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

Mojmír Křetínský¹, Vojtěch Řehák²,³, Jan Strejček⁴,⁵

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 \textit{strict eventually} and \textit{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.

\textbf{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 \textit{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].

Another merit of PRS is that the \textit{reachability problem} is decidable for PRS [16]. In [13], we have presented \textit{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-

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.

This paper is electronically published in

\textit{Electronic Notes in Theoretical Computer Science}

URL: \url{www.elsevier.nl/locate/entcs}
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 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 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 problem for PN [6]).

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(\(F_3, G_6\)), 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(\(F_3, G_6\)) semantically coincides with formulae of First-Order Monadic Logic of Order containing at most 2 variables and no successor predicate (FO\(^2[<]\)), see [8] for effective translations. Thus we also positively solve the model checking problem for the wPRS class and FO\(^2[<]\).

2 Preliminaries

2.1 Weakly Extended PRS (wPRS)

Let \(\text{Const} = \{X, \ldots\}\) be a set of process constants. A set \(\mathcal{T}\) of process terms \(t\) is defined by the abstract syntax \(t ::= \varepsilon \mid X \mid t.t \mid t\parallel t\), where \(\varepsilon\) is the empty term, \(X \in \text{Const}\), 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 \(\varepsilon\), i.e. \(\varepsilon.t = t.\varepsilon = t\parallel \varepsilon = \varepsilon\).

Let \(M = \{a, p, q, \ldots\}\) be a set of control states, \(\leq\) be a partial ordering on this set, and \(\text{Act} = \{a, b, c, \ldots\}\) be a set of actions. An wPRS (weakly extended process rewrite system) \(\Delta\) is a tuple \((R, p_0, t_0)\), where

- \(R\) is a finite set of rewrite rules of the form \((p, t_1) \xrightarrow{a} (q, t_2)\), where \(t_1, t_2 \in \mathcal{T}\), \(t_1 \neq \varepsilon\), \(a \in \text{Act}\), 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 \(\text{Act}(\Delta)\), \(\text{Const}(\Delta)\), 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 \(\Delta\).

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 \(\text{Const}(\Delta)\). The transition relation \(\xrightarrow{}\) is
the least relation satisfying the following inference rules:

\[
\frac{(p,t_1) \xrightarrow{a} (q,t_2)}{(p,t_1) \xrightarrow{a} (q,t_2)} \quad \frac{(p,t_1) \xrightarrow{a} (q,t_2)}{(p,t_1) \xrightarrow{a} (q,t_2)} \quad \frac{(p,t_1) \xrightarrow{a} (q,t_2)}{(p,t_1) \xrightarrow{a} (q,t_2)}
\]

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

\[
σ = p_0a_0 \xrightarrow{a_0} p_1a_1 \xrightarrow{a_n} \ldots \xrightarrow{a_{n+1}} p_{n+1}t_{n+1}(a_{n+1}, \ldots)
\]

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

If \(M(Δ)\) 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:

\[
ϕ := tt \mid a \mid \negϕ \mid ϕ \land ϕ \mid Xϕ \mid ϕ U ϕ \mid Yϕ \mid ϕ S ϕ,
\]

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 \(u = a_0a_1a_2 \ldots \in Act^+ \cup Act^\omega\), \(|u|\) 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 \(0 \leq i < |u|\) in this word.

The semantics of LTL formulae is defined inductively as follows:

\[
\begin{align*}
(u,i) \models tt & \quad \text{iff} \quad u = a_0a_1a_2 \ldots \text{ and } a_i = a \\
(u,i) \models \negϕ & \quad \text{iff} \quad (u,i) \not\models ϕ \\
(u,i) \models ϕ_1 \land ϕ_2 & \quad \text{iff} \quad (u,i) \models ϕ_1 \quad \text{and} \quad (u,i) \models ϕ_2 \\
(u,i) \models Xϕ & \quad \text{iff} \quad i + 1 < |u| \quad \text{and} \quad (u,i + 1) \models ϕ \\
(u,i) \models ϕ_1 U ϕ_2 & \quad \text{iff} \quad \exists k. \left( i \leq k < |u| \land (u,k) \models ϕ_2 \land \forall j. \left( i \leq j < k \Rightarrow (u,j) \models ϕ_1 \right) \right) \\
(u,i) \models Yϕ & \quad \text{iff} \quad 0 < i \quad \text{and} \quad (u,i - 1) \models ϕ \\
(u,i) \models ϕ_1 S ϕ_2 & \quad \text{iff} \quad \exists k. \left( 0 \leq k \leq i \land (u,k) \models ϕ_2 \land \forall j. \left( k < j \leq i \Rightarrow (u,j) \models ϕ_1 \right) \right)
\end{align*}
\]

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

Formulae \(ϕ,ψ\) are (initially) equivalent, written \(ϕ \equiv_i ψ\), if, for all words \(u\), it holds that \(u \models ϕ \iff u \models ψ\). Formulae \(ϕ,ψ\) are globally equivalent, written \(ϕ \equiv ψ\), if, for all
past modality meaning | future modality meaning |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pϕ eventually in the past</td>
<td>Fϕ eventually</td>
</tr>
<tr>
<td>Hϕ always in the past</td>
<td>Gϕ always</td>
</tr>
<tr>
<td>P_sϕ eventually in the strict past</td>
<td>F_sϕ strict eventually</td>
</tr>
<tr>
<td>H_sϕ always in the strict past</td>
<td>G_sϕ strict always</td>
</tr>
<tr>
<td>lϕ initially</td>
<td>F_pϕ infinitely often</td>
</tr>
</tbody>
</table>

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 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\((U,X) \equiv \) LTL\((U,S,X,Y)\). However note that LTL\((F_s, P_s, G_s, H_s)\) \(\equiv\) LTL\((F_s, P_s)\) is strictly more expressive than LTL\((F_s, G_s)\) as can be exemplified by a formula \( F_s(b \land H_s a) \equiv_i a \land X(a \lor b) \). 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 \( \Delta \), whether \( L(\Delta) \models \varphi \) holds. Further, given any formula \( \varphi \in \mathcal{F} \), any wPRS system \( \Delta \), and any nonterminal state \( pt \) of \( \Delta \), 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 \( \Delta \) has a (finite or infinite) run \( p_0 a_0 \rightarrow p_1 a_1 \rightarrow \ldots a_{n-1} \rightarrow p_n a_n \rightarrow \ldots \) satisfying \( u = a_0 a_1 a_2 \ldots \) and \( pt = p_n a_n \).

### 3 Main Result

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

First, the proof shows that every LTL\((F_s, G_s)\) formula can be effectively translated into an equivalent disjunction of so-called \( \alpha \)-formulae, which are defined below. Note that LTL\((\cdot)\) denotes the fragment of formulae without any modality, i.e. boolean combinations of actions. In what follows, we use \( \varphi_1 U_+ \varphi_2 \) to abbreviate \( \varphi_1 \land X(\varphi_1 U_+ \varphi_2) \). Let \( \delta = \theta_0 O_1 \theta_2 O_2 \ldots \theta_n O_n \theta_{n+1} \), where \( n > 0 \), each \( \theta_i \in \text{LTL}(\cdot) \), \( O_n \) is \( \land G_s \), and, for each \( i < n, O_i \) is either \( 'U' \) or \( 'U_+' \) or \( '\land X' \). Further, let \( B \subseteq \text{LTL}(\cdot) \) be a finite set. An \( \alpha \)-formula

---

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.
is defined as
\[
\alpha(\delta, B) = (\theta_1 O_1 (\theta_2 O_2 \ldots (\theta_n O_n \theta_{n+1}) \ldots)) \land \bigwedge_{\psi \in B} G \delta F \psi.
\]
Hence, a word \( u \) satisfies \( \alpha(\delta, B) \) iff \( u \) can be written as a concatenation \( v_1 \cdot v_2 \ldots v_{n+1} \) of words, where
- each word \( v_i \) consists only of actions satisfying \( \theta_i \), 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 ‘\& X’ or ‘\& G_s’.
- and \( v_{n+1} \) satisfies \( G \delta F \psi \) for every \( \psi \in B \).

Second, decidability of the model checking problem for \( \text{LTL}(F_s, G_s) \) 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 \( \alpha \)-formula is decidable.*

To prove decidability for \( \text{LTL}(F_s, P_s) \), we show that every \( \text{LTL}(F_s, P_s) \) formula can be effectively translated into a disjunction of \( P\alpha \)-formulae. Intuitively, a \( P\alpha \)-formula is a conjunction of an \( \alpha \)-formula and a past version of the \( \alpha \)-formula. A formal definition of a \( P\alpha \)-formula makes use of \( \phi_1 S \phi_2 \) to abbreviate \( \phi_1 \land Y(\phi_1 S \phi_2) \).
Definition 3.2 Let η = t₁P₁t₂P₂...tₘPₘtₘ₊₁, where m > 0, each tᵢ ∈ LTL(), and, for each j < m, Pᵢ is either ‘S’ or ‘S,’ or ‘∧ ’. Further, let α(δ, B) be an α-formula. Then a \( Pα \)-formula is defined as

\[
Pα(η, δ, B) = (t_1P_1(t_2P_2...t_mP_{m+1}...)) \land α(δ, B).
\]

Note that the definition of a \( Pα \)-formula does not contain any past counterpart of \( \land ψ \in B \) \( G_5Fsψ \) as every history is finite — the semantics of LTL is given in terms of words with a fixed beginning.

Therefore, a pointed word \((u,k) \models Pα(η, δ, B)\) if and only if \((u,k)\) satisfies \( α(δ, B)\) and \(a_0...a_{k-1}a_k\) can be written as a concatenation \(v_{m+1}v_m...v_2v_1\), where each word \(v_i\) consists only of actions satisfying \(t_i\) and

- \(|v_i| > 0\) if \(i = m + 1\) or \(P_i\) is ‘S’,
- \(|v_i| > 0\) if \(P_i\) is ‘S,’,
- \(|v_i| = 1\) if \(P_i\) is ‘∧ Y’ or ‘∧ H’.

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 \((Xφ, Yφ, p∪φ, pSφ, Fsφ, P_5(φ))\), 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(\( F_5, P_5 \)) formula \( φ \) can be translated into a globally equivalent disjunction of \( Pα \)-formulae.

Proof. As \( F_5, G_5 \) and \( P_5, H_5 \) are dual modalities, we can assume that every LTL(\( F_5, G_5, P_5, H_5 \)) formula contains negations only in front of actions. Given an LTL(\( F_5, G_5, P_5, H_5 \)) 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 \( φ \notin LTL()\), then we assume that for every LTL(\( F_5, G_5, P_5, H_5 \)) formula \( φ' \) shorter than \( φ \) we can construct the corresponding set \( A_φ' \). 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 \( p \land G_5tt \). Hence \( A_φ = \{ Pα(tt \land H_5tt, p \land G_5tt, θ) \} \).
- **v** Case \( φ_1 \lor φ_2 \): Due to induction hypothesis, we can assume that we have sets \( A_{φ_1} \) and \( A_{φ_2} \). Clearly, \( A_φ = A_{φ_1}∪A_{φ_2} \).
- **∧** Case \( φ_1 \land φ_2 \): Due to Lemma 3.3, \( A_φ \) can be constructed from the sets \( A_{φ_1} \) and \( A_{φ_2} \).
- **F_5** Case \( F_5φ_1 \): Due to Lemma 3.3, the set \( A_φ \) can be constructed from the set \( A_{φ_1} \).
- **P_5** Case \( P_5φ_1 \): Due to Lemma 3.3, the set \( A_φ \) can be constructed from the set \( A_{φ_1} \).
- **G_5** Case \( G_5φ_1 \) is divided into the following subcases according to the structure of \( φ_1 \):
  - **p** Case \( G_5p \): As \( G_5p \) is equivalent to \( tt \land G_5p \), we set \( A_φ = \{ Pα(tt \land H_5tt, tt \land G_5p, θ) \} \).
  - **∧** Case \( G_5(φ_2 \land φ_3) \): As \( G_5(φ_2 \land φ_3) = (G_5φ_2) \land (G_5φ_3) \), the set \( A_φ \) can be constructed from \( A_{G_5φ_2} \) and \( A_{G_5φ_3} \) using Lemma 3.3. Note that \( A_{G_5φ_2} \) and \( A_{G_5φ_3} \) can be constructed because \( G_5φ_2 \) and \( G_5φ_3 \) are shorter than \( G_5(φ_2 \land φ_3) \).
  - **F_5** Case \( G_5F_5φ_2 \): This case is again divided into the following subcases.
\(-p\) Case $G_F p$: As $p \in \text{LTL}()$, we directly set $A_p = \{\text{Pref}(\text{tt} \land H_\text{tt} \land G_\text{tt}, \{p\})\}$.

\(-\lor\) Case $G_F (\phi_3 \lor \phi_4)$: As $G_F (\phi_3 \lor \phi_4) \equiv (G_F \phi_3) \lor (G_F \phi_4)$, we set $A_p = A_{G_F \phi_3} \lor A_{G_F \phi_4}$.

\(-\land\) Case $G_F (\phi_3 \land \phi_4)$: This case is also divided into subcases depending on the formulae $\phi_3$ and $\phi_4$.

\(*p\) Case $G_F (p_3 \land p_4)$: As $p_3 \land p_4 \in \text{LTL}()$, this subcase has already been covered by Case $G_F p$.

\(*\lor\) Case $G_F (\phi_3 \land (\phi_5 \lor \phi_6))$: As $G_F \phi_3 \equiv (G_F \phi_3 \land \phi_3) \lor (G_F \phi_3 \land \phi_6)$, we set $A_p = A_{G_F \phi_3} \cup A_{G_F \phi_6}$.

\(*_2\lor\) Case $G_F (\phi_3 \land F_2 \phi_5)$: As $G_F \phi_3 \equiv (G_F \phi_3) \land (G_F \phi_3 \lor F_2 \phi_5)$, the set $A_p$ can be constructed from $A_{G_F \phi_3}$ and $A_{G_F \phi_5}$ using Lemma 3.3.

\(*p_3\) Case $G_F (\phi_3 \land P_3 \phi_3)$: As $G_F \phi_3 \equiv (G_F \phi_3) \land (G_F \phi_3 \land P_3 \phi_3)$, the set $A_p$ can be constructed from $A_{G_F \phi_3}$ and $A_{G_F P_3 \phi_3}$ using Lemma 3.3.

\(*_3\land\) Case $G_F (\phi_3 \land G_3 \phi_3)$: As $G_F \phi_3 \equiv (G_F \phi_3) \land (G_F \phi_3 \land G_3 \phi_3)$, the set $A_p$ can be constructed from $A_{G_F \phi_3}$ and $A_{G_F G_3 \phi_3}$ using Lemma 3.3.

\(*_3\lor\) Case $G_F (\phi_3 \land H_3 \phi_3)$: As $G_F \phi_3 \equiv (G_F \phi_3) \land (G_F \phi_3 \land H_3 \phi_3)$, the set $A_p$ can be constructed from $A_{G_F \phi_3}$ and $A_{G_F H_3 \phi_3}$ using Lemma 3.3.

\(-F_2\) Case $G_F F_2 \phi_3$: As $G_F F_2 \phi_3 \equiv G_F F_2 \phi_3$, we set $A_p = A_{G_F \phi_3}$.

\(-F_2\lor\) Case $G_F F_2 P_2 \phi_3$: A pointed word $(u, i)$ satisfies $G_F F_2 P_2 \phi_3$ iff $i = |u| - 1$ or $u$ is an infinite word satisfying $F_2 \phi_3$. Note that $G_2 \neg \text{tt}$ is satisfied only by finite words at their last position. Further, a word $u$ satisfies $(F_2 \text{tt}) \land (G_F \text{tt})$ iff $u$ is infinite. Thus, $G_F F_2 P_2 \phi_3 \equiv (G_2 \neg \text{tt}) \lor \phi'$ where $\phi' = (F_2 \text{tt}) \land (G_F \text{tt}) \land (\phi_3 \lor P_3 \phi_3 \lor F_3 \phi_3)$. Hence, $A_p = A_{G_2 \neg \text{tt}} \cup A_{\phi'}$, where $A_{\phi'}$ is constructed from $A_{F_2 \text{tt}}, A_{G_F \text{tt}}$, and $A_{\phi_3} \lor A_{P_3 \phi_3} \lor A_{F_3 \phi_3}$ using Lemma 3.3.

\(-G_3\) Case $G_F G_3 \phi_3$: A pointed word $(u, i)$ satisfies $G_F G_3 \phi_3$ iff $i = |u| - 1$ or $u$ is an infinite word satisfying $F_2 \phi_3$. Thus, $G_F G_3 \phi_3 \equiv (G_3 \neg \text{tt}) \land (G_F \phi_3 \land \phi_3)$. Hence, $A_p = A_{G_3 \neg \text{tt}} \land A_{\phi_3}$ where $A_{\phi_3}$ is constructed from $A_{F_3 \text{tt}}, A_{G_F \text{tt}}$, and $A_{G_3 \phi_3}$ using Lemma 3.3.

\(-H_3\) Case $G_F H_3 \phi_3$: A pointed word $(u, i)$ satisfies $G_F H_3 \phi_3$ iff $i = |u| - 1$ or $u$ is an infinite word satisfying $G_3 \phi_3$. Thus, $G_F H_3 \phi_3 \equiv (G_3 \neg \text{tt}) \lor (G_F \text{tt} \lor (G_3 \neg \text{tt} \lor \phi_3 \land G_3 \phi_3)$. Hence, $A_p = A_{G_3 \neg \text{tt}} \lor A_{\phi_3}$, where $A_{\phi_3}$ is constructed from $A_{F_3 \text{tt}}, A_{G_F \text{tt}}$, $A_{G_3 \phi_3}$, and $A_{G_3 \phi_3}$ using Lemma 3.3.

\(-P_3\) Case $G_F P_3 \phi_3$: A pointed word $(u, i)$ satisfies $G_F P_3 \phi_3$ iff $i = |u| - 1$ or $(u, i)$ satisfies $P_3 \phi_3$. Hence, $A_p = A_{G_3 \neg \text{tt}} \lor A_{\phi_3} \lor A_{P_3 \phi_3}$.

\(-\lor\) Case $G_F (\phi_2 \lor \phi_3)$: According to the structure of $\phi_2$ and $\phi_3$, there are the following subcases.

\(-p\) Case $G_F (p_2 \lor p_3)$: As $p_2 \lor p_3 \in \text{LTL}()$, this subcase has already been covered by Case $G_F p$.

\(-\land\) Case $G_F (\phi_2 \lor (\phi_4 \lor \phi_5))$: As $G_F (\phi_2 \lor (\phi_4 \lor \phi_5)) \equiv G_F (\phi_2 \lor \phi_4) \lor G_F (\phi_2 \lor \phi_5)$, the set $A_p$ can be constructed from $A_{G_F (\phi_2 \lor \phi_4)}$ and $A_{G_F (\phi_2 \lor \phi_5)}$ using Lemma 3.3.

\(-F_2\) Case $G_F (\phi_2 \lor F_2 \phi_4)$: It holds that $G_F (\phi_2 \lor F_2 \phi_4) \equiv (G_F \phi_2) \lor (F_2 \phi_4 \lor G_F \phi_2) \lor G_F \phi_2$. Therefore, the set $A_p$ can be constructed as $A_{G_F \phi_2} \lor A_{F_2 (F_2 \phi_4 \lor G_F \phi_2)} \lor A_{G_F \phi_2}$, where $A_{F_2 (F_2 \phi_4 \lor G_F \phi_2)}$ is obtained from $A_{F_2 \phi_4}$ and $A_{G_F \phi_2}$ using Lemma 3.3.

\(-H_3\) Case $G_F (\phi_2 \lor H_3 \phi_4)$: As $G_F (\phi_2 \lor H_3 \phi_4) \equiv (G_F \phi_2) \lor (H_3 \phi_4 \lor G_F \phi_2) \lor G_H \phi_4 \lor G_F \phi_4$. Hence, $A_p = A_{G_F \phi_2} \lor A_{F_2 (H_3 \phi_4 \lor G_F \phi_2)} \lor A_{G_H \phi_4}$ where $A_{F_2 (H_3 \phi_4 \lor G_F \phi_2)}$ can be obtained from $A_{H_3 \phi_4}$ and $A_{G_F \phi_2}$ using Lemma 3.3.
Theorem 4.4.

- $G_s, P_s$ There are only the following six subcases (the others fit to some of the previous cases).
  
  (i) **Case** $G_s (\forall \varphi' \in G_s \varphi')$: It holds that $G_s (\forall \varphi' \in G_s \varphi') \equiv (G_s \neg t t) \lor (G_s X G_s \varphi')$. Therefore, the set $A_\varphi$ can be constructed as $A_{G_s \neg t t} \cup \bigcup_{\varphi' \in G_s \varphi'} A_{XG_s \varphi'}$ where each $A_{XG_s \varphi'}$ is obtained from $A_{G_s \varphi'}$ using Lemma 3.3.

  (ii) **Case** $G_s (p_2 \lor \forall \varphi' \in G_s \varphi')$: As $G_s (p_2 \lor \forall \varphi' \in G_s \varphi') \equiv (G_s p_2) \lor (G_s \forall \varphi' \in (G_s \varphi'))$, the set $A_\varphi$ can be constructed as $A_{G_s p_2} \cup \bigcup_{\varphi' \in G_s \varphi'} A_{X(p_2 \lor (G_s \varphi'))}$ where each $A_{X(p_2 \lor (G_s \varphi'))}$ is obtained from $A_{G_s \varphi'}$ using Lemma 3.3.

  (iii) **Case** $G_s (\forall \varphi' \in P_s \varphi'')$: It holds that $G_s (\forall \varphi' \in P_s \varphi'') \equiv (G_s \neg t t) \lor (G_s \forall \varphi' \in (P_s \varphi''))$. Therefore, the set $A_\varphi$ can be constructed as $A_{G_s \neg t t} \cup \bigcup_{\varphi' \in P_s \varphi''} A_{XP_s \varphi''}$ where each $A_{XP_s \varphi''}$ is obtained from $A_{P_s \varphi''}$ using Lemma 3.3.

  (iv) **Case** $G_s (p_2 \lor \forall \varphi' \in P_s \varphi'')$: As $G_s (p_2 \lor \forall \varphi' \in P_s \varphi'') \equiv (G_s p_2) \lor (G_s \forall \varphi' \in (P_s \varphi''))$, the set $A_\varphi$ can be constructed as $A_{G_s p_2} \cup \bigcup_{\varphi' \in P_s \varphi''} A_{XP_s \varphi''}$ where each $A_{XP_s \varphi''}$ is obtained from $A_{P_s \varphi''}$ using Lemma 3.3.

  (v) **Case** $G_s (\forall \varphi' \in G_s \varphi' \lor \forall \varphi' \in P_s \varphi'')$: As $G_s (\forall \varphi' \in G_s \varphi' \lor \forall \varphi' \in P_s \varphi'') \equiv (G_s \neg t t) \lor (G_s \forall \varphi' \in (G_s \varphi') \lor (G_s \varphi''))$ \lor (G_s \forall \varphi' \in (P_s \varphi''))$, the set $A_\varphi$ can be constructed as $A_{G_s \neg t t} \cup \bigcup_{\varphi' \in G_s \varphi'} A_{X \varphi'} \lor \bigcup_{\varphi' \in P_s \varphi''} A_{XP_s \varphi''}$ where each $A_{XP_s \varphi''}$ is obtained from $A_{P_s \varphi''}$ using Lemma 3.3.

  (vi) **Case** $G_s (p_2 \lor \forall \varphi' \in G_s \varphi' \lor \forall \varphi' \in P_s \varphi'')$: As $G_s (p_2 \lor \forall \varphi' \in G_s \varphi' \lor \forall \varphi' \in P_s \varphi'') \equiv (G_s p_2) \lor (G_s \forall \varphi' \in (G_s \varphi') \lor (G_s \varphi''))$ \lor (G_s \forall \varphi' \in (P_s \varphi''))$, the set $A_\varphi$ can be constructed as $A_{G_s p_2} \lor \bigcup_{\varphi' \in G_s \varphi'} A_{X \varphi'} \lor \bigcup_{\varphi' \in P_s \varphi''} A_{XP_s \varphi''}$ where each $A_{XP_s \varphi''}$ is obtained from $A_{P_s \varphi''}$ using Lemma 3.3.

- $G_s$ Case $G_s \varphi_2$: As $G_s (G_s \varphi_2) \equiv (G_s \neg t t) \lor (G_s \varphi_2)$, the set $A_\varphi$ can be constructed as $A_{G_s \neg t t} \lor A_{G_s \varphi_2}$ where $A_{G_s \varphi_2}$ is obtained from $A_{G_s \varphi_2}$ using Lemma 3.3.

- $H_s$ Case $H_s \varphi_2$: A pointed word $(u, i)$ satisfies $H_s (H_s \varphi_2)$ iff $i = |u| - 1$ or $(u, |u| - 1)$ satisfies $H_s \varphi_2$ or $u$ is infinite and all its positions satisfy $\varphi_2$. Hence, $A_\varphi = A_{H_s \neg t t} \lor A_{H_s \varphi_2}$, where $A_{H_s \varphi_2}$ is obtained from $A_{H_s \neg t t} \lor A_{H_s \varphi_2}$ using Lemma 3.3.

- $F_s$ Case $F_s \varphi_2$: A pointed word $(u, i)$ satisfies $F_s (F_s \varphi_2)$ iff $i = 0$ or $(u, i)$ satisfies $F_s \varphi_2$. Therefore, $A_\varphi = A_{F_s \neg t t} \lor A_{F_s \varphi_2}$.

- $P_s$ Case $P_s \varphi_2$: A pointed word $(u, i)$ satisfies $P_s (P_s \varphi_2)$ iff $i = 0$. Therefore, $A_\varphi = A_{P_s \varphi_2}$.

- $V$ Case $V_s (\forall \varphi' \lor \forall \varphi'')$: According to the structure of $\varphi_2$ and $\varphi_3$, there are the following subcases.

  - $p$ Case $H_s (p_2 \lor p_3)$: As $p_2 \lor p_3 \in LTL()$, this subcase has already been covered by Case $H_s p$.

  - $\land$ Case $H_s (\varphi_2 \lor (\varphi_4 \lor \varphi_5))$: As $H_s (\varphi_2 \lor (\varphi_4 \lor \varphi_5)) \equiv H_s (\varphi_2 \lor \varphi_4) \lor H_s (\varphi_2 \lor \varphi_5)$. 8
the set $A_0$ can be constructed from $A_{H_0(\varphi_2 \lor \varphi_4)}$ and $A_{H_0(\varphi_2 \land \varphi_3)}$ using Lemma 3.3.

- $P_5$ Case $H_0(\varphi_2 \lor P_5 \varphi_4)$: It holds that $H_0(\varphi_2 \lor P_5 \varphi_4) \equiv (H_0 \varphi_2) \lor P_5 (P_5 \varphi_4 \land H_0 \varphi_3)$. Therefore, the set $A_0$ can be constructed as $A_{H_0(\varphi_2 \lor P_5 \varphi_4)} \cup A_{P_5 (P_5 \varphi_4 \land H_0 \varphi_3)}$, where $A_{P_5 (P_5 \varphi_4 \land H_0 \varphi_3)}$ is obtained from $A_0$ and $A_{H_0(\varphi_2 \lor P_5 \varphi_4)}$ using Lemma 3.3.

- $G_3$ Case $H_0(\varphi_2 \lor G_3 \varphi_4)$: As $H_0(\varphi_2 \lor G_3 \varphi_4) \equiv (H_0 \varphi_2) \lor G_3 (G_3 \varphi_4 \land H_0 \varphi_2)$, $A_0$ is constructed as $A_{H_0(\varphi_2 \lor G_3 \varphi_4)} \cup A_{P_5 (G_3 \varphi_4 \land H_0 \varphi_2)}$ where $A_{P_5 (G_3 \varphi_4 \land H_0 \varphi_2)}$ is obtained from $A_{G_3 \varphi_4}$ and $A_{H_0(\varphi_2 \lor P_5 \varphi_4)}$ using Lemma 3.3.

- $F_3, H_0$ There are only the following six subcases (the others fit to some of the previous cases).
  
  (i) Case $H_0(\forall \varphi' \in F_3 \varphi')$: It holds that $H_0(\forall \varphi' \in F_3 \varphi') \equiv (H_0 - t) \lor \forall \varphi' \in F \left( Y_F \varphi' \right)$. Therefore, the set $A_0$ can be constructed as $A_{H_0 - t} \cup A_{\forall \varphi' \in F} A_{Y_F \varphi'}$, where each $A_{Y_F \varphi'}$ is obtained from $A_{F \varphi'}$ using Lemma 3.3.

  (ii) Case $H_0(\exists \varphi' \in F_3 \varphi')$: As $H_0(\exists \varphi' \in F_3 \varphi') \equiv (H_0 \exists \varphi' \in F \left( Y_F \varphi' \right))$, the set $A_0$ can be constructed as $A_{H_0 \exists \varphi' \in F} \cup \forall \varphi' \in F A_{Y_F \varphi'}$, where each $A_{Y_F \varphi'}$ is obtained from $A_{F \varphi'}$ using Lemma 3.3.

  (iii) Case $H_0(\forall \varphi' \in H_0 \varphi')$: It holds that $H_0(\forall \varphi' \in H_0 \varphi') \equiv (H_0 - t) \lor \forall \varphi' \in H_0 \left( Y_{H_0} \varphi' \right)$. Therefore, the set $A_0$ can be constructed as $A_{H_0 - t} \cup \forall \varphi' \in H_0 \left( A_{Y_{H_0} \varphi'} \right)$ where each $A_{Y_{H_0} \varphi'}$ is obtained from $A_{H_0 \varphi'}$ using Lemma 3.3.

  (iv) Case $H_0(\exists \varphi' \in H_0 \varphi')$: As $H_0(\exists \varphi' \in H_0 \varphi') \equiv (H_0 \exists \varphi' \in H_0 \left( Y_{H_0} \varphi' \right))$, the set $A_0$ can be constructed as $A_{H_0 \exists \varphi' \in H_0} \cup \forall \varphi' \in H_0 \left( A_{Y_{H_0} \varphi'} \right)$ where each $A_{Y_{H_0} \varphi'}$ is obtained from $A_{H_0 \varphi'}$ using Lemma 3.3.

  (v) Case $H_0(\forall \varphi' \in F_3 \varphi' \lor \forall \varphi' \in H_0 \varphi')$: As $H_0(\forall \varphi' \in F_3 \varphi' \lor \forall \varphi' \in H_0 \varphi') \equiv (H_0 - t) \lor \forall \varphi' \in F \left( Y_F \varphi' \right) \lor \forall \varphi' \in H_0 \left( Y_{H_0} \varphi' \right)$, the set $A_0$ can be constructed as $A_{H_0 - t} \cup \forall \varphi' \in F \left( A_{Y_F \varphi'} \cup A_{Y_{H_0} \varphi'} \right)$, where each $A_{Y_{H_0} \varphi'}$ is obtained from $A_{F \varphi'}$, and each $A_{Y_{H_0} \varphi'}$ is obtained from $A_{H_0 \varphi'}$ using Lemma 3.3.

  (vi) Case $H_0(\exists \varphi' \in F_3 \varphi' \lor \forall \varphi' \in H_0 \varphi')$: As $H_0(\exists \varphi' \in F_3 \varphi' \lor \forall \varphi' \in H_0 \varphi') \equiv (H_0 \exists \varphi' \in F \left( Y_F \varphi' \right) \lor \forall \varphi' \in H_0 \left( Y_{H_0} \varphi' \right))$, the set $A_0$ can be constructed as $A_{H_0 \exists \varphi' \in F} \cup \forall \varphi' \in F \left( A_{Y_F \varphi'} \cup A_{Y_{H_0} \varphi'} \right)$, where each $A_{Y_{H_0} \varphi'}$ is obtained from $A_{F \varphi'}$, and each $A_{Y_{H_0} \varphi'}$ is obtained from $A_{H_0 \varphi'}$ using Lemma 3.3.

$G_3$ Case $H_0 \varphi_2$: A pointed word $(u, i)$ satisfies $H_0(\varphi_2)$ iff $i = 0$ or $(u, 0)$ satisfies $G_0 \varphi_2$. Hence, $A_0 = A_{H_0 - t} \cup A_{P_5 (H_0 - t) \land (G_0 \varphi_2)}$ where $A_{P_5 (H_0 - t) \land (G_0 \varphi_2)}$ is obtained from $A_{H_0 - t}$ and $A_{G_0 \varphi_2}$ using Lemma 3.3.

$H_0$ Case $H_0 \varphi_2$: As $H_0(\varphi_2) \equiv (H_0 - t) \lor (YH_0 \varphi_2)$, the set $A_0$ can be constructed as $A_{H_0 - t} \cup A_{YH_0 \varphi_2}$, where $A_{YH_0 \varphi_2}$ is obtained from $A_{H_0 \varphi_2}$ using Lemma 3.3.

\[ \square \]

Remark 3.5 In other words, we have just shown that $\text{LTL}(F_3, P_5)$ is a semantic subset (with respect to global equivalence) of every formalism that is (i) able to express $p$, $G_p$, $H_p$, and $G_F p$, where $p \in \text{LTL}()$; and (ii) is closed under disjunction, conjunction, and applications of $\lor_\omega$, $\land_\omega$, $\lor p \land_\omega$, and $p S_\omega$, where $p \in \text{LTL}()$.

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 $P\alpha$-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 $\text{P}\alpha$-formula is decidable.

Proof. A run over a nonempty (finite or infinite) word $u = a_0 a_1 a_2 \ldots$ satisfies a formula $\varphi$ if and only if $(u, 0) \models \varphi$. Moreover, $(u, 0) \models \text{P} \alpha(\eta, \delta, \beta)$ if and only if $(\alpha(\delta, \beta)).$ Let $\eta = \alpha(\delta, \beta)$. It follows from the semantics of LTL that $(a_0, 0) \models \eta$ if and only if $(a_0, 0) \models \alpha(\delta, \beta)$. Therefore, the problem is to check whether $P_i = S$ for all $i < m$ and whether the given wPRS system has a run satisfying $t_m \land \alpha(\delta, \beta)$. As $t_m \land \alpha(\delta, \beta)$ can be easily translated into a disjunction of $\alpha$-formulae, Theorem 3.1 finishes the proof.

As LTL($F_a, P_a$) 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($F_s, P_s$) is decidable.

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

Theorem 3.8 Let $\Delta$ be a wPRS and $pt$ be a reachable nonterminal state of $\Delta$. The problem whether $L(pt, \Delta)$ contains a pointed word $(u, i)$ satisfying any given $\text{P} \alpha$-formula is decidable.

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

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

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

It is easy to see that $(u, i)$ is in $L(pt, \Delta)$ if and only if $u = a_0 a_1 \ldots a_{i-1} a_i, f, a_i, a_{i+1} \ldots$ is in $L(\Delta')$. Hence, for any given $\text{P} \alpha$-formula $\varphi = \text{P} \alpha(\eta, \delta, \beta)$ we construct a $\text{P} \alpha$-formula $\varphi' = \text{P} \alpha(\eta, \delta, \beta)$.

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

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($F_s, P_s$), i.e. the fragment with modalities $\text{strict eventually, eventually in the strict
past, and derived modalities strict always and always in the strict past.

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(\(F, P_\alpha\)) semantically coincides with formulae of First-Order Monadic Logic of Order containing at most 2 variables and no successor predicate (\(\text{FO}^2[\leq]\)), and that First-Order Monadic Logic of Order containing at most 2 variables (\(\text{FO}^2\)) coincides with an LTL(\(F, X, P, Y\)) fragment [8]. Further, let us recall our undecidability results for model checking of PA systems (a subclass of PRS) and fragments LTL(\(F, X\)) 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 \(\text{FO}^2[\leq]\) and undecidable for \(\text{FO}^2\). For the sake of completeness, we note that the First-Order Monadic Logic of Order containing at most 3 variables (\(\text{FO}^3\)) 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


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


