LTL$^{\text{det}}$, we show that the *negation* of every LTL$^{\text{det}}$ formula can be converted into an equivalent disjunction of α-formulae.

Theorem 12 *A negation of every LTL^{det} formula can be translated into an equivalent disjunction of α-formulae.*

Proof Given an LTL$^{\text{det}}$ formula φ, we construct a finite set $A_{\neg\varphi}$ of α-formulae such that $\neg\varphi$ is equivalent to the disjunction of formulae in $A_{\neg\varphi}$. The proof uses the following equivalences.

$$\mathsf{G_s} tt \equiv tt \tag{1}$$

$$\neg\mathsf{X}\varphi \equiv \mathsf{G_s}\neg tt \vee \mathsf{X}\neg\varphi \tag{2}$$

The formula $\mathsf{G_s}\neg tt$ occurring in the second equivalence is satisfied only by words of length 1. These words satisfy also every formula of the form $\neg\mathsf{X}\varphi$, but no formula of the form $\mathsf{X}\neg\varphi$.

The proof is by induction on the structure of φ. The formula has one of the following forms:

- $\bullet p$ *Case* p Using (1), we get that $\neg p \equiv \neg p \wedge \mathsf{G_s} tt$. Hence, we define $A_{\neg\varphi} = \{\alpha(\neg p \wedge \mathsf{G_s} tt, \emptyset)\}$.
- $\bullet\mathsf{X}$ *Case* $\mathsf{X}\varphi_1$ Using (2), we get that $\neg\mathsf{X}\varphi_1 \equiv \mathsf{G_s}\neg tt \vee \mathsf{X}\neg\varphi_1$. Hence, we set $A_{\neg\varphi} = \{\alpha(tt \wedge \mathsf{G_s}\neg tt, \emptyset)\} \cup \{\alpha(tt \wedge \mathsf{X}\delta, \mathcal{B}) \mid \alpha(\delta, \mathcal{B}) \in A_{\neg\varphi_1}\}$.
- $\bullet\wedge$ *Case* $\varphi_1 \wedge \varphi_2$ Clearly, we set $A_{\neg\varphi} = A_{\neg\varphi_1} \cup A_{\neg\varphi_2}$.
- $\bullet\vee$ *Case* $(p \wedge \varphi_1) \vee (\neg p \wedge \varphi_2)$ We obtain $A_{\neg\varphi}$ from the set of conjunctions $\{\alpha(\delta_1, \mathcal{B}_1) \wedge \alpha(\delta_2, \mathcal{B}_2) \mid \alpha(\delta_1, \mathcal{B}_1) \in A_{\neg(p\wedge\varphi_1)}$ and $\alpha(\delta_2, \mathcal{B}_2) \in A_{\neg(\neg p\wedge\varphi_2)}\}$ using Lemma 1.
- $\bullet\,\mathsf{U}$ *Case* $(p \wedge \varphi_1)\,\mathsf{U}\,(\neg p \wedge \varphi_2)$ As $\neg((p \wedge \varphi_1)\,\mathsf{U}\,(\neg p \wedge \varphi_2)) \equiv p\,\mathsf{W}\,((p \wedge \neg\varphi_1) \vee (\neg p \wedge \neg\varphi_2)) \equiv \mathsf{G}p \vee p\,\mathsf{U}\,((p \wedge \neg\varphi_1) \vee (\neg p \wedge \neg\varphi_2)) \equiv (p \wedge \mathsf{G_s} p) \vee p\,\mathsf{U}\,(\neg((\neg p \vee \varphi_1) \wedge (p \vee \varphi_2)))$, the construction can be done as follows. Applying the previous constructions, we obtain $A' = A_{\neg((\neg p\vee\varphi_1)\wedge(p\vee\varphi_2))}$. Now, $A_{\neg\varphi}$ can be defined as $\{\alpha(p \wedge \mathsf{G_s} p, \emptyset)\} \cup \{\alpha(p\,\mathsf{U}\,\delta, \mathcal{B}) \mid \alpha(\delta, \mathcal{B}) \in A'\}$.
- $\bullet\,\mathsf{W}$ *Case* $(p \wedge \varphi_1)\,\mathsf{W}\,(\neg p \wedge \varphi_2)$ Similarly to the case $(p \wedge \varphi_1)\,\mathsf{U}\,(\neg p \wedge \varphi_2)$, we get $\neg((p \wedge \varphi_1)\,\mathsf{W}\,(\neg p \wedge \varphi_2)) \equiv p\,\mathsf{U}\,(\neg((\neg p \vee \varphi_1) \wedge (p \vee \varphi_2)))$. Therefore, $A_{\neg\varphi}$ can be constructed as $\{\alpha(p\,\mathsf{U}\,\delta, \mathcal{B}) \mid \alpha(\delta, \mathcal{B}) \in A_{\neg((\neg p\vee\varphi_1)\wedge(p\vee\varphi_2))}\}$. □

The previous theorem and Corollary 1 give us the following.

Corollary 5 *The model checking problem for wPRS and LTL^{det} is decidable.*

8 Conclusion

The paper brings several new (un)decidability results on model checking of wPRS classes and fragments of LTL with both future and past modalities (see Fig. 2). In particular, we have established the decidability border of the problem for basic LTL fragments by showing that it is decidable for wPRS and LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$), but it is undecidable even for PA and LTL(U) or LTL($\overset{\infty}{\mathsf{F}}, \mathsf{X}$). It is known that the problem is decidable for all wPRS classes not subsuming PA (i.e., pushdown processes, Petri nets, and all their subclasses) and the whole LTL.

Now we try to provide some intuitive explanations of the decidability boundary location. Going through the paper, one can verify that every formula of LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$) can be translated into an initially equivalent disjunction of α-formulae. Hence, the model checking problem for LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$) reduces to the problem whether a given wPRS system has a run satisfying a given α-formula. Every α-formula $\alpha(\delta, \mathcal{B})$ (see Definition 1) consists of two parts. The first part, corresponding to $\alpha(\delta, \emptyset)$, can be translated into a *1-weak automaton* (also called *very weak automaton* – an automaton without cycles except of self loops). The problem of existence of a run accepted by such an automaton reduces to the reachability problem for wPRS, which is decidable due to [9]. The second part is a conjunction of formulae of the form $\mathsf{G_s}\mathsf{F_s}\psi$, i.e., a fairness condition. Such a fairness condition corresponds to an automaton that is not 1-weak. Fortunately, there is a result of [3] saying that the problem whether a PRS has an infinite run satisfying a given fairness condition is decidable. These observations support an intuition for decidability of the model checking problem for wPRS and LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$).

Looking at the decidability border passing between LTL($\overset{\infty}{\mathsf{F}}$) and LTL($\overset{\infty}{\mathsf{F}}, \mathsf{X}$), one may naturally ask whether the X operator causes undecidability. Let us note that the X operator does not lead to undecidability in general. For example, α-formulae employs next operators too. The proof showing undecidability of model checking for LTL($\overset{\infty}{\mathsf{F}}, \mathsf{X}$) contains an LTL formula where the X operator is nested in the left argument of an U operator. Similarly, in the case of the undecidability proof for LTL(U), the constructed formula employs U operator nested in the left argument of another U operator. These are quintessential LTL constructions leading to (non-self) loops in the corresponding automata. That is why our decidability proof cannot work for these fragments.

Acknowledgments We would like to thank an anonymous referee for valuable comments. The authors have been partially supported as follows: M. Křetínský by Ministry of Education of the Czech Republic, project No. MSM 0021622419, and by the Czech Science Foundation, grant No. 201/06/1338. V. Řehák by the Research Centre *Institute for Theoretical Computer Science (ITI)*, project No. 1M0545. J. Strejček by the Czech Science Foundation, grant No. 201/08/P375. A preliminary version of this paper has been written during L. Bozzelli's postdoc stay in LSV, CNRS & ENS Cachan and J. Strejček's postdoc stay in LaBRI, Université Bordeaux 1.

Appendix A: Proof of Theorem 8

Theorem 8 Every LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$) formula φ can be translated into a globally equivalent disjunction of $\mathrm{P}\alpha$-formulae.

Proof As $\mathsf{F_s}, \mathsf{G_s}$ and $\mathsf{P_s}, \mathsf{H_s}$ are dual modalities, we can assume that φ is an LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$) formula containing negations in front of actions only. We construct a finite set A_φ of $\mathrm{P}\alpha$-formulae such that φ is globally equivalent to the disjunction of formulae in A_φ. As in the case of Theorem 6, the proof is done by induction on the length of φ. Thus, if $\varphi \notin$ LTL(), then we assume that, for each LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$) formula φ' shorter than φ, we can construct the corresponding set $A_{\varphi'}$. Let p be a formula of LTL(). The structure of φ fits into one of the following cases.

- $\bullet p$ *Case* p In this case, φ is equivalent to $p \wedge \mathsf{G_s}tt$. Hence $A_\varphi = \{\mathrm{P}\alpha(tt \wedge \mathsf{H_s}tt, p \wedge \mathsf{G_s}tt, \emptyset)\}$.
- $\bullet\vee$ *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 5, A_φ can be constructed from the sets A_{φ_1} and A_{φ_2}.
- $\bullet\mathsf{F_s}$ *Case* $\mathsf{F_s}\varphi_1$ Due to Lemma 5, the set A_φ can be constructed from the set A_{φ_1}.
- $\bullet\mathsf{P_s}$ *Case* $\mathsf{P_s}\varphi_1$ Due to Lemma 5, the set A_φ can be constructed from the set A_{φ_1}.
- $\bullet\mathsf{G_s}$ *Case* $\mathsf{G_s}\varphi_1$ 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 $A_\varphi = \{\mathrm{P}\alpha(tt \wedge \mathsf{H_s}tt, tt \wedge \mathsf{G_s}p, \emptyset)\}$.
  - $\circ\wedge$ *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 5. 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 $p \in$ LTL(), we directly set $A_\varphi = \{\mathrm{P}\alpha(tt \wedge \mathsf{H_s}tt, tt \wedge \mathsf{G_s}tt, \{p\})\}$.
    - $-\vee$ *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 $p_3 \wedge p_4 \in$ 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 5.

$*\mathsf{P_s}$ *Case* $\mathsf{G_sF_s}(\varphi_3 \wedge \mathsf{P_s}\varphi_5)$ As $\mathsf{G_sF_s}(\varphi_3 \wedge \mathsf{P_s}\varphi_5) \equiv (\mathsf{G_sF_s}\varphi_3) \wedge (\mathsf{G_sF_sP_s}\varphi_5)$, the set A_φ can be constructed from $A_{\mathsf{G_sF_s}\varphi_3}$ and $A_{\mathsf{G_sF_sP_s}\varphi_5}$ using Lemma 5.

$*\mathsf{G_s}$ *Case* $\mathsf{G_sF_s}(\varphi_3 \wedge \mathsf{G_s}\varphi_5)$ As $\mathsf{G_sF_s}(\varphi_3 \wedge \mathsf{G_s}\varphi_5) \equiv (\mathsf{G_sF_s}\varphi_3) \wedge (\mathsf{G_sF_sG_s}\varphi_5)$, the set A_φ can be constructed from $A_{\mathsf{G_sF_s}\varphi_3}$ and $A_{\mathsf{G_sF_sG_s}\varphi_5}$ using Lemma 5.

$*\mathsf{H_s}$ *Case* $\mathsf{G_sF_s}(\varphi_3 \wedge \mathsf{H_s}\varphi_5)$ As $\mathsf{G_sF_s}(\varphi_3 \wedge \mathsf{H_s}\varphi_5) \equiv (\mathsf{G_sF_s}\varphi_3) \wedge (\mathsf{G_sF_sH_s}\varphi_5)$, the set A_φ can be constructed from $A_{\mathsf{G_sF_s}\varphi_3}$ and $A_{\mathsf{G_sF_sH_s}\varphi_5}$ using Lemma 5.

$-\mathsf{F_s}$ *Case* $\mathsf{G_sF_sF_s}\varphi_3$ As $\mathsf{G_sF_sF_s}\varphi_3 \equiv \mathsf{G_sF_s}\varphi_3$, we set $A_\varphi = A_{\mathsf{G_sF_s}\varphi_3}$.

$-\mathsf{P_s}$ *Case* $\mathsf{G_sF_sP_s}\varphi_3$ A pointed word (u, i) satisfies $\mathsf{G_sF_sP_s}\varphi_3$ iff $i = |u| - 1$ or u is an infinite word satisfying $\mathsf{F}\varphi_3$. Note that $\mathsf{G_s}\neg tt$ is satisfied only by finite words at their last position. Further, a word u satisfies $(\mathsf{F_s}tt) \wedge (\mathsf{G_sF_s}tt)$ iff u is infinite. Thus, $\mathsf{G_sF_sP_s}\varphi_3 \equiv (\mathsf{G_s}\neg tt) \vee \varphi'$ where $\varphi' = (\mathsf{F_s}tt) \wedge (\mathsf{G_sF_s}tt) \wedge (\varphi_3 \vee \mathsf{P_s}\varphi_3 \vee \mathsf{F_s}\varphi_3)$. Hence, $A_\varphi = A_{\mathsf{G_s}\neg tt} \cup A_{\varphi'}$ where $A_{\varphi'}$ is constructed from $A_{\mathsf{F_s}tt}$, $A_{\mathsf{G_sF_s}tt}$, and $A_{\varphi_3} \cup A_{\mathsf{P_s}\varphi_3} \cup A_{\mathsf{F_s}\varphi_3}$ using Lemma 5.

$-\mathsf{G_s}$ *Case* $\mathsf{G_sF_sG_s}\varphi_3$ A pointed word (u, i) satisfies $\mathsf{G_sF_sG_s}\varphi_3$ iff $i = |u| - 1$ or u is an infinite word satisfying $\mathsf{F_sG_s}\varphi_3$. Thus, $\mathsf{G_sF_sG_s}\varphi_3 \equiv (\mathsf{G_s}\neg tt) \vee \varphi'$ where $\varphi' = (\mathsf{F_s}tt) \wedge (\mathsf{G_sF_s}tt) \wedge (\mathsf{F_sG_s}\varphi_3)$. Hence, $A_\varphi = A_{\mathsf{G_s}\neg tt} \cup A_{\varphi'}$ where $A_{\varphi'}$ is constructed from $A_{\mathsf{F_s}tt}$, $A_{\mathsf{G_sF_s}tt}$, and $A_{\mathsf{F_sG_s}\varphi_3}$ using Lemma 5.

$-\mathsf{H_s}$ *Case* $\mathsf{G_sF_sH_s}\varphi_3$ A pointed word (u, i) satisfies $\mathsf{G_sF_sH_s}\varphi_3$ iff $i = |u| - 1$ or u is an infinite word satisfying $\mathsf{G}\varphi_3$. Thus, $\mathsf{G_sF_sH_s}\varphi_3 \equiv (\mathsf{G_s}\neg tt) \vee \varphi'$ where $\varphi' = (\mathsf{F_s}tt) \wedge (\mathsf{G_sF_s}tt) \wedge (\varphi_3 \wedge \mathsf{H_s}\varphi_3 \wedge \mathsf{G_s}\varphi_3)$. Hence, $A_\varphi = A_{\mathsf{G_s}\neg tt} \cup A_{\varphi'}$ where $A_{\varphi'}$ is constructed from $A_{\mathsf{F_s}tt}$, $A_{\mathsf{G_sF_s}tt}$, A_{φ_3}, $A_{\mathsf{H_s}\varphi_3}$, and $A_{\mathsf{G_s}\varphi_3}$ using Lemma 5.

$\circ\mathsf{P_s}$ *Case* $\mathsf{G_sP_s}\varphi_2$ A pointed word (u, i) satisfies $\mathsf{G_sP_s}\varphi_2$ iff $i = |u| - 1$ or (u, i) satisfies $\mathsf{P}\varphi_2$. Hence, $A_\varphi = A_{\mathsf{G_s}\neg tt} \cup A_{\varphi_2} \cup A_{\mathsf{P_s}\varphi_2}$.

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

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

$-\wedge$ *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 5.

$-\mathsf{F_s}$ *Case* $\mathsf{G_s}(\varphi_2 \vee \mathsf{F_s}\varphi_4)$ It holds that $\mathsf{G_s}(\varphi_2 \vee \mathsf{F_s}\varphi_4) \equiv (\mathsf{G_s}\varphi_2) \vee \mathsf{F_s}(\mathsf{F_s}\varphi_4 \wedge \mathsf{G_s}\varphi_2) \vee \mathsf{G_sF_s}\varphi_4$. Therefore, the set A_φ can be constructed as $A_{\mathsf{G_s}\varphi_2} \cup A_{\mathsf{F_s}(\mathsf{F_s}\varphi_4 \wedge \mathsf{G_s}\varphi_2)} \cup A_{\mathsf{G_sF_s}\varphi_4}$, where $A_{\mathsf{F_s}(\mathsf{F_s}\varphi_4 \wedge \mathsf{G_s}\varphi_2)}$ is obtained from $A_{\mathsf{F_s}\varphi_4}$ and $A_{\mathsf{G_s}\varphi_2}$ using Lemma 5.

$-\mathsf{H_s}$ *Case* $\mathsf{G_s}(\varphi_2 \vee \mathsf{H_s}\varphi_4)$ As $\mathsf{G_s}(\varphi_2 \vee \mathsf{H_s}\varphi_4) \equiv (\mathsf{G_s}\varphi_2) \vee \mathsf{F_s}(\mathsf{H_s}\varphi_4 \wedge \mathsf{G_s}\varphi_2) \vee \mathsf{G_sH_s}\varphi_4$. Hence, $A_\varphi = A_{\mathsf{G_s}\varphi_2} \cup A_{\mathsf{F_s}(\mathsf{H_s}\varphi_4 \wedge \mathsf{G_s}\varphi_2)} \cup A_{(\mathsf{G_sH_s}\varphi_4)}$ where $A_{\mathsf{F_s}(\mathsf{H_s}\varphi_4 \wedge \mathsf{G_s}\varphi_2)}$ can be obtained from $A_{\mathsf{H_s}\varphi_4}$ and $A_{\mathsf{G_s}\varphi_2}$ using Lemma 5.

$-\mathsf{G_s}, \mathsf{P_s}$ There are only the following six 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 $\mathsf{G_s}(\bigvee_{\varphi' \in G} \mathsf{G_s}\varphi') \equiv (\mathsf{G_s}\neg tt) \vee \bigvee_{\varphi' \in G}(\mathsf{XG_s}\varphi')$. Therefore, the set A_φ can be constructed as $A_{\mathsf{G_s}\neg tt} \cup \bigcup_{\varphi' \in G} A_{\mathsf{XG_s}\varphi'}$ where each $A_{\mathsf{XG_s}\varphi'}$ is obtained from $A_{\mathsf{G_s}\varphi'}$ using Lemma 5.

(ii) *Case* $\mathsf{G_s}(p_2 \vee \bigvee_{\varphi' \in G} \mathsf{G_s}\varphi')$ As $\mathsf{G_s}(p_2 \vee \bigvee_{\varphi' \in G} \mathsf{G_s}\varphi') \equiv (\mathsf{G_s}p_2) \vee \bigvee_{\varphi' \in G}(\mathsf{X}(p_2 \,\mathsf{U}\, (\mathsf{G_s}\varphi')))$, the set A_φ can be constructed as $A_{\mathsf{G_s}p_2} \cup$

$\bigcup_{\varphi' \in G} A_{\mathsf{X}(p_2 \,\mathsf{U}\, (\mathsf{G_s}\varphi'))}$ where each $A_{\mathsf{X}(p_2 \,\mathsf{U}\, (\mathsf{G_s}\varphi'))}$ is obtained from $A_{\mathsf{G_s}\varphi'}$ using Lemma 5.

(iii) *Case* $\mathsf{G_s}(\bigvee_{\varphi'' \in P} \mathsf{P_s}\varphi'')$ It holds that $\mathsf{G_s}(\bigvee_{\varphi'' \in P} \mathsf{P_s}\varphi'') \equiv (\mathsf{G_s}\neg tt) \vee \bigvee_{\varphi'' \in P}(\mathsf{X}\mathsf{P_s}\varphi'')$. Therefore, the set A_φ can be constructed as $A_{\mathsf{G_s}\neg tt} \cup \bigcup_{\varphi'' \in P} A_{\mathsf{X}\mathsf{P_s}\varphi''}$ where each $A_{\mathsf{X}\mathsf{P_s}\varphi''}$ is obtained from $A_{\mathsf{P_s}\varphi''}$ using Lemma 5.

(iv) *Case* $\mathsf{G_s}(p_2 \vee \bigvee_{\varphi'' \in P} \mathsf{P_s}\varphi'')$ As $\mathsf{G_s}(p_2 \vee \bigvee_{\varphi'' \in P} \mathsf{P_s}\varphi'') \equiv (\mathsf{G_s} p_2) \vee \bigvee_{\varphi'' \in P}(\mathsf{X}(p_2 \,\mathsf{U}\, (\mathsf{P_s}\varphi'')))$, the set A_φ can be constructed as $A_{\mathsf{G_s} p_2} \cup \bigcup_{\varphi'' \in P} A_{\mathsf{X}(p_2 \,\mathsf{U}\, (\mathsf{P_s}\varphi''))}$ where each $A_{\mathsf{X}(p_2 \,\mathsf{U}\, (\mathsf{P_s}\varphi''))}$ is obtained from $A_{\mathsf{P_s}\varphi''}$ using Lemma 5.

(v) *Case* $\mathsf{G_s}(\bigvee_{\varphi' \in G} \mathsf{G_s}\varphi' \vee \bigvee_{\varphi'' \in P} \mathsf{P_s}\varphi'')$ As $\mathsf{G_s}(\bigvee_{\varphi' \in G} \mathsf{G_s}\varphi' \vee \bigvee_{\varphi'' \in P} \mathsf{P_s}\varphi'') \equiv (\mathsf{G_s}\neg tt) \vee \bigvee_{\varphi' \in G}(\mathsf{X}\mathsf{G_s}\varphi') \vee \bigvee_{\varphi'' \in P}(\mathsf{X}\mathsf{P_s}\varphi'')$, the set A_φ can be constructed as $A_{\mathsf{G_s}\neg tt} \cup \bigcup_{\varphi' \in G} A_{\mathsf{X}\mathsf{G_s}\varphi'} \cup \bigcup_{\varphi'' \in P} A_{\mathsf{X}\mathsf{P_s}\varphi''}$ where each $A_{\mathsf{X}\mathsf{G_s}\varphi'}$ is obtained from $A_{\mathsf{G_s}\varphi'}$ and each $A_{\mathsf{X}\mathsf{P_s}\varphi''}$ is obtained from $A_{\mathsf{P_s}\varphi''}$ using Lemma 5.

(vi) *Case* $\mathsf{G_s}(p_2 \vee \bigvee_{\varphi' \in G} \mathsf{G_s}\varphi' \vee \bigvee_{\varphi'' \in P} \mathsf{P_s}\varphi'')$ As $\mathsf{G_s}(p_2 \vee \bigvee_{\varphi' \in G} \mathsf{G_s}\varphi' \vee \bigvee_{\varphi'' \in P} \mathsf{P_s}\varphi'') \equiv (\mathsf{G_s} p_2) \vee \bigvee_{\varphi' \in G}(\mathsf{X}(p_2 \,\mathsf{U}\, (\mathsf{G_s}\varphi'))) \vee \bigvee_{\varphi'' \in P}(\mathsf{X}(p_2 \,\mathsf{U}\, (\mathsf{P_s}\varphi'')))$, the set A_φ can be constructed as $A_{\mathsf{G_s} p_2} \cup \bigcup_{\varphi' \in G} A_{\mathsf{X}(p_2 \,\mathsf{U}\, (\mathsf{G_s}\varphi'))} \cup \bigcup_{\varphi'' \in P} A_{\mathsf{X}(p_2 \,\mathsf{U}\, (\mathsf{P_s}\varphi''))}$ where each $A_{\mathsf{X}(p_2 \,\mathsf{U}\, (\mathsf{G_s}\varphi'))}$ is obtained from $A_{\mathsf{G_s}\varphi'}$ and each $A_{\mathsf{X}(p_2 \,\mathsf{U}\, (\mathsf{P_s}\varphi''))}$ is obtained from $A_{\mathsf{P_s}\varphi''}$ using Lemma 5.

∘$\mathsf{G_s}$ *Case* $\mathsf{G_s}\mathsf{G_s}\varphi_2$ As $\mathsf{G_s}(\mathsf{G_s}\varphi_2) \equiv (\mathsf{G_s}\neg tt) \vee (\mathsf{X}\mathsf{G_s}\varphi_2)$, the set A_φ can be constructed as $A_{\mathsf{G_s}\neg tt} \cup A_{\mathsf{X}\mathsf{G_s}\varphi_2}$ where $A_{\mathsf{X}\mathsf{G_s}\varphi_2}$ is obtained from $A_{\mathsf{G_s}\varphi_2}$ using Lemma 5.

∘$\mathsf{H_s}$ *Case* $\mathsf{G_s}\mathsf{H_s}\varphi_2$ A pointed word (u, i) satisfies $\mathsf{G_s}(\mathsf{H_s}\varphi_2)$ iff $i = |u| - 1$ or $(u, |u| - 1)$ satisfies $\mathsf{H_s}\varphi_2$ or u is infinite and all its positions satisfy φ_2. Hence, $A_\varphi = A_{\mathsf{G_s}\neg tt} \cup A_{\mathsf{F_s}((\mathsf{G_s}\neg tt) \wedge (\mathsf{H_s}\varphi_2))} \cup A_{(\mathsf{H_s}\varphi_2) \wedge \varphi_2 \wedge (\mathsf{G_s}\varphi_2)}$ where $A_{\mathsf{F_s}((\mathsf{G_s}\neg tt) \wedge (\mathsf{H_s}\varphi_2))}$ and $A_{(\mathsf{H_s}\varphi_2) \wedge \varphi_2 \wedge (\mathsf{G_s}\varphi_2)}$ are obtained from $A_{\mathsf{G_s}\neg tt}$, $A_{\mathsf{H_s}\varphi_2}$, A_{φ_2}, and $A_{\mathsf{G_s}\varphi_2}$ using Lemma 5.

•$\mathsf{H_s}$ *Case* $\mathsf{H_s}\varphi_1$ This case is divided into the following subcases according to the structure of φ_1.

∘p *Case* $\mathsf{H_s} p$ As $\mathsf{H_s} p$ is globally equivalent to $tt \wedge \mathsf{H_s} p$, we set $A_\varphi = \{\mathrm{P}\alpha(tt \wedge \mathsf{H_s} p, tt \wedge \mathsf{G_s} tt, \emptyset)\}$.

∘$\wedge$ *Case* $\mathsf{H_s}(\varphi_2 \wedge \varphi_3)$ As $\mathsf{H_s}(\varphi_2 \wedge \varphi_3) \equiv (\mathsf{H_s}\varphi_2) \wedge (\mathsf{H_s}\varphi_3)$, the set A_φ can be constructed from $A_{\mathsf{H_s}\varphi_2}$ and $A_{\mathsf{H_s}\varphi_3}$ using Lemma 5.

∘$\mathsf{F_s}$ *Case* $\mathsf{H_s}\mathsf{F_s}\varphi_2$ A pointed word (u, i) satisfies $\mathsf{H_s}\mathsf{F_s}\varphi_2$ iff $i = 0$ or (u, i) satisfies $\mathsf{F}\varphi_2$. Note that $\mathsf{H_s}\neg tt$ is satisfied by (u, i) only if $i = 0$. Therefore, $A_\varphi = A_{\mathsf{H_s}\neg tt} \cup A_{\varphi_2} \cup A_{\mathsf{F_s}\varphi_2}$.

∘$\mathsf{P_s}$ *Case* $\mathsf{H_s}\mathsf{P_s}\varphi_2$ A pointed word (u, i) satisfies $\mathsf{H_s}\mathsf{P_s}\varphi_2$ iff $i = 0$. Therefore, $A_\varphi = A_{\mathsf{H_s}\neg tt}$.

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

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

–$\wedge$ *Case* $\mathsf{H_s}(\varphi_2 \vee (\varphi_4 \wedge \varphi_5))$ As $\mathsf{H_s}(\varphi_2 \vee (\varphi_4 \wedge \varphi_5)) \equiv \mathsf{H_s}(\varphi_2 \vee \varphi_4) \wedge \mathsf{H_s}(\varphi_2 \vee \varphi_5)$, the set A_φ can be constructed from $A_{\mathsf{H_s}(\varphi_2 \vee \varphi_4)}$ and $A_{\mathsf{H_s}(\varphi_2 \vee \varphi_5)}$ using Lemma 5.

–$\mathsf{P_S}$ *Case* $\mathsf{H_S}(\varphi_2 \vee \mathsf{P_S}\varphi_4)$ It holds that $\mathsf{H_S}(\varphi_2 \vee \mathsf{P_S}\varphi_4) \equiv (\mathsf{H_S}\varphi_2) \vee \mathsf{P_S}(\mathsf{P_S}\varphi_4 \wedge \mathsf{H_S}\varphi_2)$. Therefore, the set A_φ can be constructed as $A_{\mathsf{H_S}\varphi_2} \cup A_{\mathsf{P_S}(\mathsf{P_S}\varphi_4 \wedge \mathsf{H_S}\varphi_2)}$, where $A_{\mathsf{P_S}(\mathsf{P_S}\varphi_4 \wedge \mathsf{H_S}\varphi_2)}$ is obtained from $A_{\mathsf{P_S}\varphi_4}$ and $A_{\mathsf{H_S}\varphi_2}$ using Lemma 5.

–$\mathsf{G_S}$ *Case* $\mathsf{H_S}(\varphi_2 \vee \mathsf{G_S}\varphi_4)$ As $\mathsf{H_S}(\varphi_2 \vee \mathsf{G_S}\varphi_4) \equiv (\mathsf{H_S}\varphi_2) \vee \mathsf{P_S}(\mathsf{G_S}\varphi_4 \wedge \mathsf{H_S}\varphi_2)$, A_φ is constructed as $A_{\mathsf{H_S}\varphi_2} \cup A_{\mathsf{P_S}(\mathsf{G_S}\varphi_4 \wedge \mathsf{H_S}\varphi_2)}$ where $A_{\mathsf{P_S}(\mathsf{G_S}\varphi_4 \wedge \mathsf{H_S}\varphi_2)}$ is obtained from $A_{\mathsf{G_S}\varphi_4}$ and $A_{\mathsf{H_S}\varphi_2)}$ using Lemma 5.

–$\mathsf{F_S}, \mathsf{H_S}$ There are only the following six subcases (the others fit to some of the previous cases).

(i) *Case* $\mathsf{H_S}(\bigvee_{\varphi' \in F} \mathsf{F_S}\varphi')$ It holds that $\mathsf{H_S}(\bigvee_{\varphi' \in F} \mathsf{F_S}\varphi') \equiv (\mathsf{H_S}\neg tt) \vee \bigvee_{\varphi' \in F}(\mathsf{Y}\mathsf{F_S}\varphi')$. Therefore, the set A_φ can be constructed as $A_{\mathsf{H_S}\neg tt} \cup \bigcup_{\varphi' \in F} A_{\mathsf{Y}\mathsf{F_S}\varphi'}$ where each $A_{\mathsf{Y}\mathsf{F_S}\varphi'}$ is obtained from $A_{\mathsf{F_S}\varphi'}$ using Lemma 5.

(ii) *Case* $\mathsf{H_S}(p_2 \vee \bigvee_{\varphi' \in F} \mathsf{F_S}\varphi')$ As $\mathsf{H_S}(p_2 \vee \bigvee_{\varphi' \in F} \mathsf{F_S}\varphi') \equiv (\mathsf{H_S}p_2) \vee \bigvee_{\varphi' \in F}(\mathsf{Y}(p_2\, \mathsf{S}\, (\mathsf{F_S}\varphi')))$, the set A_φ can be constructed as $A_{\mathsf{H_S}p_2} \cup \bigcup_{\varphi' \in F} A_{\mathsf{Y}(p_2\, \mathsf{S}\, (\mathsf{F_S}\varphi'))}$ where each $A_{\mathsf{Y}(p_2\, \mathsf{S}\, (\mathsf{F_S}\varphi'))}$ is obtained from $A_{\mathsf{F_S}\varphi'}$ using Lemma 5.

(iii) *Case* $\mathsf{H_S}(\bigvee_{\varphi'' \in H} \mathsf{H_S}\varphi'')$ It holds that $\mathsf{H_S}(\bigvee_{\varphi'' \in H} \mathsf{H_S}\varphi'') \equiv (\mathsf{H_S}\neg tt) \vee \bigvee_{\varphi'' \in H}(\mathsf{Y}\mathsf{H_S}\varphi'')$. Therefore, the set A_φ can be constructed as $A_{\mathsf{H_S}\neg tt} \cup \bigcup_{\varphi'' \in H} A_{\mathsf{Y}\mathsf{H_S}\varphi''}$ where each $A_{\mathsf{Y}\mathsf{H_S}\varphi''}$ is obtained from $A_{\mathsf{H_S}\varphi''}$ using Lemma 5.

(iv) *Case* $\mathsf{H_S}(p_2 \vee \bigvee_{\varphi'' \in H} \mathsf{H_S}\varphi'')$ As $\mathsf{H_S}(p_2 \vee \bigvee_{\varphi'' \in H} \mathsf{H_S}\varphi'') \equiv (\mathsf{H_S}p_2) \vee \bigvee_{\varphi'' \in H}(\mathsf{Y}(p_2\, \mathsf{S}\, (\mathsf{H_S}\varphi'')))$, the set A_φ can be constructed as $A_{\mathsf{H_S}p_2} \cup \bigcup_{\varphi'' \in H} A_{\mathsf{Y}(p_2\, \mathsf{S}\, (\mathsf{H_S}\varphi''))}$ where each $A_{\mathsf{Y}(p_2\, \mathsf{S}\, (\mathsf{H_S}\varphi''))}$ is obtained from $A_{\mathsf{H_S}\varphi''}$ using Lemma 5.

(v) *Case* $\mathsf{H_S}(\bigvee_{\varphi' \in F} \mathsf{F_S}\varphi' \vee \bigvee_{\varphi'' \in H} \mathsf{H_S}\varphi'')$ As $\mathsf{H_S}(\bigvee_{\varphi' \in F} \mathsf{F_S}\varphi' \vee \bigvee_{\varphi'' \in H} \mathsf{H_S}\varphi'') \equiv (\mathsf{H_S}\neg tt) \vee \bigvee_{\varphi' \in F}(\mathsf{Y}\mathsf{F_S}\varphi') \vee \bigvee_{\varphi'' \in H}(\mathsf{Y}\mathsf{H_S}\varphi'')$, the set A_φ can be constructed as $A_{\mathsf{H_S}\neg tt} \cup \bigcup_{\varphi' \in F} A_{\mathsf{Y}\mathsf{F_S}\varphi'} \cup \bigcup_{\varphi'' \in H} A_{\mathsf{Y}\mathsf{H_S}\varphi''}$ where each $A_{\mathsf{Y}\mathsf{F_S}\varphi'}$ is obtained from $A_{\mathsf{F_S}\varphi'}$ and each $A_{\mathsf{Y}\mathsf{H_S}\varphi''}$ is obtained from $A_{\mathsf{H_S}\varphi''}$ using Lemma 5.

(vi) *Case* $\mathsf{H_S}(p_2 \vee \bigvee_{\varphi' \in F} \mathsf{F_S}\varphi' \vee \bigvee_{\varphi'' \in H} \mathsf{H_S}\varphi'')$ As $\mathsf{H_S}(p_2 \vee \bigvee_{\varphi' \in F} \mathsf{F_S}\varphi' \vee \bigvee_{\varphi'' \in H} \mathsf{H_S}\varphi'') \equiv (\mathsf{H_S}p_2) \vee \bigvee_{\varphi' \in F}(\mathsf{Y}(p_2\, \mathsf{S}\, (\mathsf{F_S}\varphi'))) \vee \bigvee_{\varphi'' \in H}(\mathsf{Y}(p_2\, \mathsf{S}\, (\mathsf{H_S}\varphi'')))$, the set A_φ can be constructed as $A_{\mathsf{H_S}p_2} \cup \bigcup_{\varphi' \in F} A_{\mathsf{Y}(p_2\, \mathsf{S}\, (\mathsf{F_S}\varphi'))} \cup \bigcup_{\varphi'' \in H} A_{\mathsf{Y}(p_2\, \mathsf{S}\, (\mathsf{H_S}\varphi''))}$ where each $A_{\mathsf{Y}(p_2\, \mathsf{S}\, (\mathsf{F_S}\varphi'))}$ is obtained from $A_{\mathsf{F_S}\varphi'}$ and each $A_{\mathsf{Y}(p_2\, \mathsf{S}\, (\mathsf{H_S}\varphi''))}$ is obtained from $A_{\mathsf{H_S}\varphi''}$ using Lemma 5.

∘$\mathsf{G_S}$ *Case* $\mathsf{H_S}\mathsf{G_S}\varphi_2$ A pointed word (u, i) satisfies $\mathsf{H_S}(\mathsf{G_S}\varphi_2)$ iff $i = 0$ or $(u, 0)$ satisfies $\mathsf{G_S}\varphi_2$. Hence, $A_\varphi = A_{\mathsf{H_S}\neg tt} \cup A_{\mathsf{P_S}((\mathsf{H_S}\neg tt) \wedge (\mathsf{G_S}\varphi_2))}$ where $A_{\mathsf{P_S}((\mathsf{H_S}\neg tt) \wedge (\mathsf{G_S}\varphi_2))}$ is obtained from $A_{\mathsf{H_S}\neg tt}$ and $A_{\mathsf{G_S}\varphi_2}$ using Lemma 5.

∘$\mathsf{H_S}$ *Case* $\mathsf{H_S}\mathsf{H_S}\varphi_2$ As $\mathsf{H_S}(\mathsf{H_S}\varphi_2) \equiv (\mathsf{H_S}\neg tt) \vee (\mathsf{Y}\mathsf{H_S}\varphi_2)$, the set A_φ can be constructed as $A_{\mathsf{H_S}\neg tt} \cup A_{\mathsf{Y}\mathsf{H_S}\varphi_2}$ where $A_{\mathsf{Y}\mathsf{H_S}\varphi_2}$ is obtained from $A_{\mathsf{H_S}\varphi_2}$ using Lemma 5. □

Remark 4 In other words, we have just shown that LTL($\mathsf{F_S}, \mathsf{G_S}, \mathsf{P_S}, \mathsf{H_S}$) is a semantic subset (with respect to the global equivalence) of every formalism that is

– able to express p, $\mathsf{G_S}p$, $\mathsf{H_S}p$, and $\mathsf{G_S}\mathsf{F_S}p$, where $p \in$ LTL(), and
– closed under disjunction, conjunction, and applications of X, Y, $p\,\mathsf{U}\,-$, and $p\,\mathsf{S}\,-$, where $p \in$ LTL().

References

1. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: application to model-checking. In: Proceedings of CONCUR'97, Lecture Notes in Computer Science, vol. 1243, pp. 135–150 (1997)
2. Bouajjani, A., Habermehl, P.: Constrained properties, semilinear systems, and petri nets. In: Proceedings of CONCUR'96, Lecture Notes in Computer Science, vol. 1119, pp. 481–497. Springer, Heidelberg (1996)
3. Bozzelli, L.: Model checking for process rewrite systems and a class of action-based regular properties. In: Proceedings of VMCAI'05, Lecture Notes in Computer Science, vol. 3385, pp. 282–297. Springer, Heidelberg (2005)
4. Bozzelli, L., Křetínský, M., Řehák, V., Strejček, J.: On decidability of LTL model checking for process rewrite systems. In: FSTTCS 2006, Lecture Notes in Computer Science, vol. 4337, pp. 248–259. Springer, Heidelberg (2006)
5. Esparza, J.: On the decidability of model checking for several mu-calculi and petri nets. In: CAAP, Lecture Notes in Computer Science, vol. 787, pp. 115–129. Springer, Heidelberg (1994)
6. Etessami, K., Vardi, M.Y., Wilke, T.: First-order logic with two variables and unary temporal logic. Inf. Comput. **179**(2), 279–295 (2002)
7. Gabbay, D.: The declarative past and imperative future: executable temporal logic for interactive systems. In: Temporal Logic in Specification, Lecture Notes in Computer Science, vol. 398, pp. 409–448 (1987)
8. Habermehl, P.: On the complexity of the linear-time μ-calculus for petri nets. In: Proceedings of ICATPN'97, Lecture Notes in Computer Science, vol. 1248, pp. 102–116. Springer, Heidelberg (1997)
9. Křetínský, M., Řehák, V., Strejček, J.: Extended process rewrite systems: expressiveness and reachability. In: Proceedings of CONCUR'04, Lecture Notes in Computer Science, vol. 3170, pp. 355–370. Springer, Heidelberg (2004)
10. Křetínský, M., Řehák, V., Strejček, J.: On extensions of process rewrite systems: rewrite systems with weak finite-state unit. In: Proceedings of INFINITY'03, Electr. Notes Theor. Comput. Sci., vol. 98, pp. 75–88. Elsevier, Amsterdam (2004)
11. Křetínský, M., Řehák, V., Strejček, J.: Reachability of Hennessy–Milner properties for weakly extended PRS. In: Proceedings of FSTTCS 2005, Lecture Notes in Computer Science, vol. 3821, pp. 213–224. Springer, Heidelberg (2005)
12. Křetínský, M., Řehák, V., Strejček, J.: On Decidability of LTL+Past Model Checking for Process Rewrite Systems. In: Proceedings of INFINITY'07, Electr. Notes Theor. Comput. Sci. Elsevier, Amsterdam (2007) (to appear)
13. Lipton, R.: The reachability problem is exponential-space hard. Tech. Rep. 62, Department of Computer Science, Yale University (1976)
14. Maidl, M.: The common fragment of CTL and LTL. In: Proceedings of 41th Annual Symposium on Foundations of Computer Science, pp. 643–652 (2000)
15. Mayr, E.W.: An algorithm for the general Petri net reachability problem. SIAM J. Comput. **13**(3), 441–460 (1984)
16. Mayr, R.: Decidability and complexity of model checking problems for infinite-state systems. PhD thesis, Technische Universität München (1998)
17. Mayr, R.: Process rewrite systems. Inf. Comput. **156**(1), 264–286 (2000)
18. Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)
19. Minsky, M.L.: Computation: Finite and Infinite Machines. Prentice-Hall, Englewood Cliffs (1967)
20. Pnueli, A.: The temporal logic of programs. In: Proceedings of 18th IEEE Symposium on the Foundations of Computer Science, pp. 46–57 (1977)
21. Řehák, V.: On Extensions of Process Rewrite Systems. PhD thesis, Faculty of Informatics, Masaryk University, Brno (2007)
22. Strejček, J.: Linear temporal logic: Expressiveness and model checking. PhD thesis, Faculty of Informatics, Masaryk University, Brno (2004)