Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science 239 (2009) 105–117
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

On Decidability of LTL+Past Model Checking for Process Rewrite Systems

Mojmír Křetínský[1,4] Vojtěch Řehák[2,5] Jan Strejček[3,6]

Faculty of Informatics, Masaryk University
Botanická 68a, 60200 Brno, Czech Republic

Abstract

The paper [4] shows that the model checking problem for (weakly extended) Process Rewrite Systems and properties given by LTL formulae with temporal operators *strict eventually* and *strict always* is decidable. The same paper contains an open question whether the problem remains decidable even if we extend the set of properties by allowing also past counterparts of the mentioned operators. The current paper gives a positive answer to this question.

Keywords: rewrite systems, infinite-state systems, model checking, decidability, linear temporal logic

1 Introduction

To specify (the classes of) infinite-state systems we employ term rewrite systems called *Process Rewrite Systems* (PRS) [16]. PRS subsume a variety of the formalisms studied in the context of formal verification, e.g. *Petri nets* (PN), *pushdown processes* (PDA), and process algebras like PA. Moreover, they are suitable to model current software systems with restricted forms of dynamic creation and synchronization of concurrent processes or recursive procedures or both. The relevance of PRS (and their subclasses) for modelling and analysing programs is shown, for example, in [7]; for automatic verification we refer to surveys [5,19].

[1] Email: kretinsky@fi.muni.cz

[2] Email: rehak@fi.muni.cz

[3] Email: strejcek@fi.muni.cz

[4] Supported by Ministry of Education of the Czech Republic, project No. MSM0021622419, and by the Czech Science Foundation, grant No. 201/09/1389.

[5] Supported by the research centre "Institute for Theoretical Computer Science (ITI)", project No. 1M0545, and by the Czech Science Foundation, grant No. 201/08/P459.

[6] Supported by the Academy of Sciences of the Czech Republic, grant No. 1ET408050503, and by the Czech Science Foundation, grant No. 201/08/P375.

doi:10.1016/j.entcs.2009.05.033

Another merit of PRS is that the *reachability problem* is decidable for PRS [16]. In [13], 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 shown that the reachability problem remains decidable for wPRS [12].

One of the mainstreams in an automatic verification of programs is model checking. Here we focus on *Linear Temporal Logic* (LTL). Recall that LTL model checking is decidable for both PDA (EXPTIME-complete [1]) and PN (at least as hard as the reachability problem for PN [6]). Conversely, LTL model checking is undecidable for all the classes subsuming PA [2,15]. So far, there are few positive results for these classes. Model checking of infinite runs is decidable for the PA class and the fragment *simple* $PLTL_{\Box}$, see [2], and also for the PRS class and a fragment of LTL expressing exactly fairness properties [3]. Recently, the model checking problem has been shown decidable for (w)PRS and properties given by an LTL fragment LTL($\mathsf{F_s}, \mathsf{G_s}$), i.e. that with operators *strict eventually* and *strict always* only, see [4].

Our contribution: As a main result we extend a proof technique used in [4] with past modalities and show that the model checking problem stays decidable even for wPRS and LTL($\mathsf{F_s}, \mathsf{P_s}$), i.e. an LTL fragment with modalities *strict eventually* and *eventually in the strict past* (and where *strict always* and *always in the strict past* can be used as derived modalities). We note that a role of past operators in program verification is advocated e.g. in [14,9]. Let us mention that the expressive power of the fragment LTL($\mathsf{F_s}, \mathsf{P_s}$) semantically coincides with formulae of First-Order Monadic Logic of Order containing at most 2 variables and no successor predicate ($\mathsf{FO}^2[<]$), see [8] for effective translations. Thus we also positively solve the model checking problem for the wPRS class and $\mathsf{FO}^2[<]$.

2 Preliminaries

2.1 Weakly Extended PRS (wPRS)

Let $Const = \{X, \ldots\}$ be a set of *process constants*. A set $\mathcal{T}$ 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. $\varepsilon.t = t.\varepsilon = t \| \varepsilon = t$.

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*. An *wPRS* (*weakly extended process rewrite system*) Δ is a tuple (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 $t_1, t_2 \in \mathcal{T}$, $t_1 \neq \varepsilon$, $a \in Act$, and $p, q \in M$ satisfy $p \leq q$,
- the pair $(p_0, t_0) \in M \times \mathcal{T}$ 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 $\longrightarrow$ is the least relation satisfying the following inference rules:

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

To shorten our notation we write pt in lieu of (p, t). A state pt is called *terminal* if there is no state $p't'$ and no action a such that $pt \xrightarrow{a} p't'$. Here, we always consider only such systems where the initial state is not terminal. A (finite or infinite) sequence

$$\sigma = p_0t_0 \xrightarrow{a_0} p_1t_1 \xrightarrow{a_1} \dots \xrightarrow{a_n} p_{n+1}t_{n+1} \left(\xrightarrow{a_{n+1}} \dots \right)$$

is called a *run of* Δ *over the word* $u = a_0a_1 \dots a_n(a_{n+1} \dots)$ if it starts in the initial state and, provided it is finite, ends in a terminal state. Further, $L(\Delta)$ denotes the set of words u such that there is a run of Δ over u.

If $M(\Delta)$ is a singleton, then wPRS Δ is called a *process rewrite system* (*PRS*) [16]. PRS, wPRS, and their respective subclasses are discussed in more detail in [18].

2.2 Linear Temporal Logic (LTL) and the Studied Problems

The syntax of *Linear Temporal Logic* (LTL) [17] is defined as follows

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

where X and U are 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 $u = a_0a_1a_2 \dots \in Act^* \cup Act^\omega$, $|u|$ 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* $0 \le i < |u|$ in this word.

The semantics of LTL formulae is defined inductively as follows:

$$\begin{array}{lll}
(u,i) \models tt & & \\
(u,i) \models a & \text{iff} & u = a_0a_1a_2\dots \text{ and } a_i = a \\
(u,i) \models \neg\varphi & \text{iff} & (u,i) \not\models \varphi \\
(u,i) \models \varphi_1 \wedge \varphi_2 & \text{iff} & (u,i) \models \varphi_1 \text{ and } (u,i) \models \varphi_2 \\
(u,i) \models \mathsf{X}\varphi & \text{iff} & i+1 < |u| \text{ and } (u,i+1) \models \varphi \\
(u,i) \models \varphi_1 \,\mathsf{U}\, \varphi_2 & \text{iff} & \exists k.\ \big(i \le k < |u| \ \wedge \ (u,k) \models \varphi_2 \ \wedge \\
& & \qquad \wedge \ \forall j.\ (i \le j < k \ \Rightarrow \ (u,j) \models \varphi_1)\big) \\
(u,i) \models \mathsf{Y}\varphi & \text{iff} & 0 < i \text{ and } (u,i-1) \models \varphi \\
(u,i) \models \varphi_1 \,\mathsf{S}\, \varphi_2 & \text{iff} & \exists k.\ \big(0 \le k \le i \ \wedge \ (u,k) \models \varphi_2 \ \wedge \\
& & \qquad \wedge \ \forall j.\ (k < j \le i \ \Rightarrow \ (u,j) \models \varphi_1)\big)
\end{array}$$

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.

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, then LTL$(O_1, \ldots, O_n)$ denotes an LTL fragment containing all formulae with modalities $O_1, \ldots, O_n$ only. Such a fragment is called *basic* if it contains future operators only or with each future operator it contains its past counterpart. For example, the fragment LTL(F, S) is not basic. Figure 1 shows an expressiveness hierarchy of all studied basic LTL fragments. Indeed, every basic LTL fragment using standard[7] 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. We also mind the result of [9] stating that each LTL formula can be converted to the one which employs future operators only, i.e. LTL$(\mathsf{U}, \mathsf{X}) \equiv_i$ LTL$(\mathsf{U}, \mathsf{S}, \mathsf{X}, \mathsf{Y})$. However note that LTL$(\mathsf{F_s}, \mathsf{P_s}, \mathsf{G_s}, \mathsf{H_s}) \equiv$ LTL$(\mathsf{F_s}, \mathsf{P_s})$ is strictly more expressive than LTL$(\mathsf{F_s}, \mathsf{G_s})$ as can be exemplified by a formula $\mathsf{F_s}(b \wedge \mathsf{H_s}a) \equiv_i a \wedge \mathsf{X}(a\,\mathsf{U}\,b)$. We refer to [20] for greater detail.

This paper deals with the following two verification problems. Let $\mathcal{F}$ be an LTL fragment. The *model checking problem* for $\mathcal{F}$ and wPRS is to decide, for any given formula $\varphi \in \mathcal{F}$ and any given wPRS system Δ, whether $L(\Delta) \models \varphi$ holds. Further, given any formula $\varphi \in \mathcal{F}$, any wPRS system Δ, and any nonterminal state pt of Δ, the *pointed model checking problem* for $\mathcal{F}$ and wPRS is to decide whether $L(pt, \Delta) \models \varphi$; here $L(pt, \Delta)$ denotes the set of all pointed words (u, i) such that Δ has a (finite or infinite) 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 $pt = p_it_i$.

[7] 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.

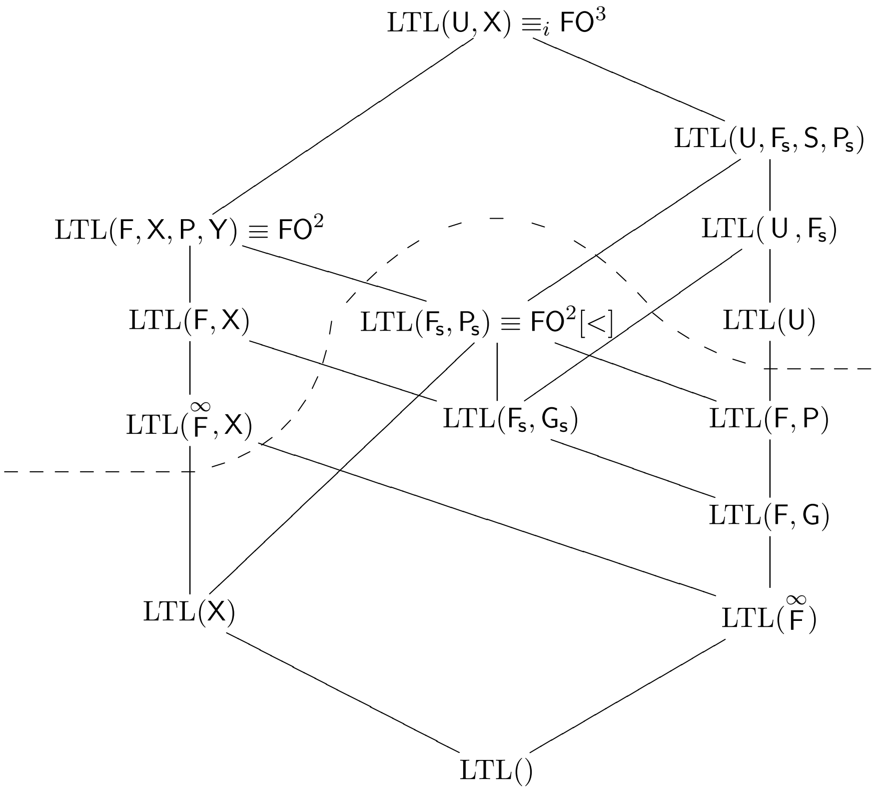

Fig. 1. The hierarchy of basic LTL fragments with respect to the initial equivalence. The dashed line shows the decidability boundary of the model checking problem for wPRS.

3 Main Result

In [4], we have shown that the model checking problem is decidable for LTL($\mathsf{F_s}, \mathsf{G_s}$). Before we prove that the problem remains decidable even for a more expressive fragment LTL($\mathsf{F_s}, \mathsf{P_s}$), we recall the basic structure of the proof for LTL($\mathsf{F_s}, \mathsf{G_s}$).

First, the proof shows that every LTL($\mathsf{F_s}, \mathsf{G_s}$) formula can be effectively translated into an equivalent disjunction of so-called *α-formulae*, which are defined below. Note that LTL() denotes the fragment of formulae without any modality, i.e. boolean combinations of actions. In what follows, we use $\varphi_1 \mathsf{U}_+ \varphi_2$ to abbreviate $\varphi_1 \wedge \mathsf{X}(\varphi_1 \mathsf{U} \varphi_2)$. Let $\delta = \theta_1 O_1 \theta_2 O_2 \ldots \theta_n O_n \theta_{n+1}$, where $n > 0$, each $\theta_i \in \text{LTL}()$, 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 $\mathcal{B} \subseteq \text{LTL}()$ be a finite set. An α-formula is defined as

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

Hence, a word u satisfies $\alpha(\delta, \mathcal{B})$ iff u can be written as a concatenation $v_1.v_2 \ldots v_{n+1}$ of words, where

- each word v_i consists only of actions satisfying θ_i and

  · $|v_i| \geq 0$ if $i = n+1$ or O_i is 'U',
  · $|v_i| > 0$ if O_i is 'U_+',
  · $|v_i| = 1$ if O_i is '$\wedge\,\mathsf{X}$' or '$\wedge\,\mathsf{G_s}$',
- and v_{n+1} satisfies $\mathsf{G_s F_s}\psi$ for every $\psi \in \mathcal{B}$.

Second, decidability of the model checking problem for LTL($\mathsf{F_s}, \mathsf{G_s}$) is then a direct consequence of the following theorem.

Theorem 3.1 **([4])** *The problem whether any given wPRS systems has a run satisfying any given α-formula is decidable.*

To prove decidability for LTL($\mathsf{F_s}, \mathsf{P_s}$), we show that every LTL($\mathsf{F_s}, \mathsf{P_s}$) formula can be effectively translated into a disjunction of 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 3.2 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(), 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

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

Note that the definition of a Pα-formula does not contain any past counterpart of $\wedge_{\psi\in\mathcal{B}} \mathsf{G_s F_s}\psi$ as every history is finite — the semantics of LTL is given in terms of words with a fixed beginning.

Therefore, a pointed word $(u, k) \models \mathrm{P}\alpha(\eta, \delta, \mathcal{B})$ if and only if (u, k) satisfies $\alpha(\delta, \mathcal{B})$ and $a_0 \ldots a_{k-1} a_k$ can be written as a concatenation $v_{m+1}.v_m \ldots v_2.v_1$, 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 proof of the following lemma is intuitively clear but it is quite a technical exercise, see [18] for some hints.

Lemma 3.3 *Let φ be a* Pα*-formula and $p \in LTL()$. Formulae $\mathsf{X}\varphi$, $\mathsf{Y}\varphi$, $p\,\mathsf{U}\,\varphi$, $p\,\mathsf{S}\,\varphi$, $\mathsf{F_s}\varphi$, $\mathsf{P_s}(\varphi)$, as well as, a conjunction of* Pα*-formulae can be effectively converted into a globally equivalent disjunction of* Pα*-formulae.*

Theorem 3.4 *Every LTL($\mathsf{F_s}, \mathsf{P_s}$) formula φ can be translated into a globally equivalent disjunction of* Pα*-formulae.*

Proof. As $\mathsf{F_s}, \mathsf{G_s}$ and $\mathsf{P_s}, \mathsf{H_s}$ are dual modalities, we can assume that every LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$) formula contains negations only in front of actions. Given an LTL($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$) 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($\mathsf{F_s}, \mathsf{G_s}, \mathsf{P_s}, \mathsf{H_s}$) 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 $A_\varphi = \{\mathrm{P}\alpha(tt \wedge \mathsf{H_s} tt, p \wedge \mathsf{G_s} tt, \emptyset)\}$.
- •$\vee$ **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 3.3, A_φ can be constructed from the sets A_{φ_1} and A_{φ_2}.
- •$\mathsf{F_s}$ **Case** $\mathsf{F_s}\varphi_1$**:** Due to Lemma 3.3, the set A_φ can be constructed from the set A_{φ_1}.
- •$\mathsf{P_s}$ **Case** $\mathsf{P_s}\varphi_1$**:** Due to Lemma 3.3, the set A_φ can be constructed from the set A_{φ_1}.
- •$\mathsf{G_s}$ **Case** $\mathsf{G_s}\varphi_1$ 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 $A_\varphi = \{\mathrm{P}\alpha(tt \wedge \mathsf{H_s} tt, tt \wedge \mathsf{G_s}p, \emptyset)\}$.
  - ∘$\wedge$ **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 3.3. 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 $A_\varphi = \{\mathrm{P}\alpha(tt \wedge \mathsf{H_s} tt, tt \wedge \mathsf{G_s} tt, \{p\})\}$.
    - −$\vee$ **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 3.3.
      - ∗$\mathsf{P_s}$ **Case** $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \mathsf{P_s}\varphi_5)$**:** As $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \mathsf{P_s}\varphi_5) \equiv (\mathsf{G_s}\mathsf{F_s}\varphi_3) \wedge (\mathsf{G_s}\mathsf{F_s}\mathsf{P_s}\varphi_5)$, the set A_φ can be constructed from $A_{\mathsf{G_s}\mathsf{F_s}\varphi_3}$ and $A_{\mathsf{G_s}\mathsf{F_s}\mathsf{P_s}\varphi_5}$ using Lemma 3.3.
      - ∗$\mathsf{G_s}$ **Case** $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \mathsf{G_s}\varphi_5)$**:** As $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \mathsf{G_s}\varphi_5) \equiv (\mathsf{G_s}\mathsf{F_s}\varphi_3) \wedge (\mathsf{G_s}\mathsf{F_s}\mathsf{G_s}\varphi_5)$, the set A_φ can be constructed from $A_{\mathsf{G_s}\mathsf{F_s}\varphi_3}$ and $A_{\mathsf{G_s}\mathsf{F_s}\mathsf{G_s}\varphi_5}$ using Lemma 3.3.
      - ∗$\mathsf{H_s}$ **Case** $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \mathsf{H_s}\varphi_5)$**:** As $\mathsf{G_s}\mathsf{F_s}(\varphi_3 \wedge \mathsf{H_s}\varphi_5) \equiv (\mathsf{G_s}\mathsf{F_s}\varphi_3) \wedge (\mathsf{G_s}\mathsf{F_s}\mathsf{H_s}\varphi_5)$, the set A_φ can be constructed from $A_{\mathsf{G_s}\mathsf{F_s}\varphi_3}$ and $A_{\mathsf{G_s}\mathsf{F_s}\mathsf{H_s}\varphi_5}$ using Lemma 3.3.
    - −$\mathsf{F_s}$ **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 $A_\varphi = A_{\mathsf{G_s}\mathsf{F_s}\varphi_3}$.
    - −$\mathsf{P_s}$ **Case** $\mathsf{G_s}\mathsf{F_s}\mathsf{P_s}\varphi_3$**:** A pointed word (u, i) satisfies $\mathsf{G_s}\mathsf{F_s}\mathsf{P_s}\varphi_3$ 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 $(\mathsf{F_s} tt) \wedge (\mathsf{G_s}\mathsf{F_s} tt)$ iff u is infinite. Thus, $\mathsf{G_s}\mathsf{F_s}\mathsf{P_s}\varphi_3 \equiv (\mathsf{G_s}\neg tt) \vee \varphi'$ where $\varphi' = (\mathsf{F_s} tt) \wedge (\mathsf{G_s}\mathsf{F_s} tt) \wedge (\varphi_3 \vee \mathsf{P_s}\varphi_3 \vee \mathsf{F_s}\varphi_3)$. 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 3.3.

$-\mathsf{G_s}$ **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 3.3.

$-\mathsf{H_s}$ **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 3.3.

$\circ\mathsf{P_s}$ **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 $p_2 \vee p_3 \in$ 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 3.3.

$-\mathsf{F_s}$ **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 3.3.

$-\mathsf{H_s}$ **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 3.3.

$-\mathsf{G_s}, \mathsf{P_s}$ 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 3.3.

(*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 3.3.

(*iii*) **Case** $\mathsf{G_s}(\bigvee_{\varphi'' \in P} \mathsf{P_s}\varphi'')$**:** It holds that $\mathsf{G_s}(\bigvee_{\varphi'' \in P} \mathsf{P_s}\varphi'') \equiv (\mathsf{G_s}\neg tt) \vee \bigvee_{\varphi'' \in P}(\mathsf{XP_s}\varphi'')$. Therefore, the set A_φ can be constructed as $A_{\mathsf{G_s}\neg tt} \cup \bigcup_{\varphi'' \in P} A_{\mathsf{XP_s}\varphi''}$ where each $A_{\mathsf{XP_s}\varphi''}$ is obtained from $A_{\mathsf{P_s}\varphi''}$ using Lemma 3.3.

(*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 3.3.

(*v*) **Case** $\mathsf{G_s}(\bigvee_{\varphi' \in G} \mathsf{G_s}\varphi' \vee \bigvee_{\varphi'' \in P} \mathsf{P_s}\varphi'')$**:** As $\mathsf{G_s}(\bigvee_{\varphi' \in G} \mathsf{G_s}\varphi' \vee \bigvee_{\varphi'' \in P} \mathsf{P_s}\varphi'') \equiv (\mathsf{G_s}\neg tt) \vee \bigvee_{\varphi' \in G}(\mathsf{XG_s}\varphi') \vee \bigvee_{\varphi'' \in P}(\mathsf{XP_s}\varphi'')$, the set A_φ can be constructed as $A_{\mathsf{G_s}\neg tt} \cup \bigcup_{\varphi' \in G} A_{\mathsf{XG_s}\varphi'} \cup \bigcup_{\varphi'' \in P} A_{\mathsf{XP_s}\varphi''}$ where each $A_{\mathsf{XG_s}\varphi'}$ is obtained from $A_{\mathsf{G_s}\varphi'}$ and each $A_{\mathsf{XP_s}\varphi''}$ is obtained from $A_{\mathsf{P_s}\varphi''}$ using Lemma 3.3.

(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 3.3.

$\circ\mathsf{G_s}$ **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 $A_{\mathsf{G_s}\neg tt} \cup A_{\mathsf{X}\mathsf{G_s}\varphi_2}$ where $A_{\mathsf{X}\mathsf{G_s}\varphi_2}$ is obtained from $A_{\mathsf{G_s}\varphi_2}$ using Lemma 3.3.

$\circ\mathsf{H_s}$ **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 3.3.

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

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

$\circ\wedge$ **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 3.3.

$\circ\mathsf{F_s}$ **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, $A_\varphi = A_{\mathsf{H_s}\neg tt} \cup A_{\varphi_2} \cup A_{\mathsf{F_s}\varphi_2}$.

$\circ\mathsf{P_s}$ **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, $A_\varphi = A_{\mathsf{H_s}\neg tt}$.

$\circ\vee$ **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 3.3.

$-\mathsf{P_s}$ **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 3.3.

$-\mathsf{G_s}$ **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 3.3.

$-\mathsf{F_s}, \mathsf{H_s}$ 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 3.3.

(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 3.3.

(*iii*) **Case** $\mathsf{H_s}(\bigvee_{\varphi'' \in H} \mathsf{H_s}\varphi'')$**:** It holds that $\mathsf{H_s}(\bigvee_{\varphi'' \in H} \mathsf{H_s}\varphi'') \equiv (\mathsf{H_s}\neg tt) \vee \bigvee_{\varphi'' \in H}(\mathsf{YH_s}\varphi'')$. Therefore, the set A_φ can be constructed as $A_{\mathsf{H_s}\neg tt} \cup \bigcup_{\varphi'' \in H} A_{\mathsf{YH_s}\varphi''}$ where each $A_{\mathsf{YH_s}\varphi''}$ is obtained from $A_{\mathsf{H_s}\varphi''}$ using Lemma 3.3.

(*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 3.3.

(*v*) **Case** $\mathsf{H_s}(\bigvee_{\varphi' \in F} \mathsf{F_s}\varphi' \vee \bigvee_{\varphi'' \in H} \mathsf{H_s}\varphi'')$**:** As $\mathsf{H_s}(\bigvee_{\varphi' \in F} \mathsf{F_s}\varphi' \vee \bigvee_{\varphi'' \in H} \mathsf{H_s}\varphi'') \equiv (\mathsf{H_s}\neg tt) \vee \bigvee_{\varphi' \in F}(\mathsf{YF_s}\varphi') \vee \bigvee_{\varphi'' \in H}(\mathsf{YH_s}\varphi'')$, the set A_φ can be constructed as $A_{\mathsf{H_s}\neg tt} \cup \bigcup_{\varphi' \in F} A_{\mathsf{YF_s}\varphi'} \cup \bigcup_{\varphi'' \in H} A_{\mathsf{YH_s}\varphi''}$ where each $A_{\mathsf{YF_s}\varphi'}$ is obtained from $A_{\mathsf{F_s}\varphi'}$ and each $A_{\mathsf{YH_s}\varphi''}$ is obtained from $A_{\mathsf{H_s}\varphi''}$ using Lemma 3.3.

(*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 3.3.

$\circ\mathsf{G_s}$ **Case** $\mathsf{H_sG_s}\varphi_2$**:** 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 3.3.

$\circ\mathsf{H_s}$ **Case** $\mathsf{H_sH_s}\varphi_2$**:** As $\mathsf{H_s}(\mathsf{H_s}\varphi_2) \equiv (\mathsf{H_s}\neg tt) \vee (\mathsf{YH_s}\varphi_2)$, the set A_φ can be constructed as $A_{\mathsf{H_s}\neg tt} \cup A_{\mathsf{YH_s}\varphi_2}$ where $A_{\mathsf{YH_s}\varphi_2}$ is obtained from $A_{\mathsf{H_s}\varphi_2}$ using Lemma 3.3.

□

Remark 3.5 In other words, we have just shown that LTL($\mathsf{F_s}, \mathsf{P_s}$) is a semantic subset (with respect to global equivalence) of every formalism that is (i) able to express p, $\mathsf{G_s}p$, $\mathsf{H_s}p$, and $\mathsf{G_sF_s}p$, where $p \in$ LTL(); and (ii) is closed under disjunction, conjunction, and applications of $\mathsf{X}\_$, $\mathsf{Y}\_$, $p\,\mathsf{U}\,\_$, and $p\,\mathsf{S}\,\_$, where $p \in$ LTL().

Now, using Theorem 3.1, we can easily solve the problem dual to the model checking problem, i.e. given any wPRS system and any Pα-formula, to decide whether the system has a run satisfying the formula.

Theorem 3.6 *The problem whether any 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, $(u,0) \models \mathrm{P}\alpha(\eta,\delta,\mathcal{B})$ 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, Theorem 3.1 finishes the proof. □

As LTL($\mathsf{F_s}, \mathsf{P_s}$) is closed under negation, Theorem 3.4 and Theorem 3.6 give us the following.

Corollary 3.7 *The model checking problem for wPRS and LTL($\mathsf{F_s}, \mathsf{P_s}$) is decidable.*

Moreover, we can show that the pointed model checking problem is decidable for wPRS and LTL($\mathsf{F_s}, \mathsf{P_s}$) as well. Again, we solve the dual problem.

Theorem 3.8 *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 any given $\mathrm{P}\alpha$-formula is decidable.*

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

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

and $X_a, Y_a \notin \mathit{Const}(\Delta)$ are fresh process constants for each $a \in \mathit{Act}(\Delta)$.

It is easy to see that (u, i) is in $L(pt, \Delta)$ iff $u = a_0 a_1 \ldots a_{i-1} a_i.f.a_i.a_{i+1} \ldots$ is in $L(\Delta')$. Hence, for any given $\mathrm{P}\alpha$-formula $\varphi = \mathrm{P}\alpha(\eta, \delta, \mathcal{B})$ we construct a $\mathrm{P}\alpha$-formula $\varphi' = \mathrm{P}\alpha(\eta, tt \wedge \mathsf{X}f \wedge \mathsf{X}\delta, \mathcal{B})$. We get that

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

and due to Lemma 3.3 and Theorem 3.6 the proof is done. □

As LTL($\mathsf{F_s}, \mathsf{P_s}$) is closed under negation and Theorem 3.4 works with global equivalence, Theorem 3.8 give us the following.

Corollary 3.9 *The pointed model checking problem is decidable for wPRS and LTL($\mathsf{F_s}, \mathsf{P_s}$).*

4 Conclusion

We have examined the model checking problem for basic LTL fragments with both future and past modalities and the PRS class, i.e. the class of infinite state system generated by Process Rewrite Systems (PRS), possibly enriched with a weak finite control unit (weakly extended PRS – wPRS). We have proved that the problem is decidable for wPRS and LTL($\mathsf{F_s}, \mathsf{P_s}$), i.e. the fragment with modalities *strict eventually*, *eventually in the strict past*, and derived modalities *strict always* and *always in the strict past.*[8] However, both these problems are at least as hard as the reachability problem for PN [6] (EXPSPACE-hard without any elementary upper bound known).

Note that the expressive power of the fragment LTL($\mathsf{F_s}, \mathsf{P_s}$) semantically coincides with formulae of First-Order Monadic Logic of Order containing at most 2 variables

[8] In fact, we have shown that the problem is decidable even for a more expressive fragment containing negations of disjunctions of so-called $\mathrm{P}\alpha$-formulae (see Definition 3.2).

and no successor predicate ($\mathsf{FO}^2[<]$), and that First-Order Monadic Logic of Order containing at most 2 variables (FO^2) coincides with an LTL($\mathsf{F}, \mathsf{X}, \mathsf{P}, \mathsf{Y}$) fragment [8]. Further, let us recall our undecidability results for model checking of PA systems (a subclass of PRS) and fragments LTL($\overset{\infty}{\mathsf{F}}, \mathsf{X}$) and LTL(U), respectively (the former with modalities *infinitely often* and *next* only, the latter with *until* as the only modality), see [4].

Thus, we have located the borderline between decidability and undecidability of the problem for wPRS and the LTL fragments, as well as for wPRS and First-Order Monadic Logic of Order: it is decidable for $\mathsf{FO}^2[<]$ and undecidable for FO^2. For the sake of completeness, we note that the First-Order Monadic Logic of Order containing at most 3 variables (FO^3) coincides with the set of all LTL formulae as well as with the full First-Order Monadic Logic of Order [11,10]. Finally, we note that the decidability results are new for the PRS class too and they are illustrated by the decidability border in Figure 1.

References

[1] Bouajjani, A., J. Esparza and O. Maler, *Reachability Analysis of Pushdown Automata: Application to Model-Checking*, in: *Proc. of CONCUR'97*, LNCS **1243**, 1997, pp. 135–150.

[2] Bouajjani, A. and P. Habermehl, *Constrained Properties, Semilinear Systems, and Petri Nets*, in: *Proc. of CONCUR'96*, LNCS **1119** (1996), pp. 481–497.

[3] Bozzelli, L., *Model checking for process rewrite systems and a class of action-based regular properties*, in: *Proc. of VMCAI'05*, LNCS **3385** (2005), pp. 282–297.

[4] Bozzelli, L., M. Křetínský, V. Řehák and J. Strejček, *On decidability of LTL model checking for process rewrite systems*, in: *FSTTCS 2006*, LNCS **4337** (2006), pp. 248–259.

[5] Burkart, O., D. Caucal, F. Moller and B. Steffen, *Verification on infinite structures*, in: *Handbook of Process Algebra* (2001), pp. 545–623.

[6] Esparza, J., *On the Decidability of Model Checking for Several mu-calculi and Petri Nets*, in: *CAAP*, LNCS **787** (1994), pp. 115–129.

[7] Esparza, J., *Grammars as Processes*, in: *Formal and Natural Computing*, LNCS **2300** (2002), pp. 277–297.

[8] Etessami, K., M. Y. Vardi and T. Wilke, *First-order logic with two variables and unary temporal logic*, Information and Computation **179** (2002), pp. 279–295.

[9] Gabbay, D., *The Declarative Past and Imperative Future: Executable Temporal Logic for Interactive Systems*, in: *Temporal Logic in Specification*, LNCS **398**, 1987, pp. 409–448.

[10] Gabbay, D., A. Pnueli, S. Shelah and J. Stavi, *On the Temporal Analysis of Fairness*, in: *Conference Record of the 7th Annual ACM Symposium on Principles of Programming Languages (POPL'80)* (1980), pp. 163–173.

[11] Kamp, J. A. W., "Tense Logic and the Theory of Linear Order," Ph.D. thesis, University of California, Los Angeles (1968).

[12] Křetínský, M., V. Řehák and J. Strejček, *Extended Process Rewrite Systems: Expressiveness and Reachability*, in: *Proceedings of CONCUR'04*, LNCS **3170** (2004), pp. 355–370.

[13] Křetínský, M., V. Řehák and J. Strejček, *On Extensions of Process Rewrite Systems: Rewrite Systems with Weak Finite-State Unit*, in: *Proceedings of INFINITY'03*, Electr. Notes Theor. Comput. Sci. **98** (2004), pp. 75–88.

[14] Lichtenstein, O., A. Pnueli and L. D. Zuck, *The Glory of The Past*, in: *Logic of Programs*, LNCS **193**, 1985, pp. 196–218.

[15] Mayr, R., "Decidability and Complexity of Model Checking Problems for Infinite-State Systems," Ph.D. thesis, Technische Universität München (1998).

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

[17] Pnueli, A., *The Temporal Logic of Programs*, in: *Proc. 18th IEEE Symposium on the Foundations of Computer Science*, 1977, pp. 46–57.

[18] Řehák, V., "On Extensions of Process Rewrite Systems," Ph.D. thesis, Faculty of Informatics, Masaryk University, Brno (2007).

[19] Srba, J., "Roadmap of Infinite Results," Current Trends In Theoretical Computer Science, The Challenge of the New Century **Vol 2: Formal Models and Semantics**, World Scientific Publishing Co., 2004, 337–350 pp.

[20] Strejček, J., "Linear Temporal Logic: Expressiveness and Model Checking," Ph.D. thesis, Faculty of Informatics, Masaryk University, Brno (2004).