Electronic Communications of the EASST
Volume 64 (2013)

Proceedings of the XIII Spanish Conference on Programming and Computer Languages (PROLE 2013)

On the decidability of model checking LTL fragments in monotonic extensions of Petri nets

María Martos-Salgado and Fernando Rosa-Velardo

18 pages

Guest Editors: Clara Benac Earle, Laura Castro, Lars-Åke Fredlund
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

On the decidability of model checking LTL fragments in monotonic extensions of Petri nets

María Martos-Salgado[1] **and Fernando Rosa-Velardo**[2*]

[1] mrmartos@ucm.es
Universidad Complutense de Madrid

[2] fernandorosa@sip.ucm.es
Universidad Complutense de Madrid

Abstract: We study the model checking problem for monotonic extensions of Petri Nets, namely for two extensions of Petri nets: reset nets (nets in which places can be emptied by the firing of a transition with a reset arc) and ν-Petri nets (nets in which tokens are pure names that can be matched with equality and dynamically created). We consider several fragments of LTL for which the model checking problem is decidable for P/T nets. We first show that for those logics, model checking of reset nets is undecidable. We transfer those results to the case of ν-Petri nets. In order to cope with these negative results, we define a weaker fragment of LTL, in which negation is not allowed. We prove that for that fragment, the model checking of both reset nets and ν-Petri nets is decidable, though with a non primitive recursive complexity. Finally, we prove that the model checking problem for a version of that fragment with universal interpretation is undecidable even for P/T nets.

Keywords: LTL, model checking, Petri nets, decidability, complexity

1 Introduction

Temporal logics [4] have been established as a very expressive formalism for the specification of properties of computational concurrent systems. Model checking is the problem of deciding if a given system satisfies a given temporal formula.

For infinite state systems the model checking problem is undecidable in general [15]. A very well known formalism for infinite state concurrent systems is that of Petri nets [6]. Among them, place/Transition nets (P/T nets) are potentially infinite state, but their expressive power is below Minsky machines (e.g., reachability is decidable for them [8]). Decidability and complexity of the model checking problem for P/T nets are well studied, and the corresponding decidability frontiers are well settled [13, 9, 12, 8]. Roughly speaking, model checking of P/T nets is undecidable for any branching-time logic, while for linear time logics, "event-based" LTL is decidable, though "state-based" LTL is undecidable.

In the last two decades, several monotonic extensions of Petri nets have appeared in the literature. These extensions usually consist either on the extension of the firing rule of P/T nets,

* Authors supported by the Spanish projects STRONGSOFT TIN2012-39391-C04-04 and PROMETIDOS S2009/TIC-1465.

or on the use of colours, that is, distinguishable tokens. We consider two simple extensions of P/T nets, one in each group: reset nets [7] and ν-Petri nets (ν-PN) [21]. In reset nets the firing of a transition can empty some places. Their modeling capabilities are discussed for instance in [14]. Tokens in ν-PNs are pure names, that can be created fresh, moved along the net and used to restrict the firing of transitions with name matching. Names can be seen as process identifiers [19], so that ν-PN can serve as the basis of models in which an unbounded number of components (which are in turn unbounded) synchronize. For example, they can be used to model resource-constrained workflow nets, an extension of workflow nets in which an arbitrary number of instances of the workflow can be executed concurrently [11]. In [5], they are used to give a semantics to an extension of BPEL [16] with instance isolation.

In this paper we study the decidability of the model checking problem of these models. More precisely, we consider the logics for which model checking of P/T nets is decidable, and we study their decidability for the two extensions. In particular, we study LTL_f, which is the fragment of LTL that uses only *first* as basic predicate [9], $\mathscr{L}(\mathbf{F})$, the fragment of LTL in which negation is only applied to basic predicates (not to operators), and the operators are **X** (next), **F** (eventually), $\wedge$ and $\vee$ [12]; and $\mathscr{L}(\mathbf{GF})$, which is the fragment of LTL in which the only allowed composed operator is **GF** (globally future), the operators are **F**, $\vee$ and $\wedge$ and negation is only applied to basic predicates [13].

Unfortunately, we conclude that the decidability results for P/T nets cannot be adapted, so that model checking for any of the logics considered is undecidable. In particular, we reduce LTL_f model checking of lossy inhibitor nets, which is undecidable, to LTL_f model checking of reset nets. Moreover, we prove that repeated coverability and reachability, which are undecidable for reset nets, can be expressed in $\mathscr{L}(\mathbf{GF})$ and $\mathscr{L}(\mathbf{F})$ respectively.

As a first step to mitigate the previous undecidability result, we consider a fragment of LTL weaker than all the logics considered so far in which, in particular, we do not allow negations. We call that logic $\mathbf{F}_{cov}$. We prove that the model checking problem for this logic is decidable for both models.

In some of the subclasses of LTL considered in the literature [13, 12] a formula is said to be satisfied if there exists one run that satisfies it, as opposed to the more standard definition of LTL in which all runs are required to satisfy the formula. Even though the two interpretations are equivalent when negation can be used without restriction, this is not the case for the considered subclasses of LTL, neither for $\mathbf{F}_{cov}$. We justify this definition by proving that already for P/T nets, $\mathbf{F}_{cov}$ model checking is undecidable under the universal interpretation.

Table 1 summarizes the results on model checking of P/T nets, reset nets and ν-PNs. By "+" (resp. -) we denote that the model checking problem for the considered logic is decidable (resp. undecidable). If the references of the results are not given, then the result is either new (the ones with signs in bold letters) or follows directly from other results of the table.

Outline: The rest of the paper is structured as follows. Section 2 presents some basic results and notations we use throughout the paper. Section 3 proves undecidability of the model checking problem for the considered logics, for reset nets and ν-PN. Section 4 defines $\mathbf{F}_{cov}$ and proves decidability of the model checking problem for reset nets and ν-PN, and undecidability for P/T nets in the universal case. In Section 5 we present our conclusions.

	P/T	**Reset**	**ν-PN**
LTL	- [9]	-	-
LTL_f	+ [9]	- [3]	-
$\mathscr{L}(\mathbf{GF})$	+ [13]	-	-
$\mathscr{L}(\mathbf{F})$	+ [12]	-	-
$\mathbf{F}_{cov}$	+	+	+
$\forall\mathbf{F}_{cov}$	-	-	-

Table 1: Summary of results. Those in bold signs are the new contributions of this paper.

2 Preliminaries

A *quasi order* (qo) is a reflexive and transitive binary relation. For a qo $\leq$, we write $a < b$ if $a \leq b$ and $b \not\leq a$.

Labelled transition systems. A *transition system* is a tuple $\mathscr{S} = (S, L, \rightarrow, init)$, where S is a (possibly infinite) set of states, L is a set of labels, $init \in S$ is the initial state and $\rightarrow \subseteq S \times L \times S$ [1]. Given two states $s_1, s_2 \in S$ and $a \in L$, we write $s_1 \xrightarrow{a} s_2$ instead of $(s_1, a, s_2) \in \rightarrow$, and $s_1 \rightarrow s_2$ if $s_1 \xrightarrow{b} s_2$ for some $b \in L$. We denote by $\rightarrow^*$ the reflexive and transitive closure of $\rightarrow$ and by $\rightarrow^+$ the transitive closure of $\rightarrow$. A *run* π of $\mathscr{S}$ is any sequence $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \ldots$ such that $s_i \xrightarrow{a_i} s_{i+1}$ for $i \geq 0$. We define the length of a run π as $n \in \mathbb{N}$ if $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_n} s_n$ is a finite run, and ∞ if π is an infinite run. The *reachability problem* consists in deciding for a given state s_f whether $init \rightarrow^* s_f$. If S is endowed with a qo $\leq$ we can define the *coverability problem*, that consists in deciding, given $s_f \in S$, whether some $s \geq s_f$ is reachable. Then, we say that s covers s_f. The *repeated coverability problem* is the problem of deciding whether a given state is covered infinitely often in some infinite run starting in *init*, that is, for a given s there is an infinite run $init \rightarrow^+ s_1 \rightarrow^+ s_2 \rightarrow^+ \ldots$ such that $s \leq s_i$ for all $i \geq 1$.

Multisets. Given a (possibly infinite) arbitrary set A, we denote by $A^{\oplus}$ the set of finite multisets over A, that is, the mappings $m : A \rightarrow \mathbb{N}$ for which $supp(m) = \{a \in A \mid m(a) > 0\}$ is finite. When needed, we identify each conventional set with the multiset defined by its characteristic function, and use set notation for multisets when convenient, with repetitions to account for multiplicities greater than one. Moreover, given two sets A and B, and an injection $\alpha : A \rightarrow B$, sometimes we interpret α as the function $\alpha : A^{\oplus} \rightarrow B^{\oplus}$ such that given $M_A \in A$, $\alpha(M_A) = M_B$, where for each $b \in B$, $M_B(b) = n > 0$ if there exists $a \in A$ with $\alpha(a) = b$ and $M_A(a) = n$, and $M_B(b) = 0$ otherwise.

Given two multisets $m_1, m_2 \in A^{\oplus}$ we denote by $m_1 + m_2$ the multiset defined by $(m_1 + m_2)(a) = m_1(a) + m_2(a)$. We define multiset inclusion as $m_1 \subseteq m_2$ if $m_1(a) \leq m_2(a)$ for all $a \in A$. If $m_1 \subseteq m_2$, we can define $m_2 - m_1$, taking $(m_2 - m_1)(a) = m_2(a) - m_1(a)$. We denote by $\emptyset \in A^{\oplus}$ the empty multiset, given by $\emptyset(a) = 0$ for all $a \in A$.

Petri nets. A *Place/Transition net* (P/T net for short) is a tuple $N = (P, T, F)$ where P is a finite set of places, T is a finite set of transitions (with $P \cap T = \emptyset$) and $F : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$ is

[1] We use transition labels to homogeneously define logics based on states and events.

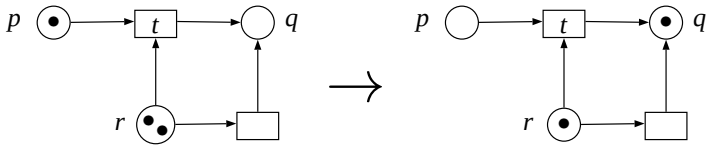

Figure 1: The firing of a transition in a P/T net.

the flow function. Given $t \in T$, the multiset of preconditions of t is ${}^\bullet t \in P^\oplus$ given by $({}^\bullet t)(p) = F(p,t)$. Analogously, the postconditions of t are given by $(t^\bullet)(p) = F(t,p)$. A marking is any $m \in P^\oplus$. For Petri nets we consider the order between markings given by multiset inclusion. This order defines the standard coverability problem for Petri nets. We say that a transition t is enabled at a marking m if for each $p \in P$, $m(p) \geq F(p,t)$. In that case, t can be fired from m, reaching a new marking m', which is denoted by $m \xrightarrow{t} m'$, where m' is given by $m'(p) = (m(p) - F(p,t)) + F(t,p)$. The reachability, the coverability and the repeated coverability problems are all decidable for P/T nets [8]. In the rest of this paper, we represent places as circles, transitions as rectangles, the flow function as arrows and markings as tokens inside places. The arcs which represent the flow function are not labelled by any constant representing the corresponding values of F. That is because for all $p \in P$, $t \in T$ with an arc going from p to t (resp. from t to p) in figures, we assume $F(p,t) = 1$ (resp. $F(t,p) = 1$).

Example 1 *In the P/T net in the left-hand side of Fig. 1, transition t is enabled, so it can be fired reaching the marking depicted in the right-hand side. Tokens have been consumed from p and r, and a token has been produced in q. As there is not a priority order over transitions, note that from the marking represented in the left-hand side, the other transition could have been fired instead of t, reaching a marking with a token in each place of the net.*

Now, we explain two extensions of P/T nets, namely reset nets and inhibitor nets. Both of them are defined from P/T nets by adding special arcs: reset arcs, which empty a place, and inhibitor arcs, which add to the enabling conditions the requirement that a certain place is empty. For both extensions, the concepts of preconditions, postconditions and marking are analogous to those for P/T nets.

A *reset net* is a tuple $N = (P,T,F,R)$, where (P,T,F) is a P/T net and $R \subseteq P \times T$ is a relation containing the so called *reset arcs*. The enabled transitions at a marking are defined as for P/T nets. An enabled transition t can be fired from a marking m reaching m' given by:

$$m'(p) = \begin{cases} (m(p) - F(p,t)) + F(t,p) & \text{if } (p,t) \notin R \\ F(t,p) & \text{if } (p,t) \in R \end{cases}$$

Notice that if $(p,t) \in R$ and $F(t,p) = 0$ (i.e., t does not put any token in p) then p has no tokens after the firing of t (hence p is reset).

Example 2 *Focus on Fig. 2. The double arc represents a reset arc from place r to t, that is, $(r,t) \in R$. Then, transition t is enabled in the marking represented in the net in the left-hand side*

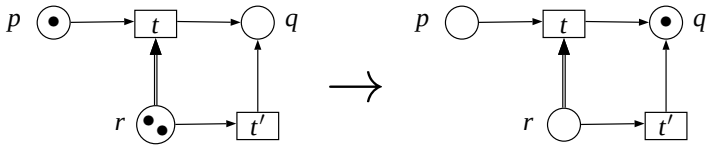

Figure 2: The firing of a transition in a reset net.

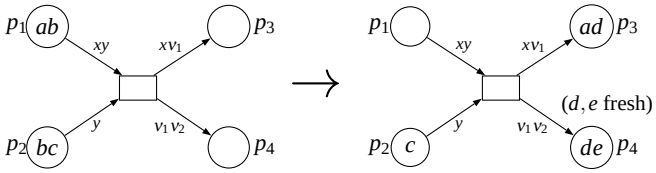

Figure 3: A simple ν-PN

of the figure, and it can be fired, reaching the marking depicted in the right. All tokens in place r have been consumed in the firing of t due to the presence of the reset arc. Note that transition t' is also enabled in the marking represented in the net in the left-hand side. Transitions with reset arcs do not have any priority over the rest of the enabled ones. Therefore, transition t' could be fired from the first marking too.

An *inhibitor net* is a tuple $N = (P,T,F,I)$ where $N = (P,T,F)$ is a P/T net, and $I \subseteq P \times T$ is a relation containing the *inhibitor arcs* (also called zero tests). We say that a transition t is enabled at a marking m if:

- for each $p \in P$, $m(p) \geq F(p,t)$ and
- for each $(q,t) \in I$, $m(q) = 0$ (place q is empty).

Then, t can be fired from m, reaching the marking m' given by $m'(p) = (m(p) - F(p,t)) + F(t,p)$ (as for P/T nets) [2]. Inhibitor nets with two inhibitor arcs are already Turing complete [15], though, interestingly, inhibitor nets with only one inhibitor arc are not [18].

ν-PN. Another way in which P/T nets are extended in the literature is by considering distinguishable tokens. Perhaps the most simple extension of P/T nets with (arbitrarily many) distinguishable tokens are ν-Petri Nets [21], that encompass unboundedly many names (via a mechanism for fresh name creation) and the unbounded occurrence of each name.

Let *Var* be a set of variables, and $\Upsilon \subset Var$ a set of special variables for fresh name creation. A *ν-Petri Net* (ν-PN for short) is a tuple $N = (P,T,F)$, where P and T are finite disjoint sets, and $F : (P \times T) \cup (T \times P) \to Var^{\oplus}$ labels every arc by a multiset of variables. We denote $pre(t) = \bigcup_{p \in P} supp(F(p,t))$ (the set of variables in pre-arcs) and $post(t) = \bigcup_{p \in P} supp(F(t,p))$ (the set of variables in post-arcs). We also take $Var(t) = pre(t) \cup post(t)$.

[2] Actually, it is straightforward to simulate a reset arc using inhibitor arcs, in a way preserving reachability, coverability and repeated coverability.

Let Id be an infinite set of names. A *marking* of a ν-PN is a mapping $M : P \to Id^{\oplus}$ assigning to each place the multiset of tokens currently in it. We take $Id(M) = \bigcup_{p \in P} supp(M(p))$, that is, the set of names in M.

Given a transition $t \in T$ of a ν-PN, a *mode* for t is any injection $\sigma_t : Var(t) \to Id$. As modes are injections, we can match names with equality (just by using the same variable more than once) and with inequality (by using different variables). We say that a transition t is enabled with a mode σ_t for a marking M, if for each $v \in \Upsilon$, $\sigma_t(v) \notin Id(M)$ and for all $p \in P$, $\sigma_t(F(p,t)) \subseteq M(p)$. Then, t can be fired, and a new marking M' is reached, given by $M'(p) = (M(p) - \sigma_t(F(p,t))) + \sigma_t(F(t,p))$ for all $p \in P$. In that case we write $M \xrightarrow{t} M'$.

Example 3 *Fig. 3 depicts the ν-PN N given by $N = (\{p_1,p_2,p_3,p_4\},\{t\},F)$ with $F(p_1,t) = \{x,y\}$, $F(p_2,t) = \{y\}$, $F(t,p_3) = \{x,v_1\}$, $F(t,p_4) = \{v_1,v_2\}$, and for $(n,m) \in \{(p_3,t),(p_4,t),(t,p_1),(t,p_2)\}$, $F(n,m) = \emptyset$. We assume that $v_1,v_2 \in \Upsilon$. The initial marking is given by $M_0(p_1) = \{a,b\}$, $M_0(p_2) = \{b,c\}$ and $M_0(p_3) = M_0(p_4) = \emptyset$. The transition is fired with respect to a mode σ given by $\sigma(x) = a$, $\sigma(y) = b$, $\sigma(v_1) = d$ and $\sigma(v_2) = e$. Note that names d and e are not in the initial marking, and therefore they are created new.*

Intuitively, each name in a marking of a ν-PN can represent a different process running in the same net. Therefore, we can represent the synchronization between processes and the creation of new ones.

Given a marking M of a ν-PN and a set I of names, a renaming of M is any injection $\alpha : Id(M) \to I$. We say that $M \sqsubseteq M' \Leftrightarrow$ there is a renaming α of M such that $\alpha(M)(p) \subseteq M'(p)$ for all $p \in P$. With this order, coverability is decidable for ν-PN, though reachability is undecidable for them [21].

Example 4 *The marking M given by $M(p_1) = \emptyset$, $M(p_2) = \emptyset$, $M(p_3) = \{a,d\}$, $M(p_4) = \{a,c\}$ is covered by the marking M' on the right-hand side of Fig. 3. Note that, although there is not a token of name a in place p_4, with the order we have defined, M is covered by M' by considering the renaming α, such that $\alpha(a) = d$, $\alpha(d) = a$ and $\alpha(e) = c$.*

Finally, note that both models induce labelled transition systems in the obvious way: the states of the labelled transition systems are the markings of the nets, the set of labels is the set of transitions, and if $\mathscr{M}$ is the set of markings of the net, the transition relation $\to \subseteq \mathscr{M} \times T \times \mathscr{M}$ is the one such that $m_1 \xrightarrow{t} m_2$ in the net $\Leftrightarrow (m_1,t,m_2) \in \to$.

3 Model checking of Petri net extensions

Temporal logics are used to specify dynamic properties of systems. They consider a set of atomic propositions (which express atomic properties), the boolean operators and several temporal operators and path quantifiers, which allow us to express temporal properties. There are mainly two kinds of temporal logics: linear time logics and branching time logics. The properties that are expressed in branching time logics are about the computation tree [1], while the ones expressed in linear time logics are about the runs [17] (where we consider the notion of run defined in the preliminaries). In this paper we focus on linear time logics because model checking of P/T nets

with branching time logics is undecidable, even for very simple fragments [9].

The basic temporal operators used in temporal logics are **X**, **F** and **U**[3]. The most representative example of a linear time logic is LTL.

Definition 1 An LTL formula is either an atomic proposition or a formula of the form $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\mathbf{X}\varphi$, $\mathbf{F}\varphi$, or $\varphi\mathbf{U}\psi$, where φ and ψ are LTL formulae.

First, we explain the semantics of the temporal operators informally. LTL formulae are interpreted over maximal runs, i.e., runs that are either infinite, or end in a deadlock state. Let φ be an LTL formula, $\mathscr{S}$ a transition system and π a maximal run starting in s. We write $\mathscr{S}, \pi \vDash \varphi$ to denote that π satisfies φ:

- $\mathscr{S}, \pi \vDash \mathbf{X}\varphi$ (next) holds if the property φ holds in the state that follows s in π.
- $\mathscr{S}, \pi \vDash \mathbf{F}\varphi$ (eventually) holds if the property φ holds in some state of π.
- $\mathscr{S}, \pi \vDash \varphi\mathbf{U}\psi$ (until) holds if there is a state of the run π such that ψ holds in that state, and φ holds at every preceding state on the run.

We also define **G** (globally) as $\mathbf{G}\varphi = \neg\mathbf{F}\neg\varphi$, so that $\mathscr{S}, \pi \vDash \mathbf{G}\varphi$ holds if φ holds in every state of π.

Now, we give the formal semantics of temporal operators. Let $\pi = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots$ be a finite or infinite run of length $n \in \mathbb{N} \cup \{\infty\}$. For $n > 0$, π^i is defined as the suffix of the run π starting by the i-th state s_i. We suppose that the states and the actions of a system are labelled by the atomic propositions that they satisfy, that is, if a state s (or an action a) satisfies an atomic proposition p, then $p \in L(s)$ ($p \in L(a)$ resp.), where $L(s)$ ($L(a)$) represents the labels of s (a). The formal definition of the semantics of the previous operators is inductively defined as [4]:

- $\mathscr{T}, \pi \vDash p \Leftrightarrow p \in L(s_0) \vee \pi^1$ is defined (that is, the length of π is greater than 0) and $\in L(a_1)$.
- $\mathscr{T}, \pi \vDash \neg\varphi_1 \Leftrightarrow \mathscr{T}, \pi \nvDash \varphi_1$.
- $\mathscr{T}, \pi \vDash \varphi_1 \vee \varphi_2 \Leftrightarrow \mathscr{T}, \pi \vDash \varphi_1$ or $\mathscr{T}, \pi \vDash \varphi_2$.
- $\mathscr{T}, \pi \vDash \varphi_1 \wedge \varphi_2 \Leftrightarrow \mathscr{T}, \pi \vDash \varphi_1$ and $\mathscr{T}, \pi \vDash \varphi_2$.
- $\mathscr{T}, \pi \vDash \mathbf{X}\varphi_1 \Leftrightarrow \pi^1$ is defined and $\mathscr{T}, \pi^1 \vDash \varphi_1$.
- $\mathscr{T}, \pi \vDash \mathbf{F}\varphi_1 \Leftrightarrow$ there exists a $k \geq 0$ such that $\mathscr{T}, \pi^k \vDash \varphi_1$.
- $\mathscr{T}, \pi \vDash \mathbf{G}\varphi_1 \Leftrightarrow$ for all $i \geq 0$ such that π^i is defined, $\mathscr{T}, \pi^i \vDash \varphi_1$.
- $\mathscr{T}, \pi \vDash \varphi_1\mathbf{U}\varphi_2 \Leftrightarrow$ there exists a $k \geq 0$ such that π^k is defined, $\mathscr{T}, \pi^k \vDash \varphi_2$ and for all $0 \leq j < k, \mathscr{T}, \pi^j \vDash \varphi_1$.

[3] Even though **F** can be defined in terms of **U**, we prefer to include it as a primitive temporal quantifier, since we will later disallow **U**.

Note that in the previous definition, we assign the propositions they satisfy to both the states and the transition labels (a), defined for the labelled transition systems. The atomic propositions usually considered for P/T nets, are given by the following predicates:

- $cov(m)$, where m is a marking: $cov(m)$ holds in π if the first marking in π covers m.
- $first(t)$, where t is a transition: $first(t)$ holds in π if the first transition fired in π is t.

Some works consider the atomic propositions $ge(p,n)$ and $en(t)$ [9]. The first one expresses that there are at least n tokens in place p, and the second states that t is enabled. We will not consider them since they are equivalent to $cov(\{p,\overset{n}{\ldots},p\})$ and $cov(^\bullet t)$, respectively.

According to the standard definition, a system $\mathscr{S}$ satisfies an LTL formula φ, denoted by $\mathscr{S} \models \varphi$ iff every maximal run of the system starting in the initial state *init* satisfies it (universal interpretation). The model checking problem consists in deciding, given $\mathscr{S}$ and φ, whether $\mathscr{S} \models \varphi$. The model checking problem is equivalent to deciding, given $\mathscr{S}$ and φ, the existence of one run starting in *init* satisfying the formula (existential interpretation), provided negation can be used without restriction, since $S \models \varphi$ (under the universal interpretation) iff $S \not\models \neg\varphi$ (under the existential interpretation).

We consider different LTL fragments, built depending on which predicates and operators we consider.

Definition 2 We consider the following fragments of LTL:

- LTL_f, the fragment of LTL that uses only *first* as basic predicate [9],
- $\mathscr{L}(\mathbf{F})$, the fragment of LTL in which negation is only applied to basic predicates (not to operators), and the operators are **X**, **F**, $\wedge$ and $\vee$ [12],
- $\mathscr{L}(\mathbf{GF})$, the fragment of LTL in which the only allowed composed operator is **GF**, the operators are **F**, $\vee$ and $\wedge$ and negation is only applied to basic predicates [13].

Example 5 *The formula* $\mathbf{F}first(t)$*, which expresses that* t *is eventually fired, is in* $\mathscr{L}(\mathbf{F})$*, but* $\mathbf{G}first(t) = \neg\mathbf{F}\neg first(t)$*, which expresses that* t *is always fired, is not.*

The formula $first(t) \rightarrow \mathbf{GF}first(t)$*, which expresses that if* t *is the first transition being fired then* t *is fired infinitely often, is in* $\mathscr{L}(\mathbf{GF})$*, but* $\mathbf{F}first(t) \rightarrow \mathbf{GF}first(t) = \neg\mathbf{F}first(t) \vee \mathbf{GF}first(t)$*, which expresses that if* t *is eventually fired, then it is fired infinitely often, is not.*

In LTL_f negation can be used without restriction, so that the universal and the existential interpretations are equivalent (i.e., their model checking decision problems are equivalent). However, this is not the case for $\mathscr{L}(\mathbf{F})$ and $\mathscr{L}(\mathbf{GF})$. Actually, they are defined using the existential interpretation in [12, 13]. For the subclasses of LTL considered, we have the following results.

- LTL model checking is undecidable (with both *first* and *cov*), but LTL_f model checking is decidable (with *cov* only) [9].
- $\mathscr{L}(\mathbf{F})$ model checking is decidable [12] (with existential interpretation),
- $\mathscr{L}(\mathbf{GF})$ model checking is decidable [13] (with existential interpretation).

3.1 Model checking of reset nets

Let us show that the three logics which are decidable for P/T nets become undecidable for reset nets. Let us first consider LTL_f. In [3] the model checking problem of LTL_f is studied for lossy vector addition systems (lossy VAS) with tests for zero, which is proved to be undecidable. Let us see that we can adapt that result for reset nets. Let us first define the lossy version of a transition system in general.

Definition 3 Given a transition system $\mathscr{S} = (S, \rightarrow, init)$ and a quasi-order $\leq$ over S, the *lossy version* of $\mathscr{S}$ is $\mathscr{S}_l = (S, \rightarrow_l, init)$, where $s_1 \rightarrow_l s_2$ if and only if there exists two states s_1' and s_2' such that $s_1 \geq s_1' \rightarrow s_2' \geq s_2$. A *lossy Petri net* is the lossy version of some Petri net.

In the lossy version of a transition system, states can be spontaneously decreased. In the case of Petri nets, tokens may be lost just before or after a transition is fired.

Example 6 *Focus on Fig. 2. In the lossy version of the P/T net obtained by replacing the reset arc by a plain arc, despite r is not reseted by any reset arc of t, the second marking could be reached from the first one by first losing a token from r and then firing t.*

Reset nets can simulate lossy inhibitor nets, as we prove next. This fact is used in the proof of the next result.

Proposition 1 *LTL_f model checking is undecidable for reset nets.*

Proof. We reduce LTL_f model checking for lossy inhibitor nets, which is undecidable [3], to the same problem for reset nets. Let $N = (P, T, F, I)$ be an inhibitor net. We define the reset net $N' = (P, T, F, I)$. We are going to prove that there is a surjective function between the runs of N and N' that preserves the sequence of labels of runs and therefore, since the only atomic proposition in LTL_f is *first*, given an LTL_f formula φ, $N \models \varphi$ iff $N' \models \varphi$. Notice that since N' is a reset net, checks for zero have been replaced by resets, that is, an inhibitor arc from a place p to a transition t in N is replaced by a reset arc from p to t in N'. The following holds:

- $m_1 \xrightarrow{t} m_2$ in $N \Rightarrow$ there is an $m_2' \geq m_2$ such that $m_1 \xrightarrow{t} m_2'$ in N': the preconditions and effects of the firings of t in N and N' are the same, except for the fact that we have replaced inhibitor arcs by reset arcs. Therefore, when a transition with an inhibitor arc in N is firing in N', the corresponding place is reseted instead of being checked for zero. If there are tokens in such a place, all the tokens in it are removed, so there exist a possibility of losing tokens in our simulation if the place was not empty. If t has no inhibitor arc, and $m_1 \xrightarrow{t} m_2$ in N, t is enabled at m_1 in N', and it can be fired reaching a marking greater or equal than m_2, because some token may have been lost in the firing in N.

  Now, suppose that t has some inhibitor arcs, and $m_1 \xrightarrow{t} m_2$ in N. First of all, note that if a transition is enabled in N at a marking, it is enabled in N' at the same marking. Moreover, the places with inhibitor arcs leading to t are empty in that marking. Therefore, when t is fired from m_1 in N', these places are reset (hence staying empty). Therefore, the only differences in the effects of the firing of t in both nets come because of lossiness. In

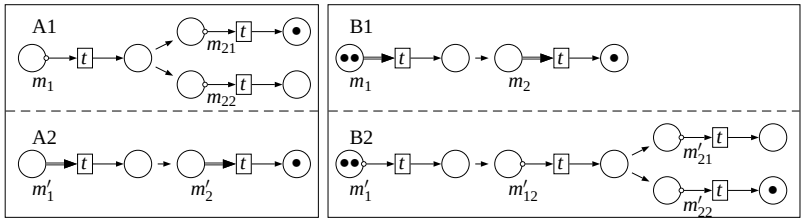

Figure 4: From lossy inhibitor nets to reset nets

particular, since t may loose tokens in N, but not in N', $m_1 \xrightarrow{t} m'_2$ in N', with $m'_2 \geq m_2$.

For example, focus on the left-hand side of Fig. 4. In the inhibitor nets depicted in this figure, given a place p and a transition t, we represent $(p,t) \in I$ by an arc from p to t, with a circle in the place p. The marking m_1 in the first lossy inhibitor net in A1 may evolve to m_{21} or m_{22} (m_{22} has lost the token). The corresponding reset net, depicted in A2 can only evolve t the marking m'_2. However, note that m'_2 covers both m_{21} and m_{22}.

- $m_1 \xrightarrow{t} m_2$ in $N' \Rightarrow m_1 \xrightarrow{t} m_2$ in N: In particular, if N' resets a place, N can first loose tokens, thus emptying it, and then test for zero in that place. Therefore, the transition is fireable in N (because the preconditions of the firings of t in N and N' are the same, except for the lossiness and the inhibitor arcs) and the place is empty at the end of both firings.

  Focus on the right-hand side of Fig. 4. Transition t is fireable from the marking m_1 in the reset net in B1, reaching the marking m_2. Despite t is not fireable from $m'_1 = m_1$, the lossy inhibitor net may lose tokens, reaching m'_{12}, from which t can be fired, reaching the markings m'_{21} or m'_{22}.

Therefore, there is a surjective function between the runs of N and N' that preserves the sequence of labels of runs. Note that this holds because reset nets are monotonic, so transitions which are fired in N at a marking m which has lost tokens, can be fired from the corresponding marking m' of N', without loss of tokens, because $m' \geq m$. Since the only atomic proposition in LTL_f is *first*, $N \models \varphi$ iff $N' \models \varphi$ and we conclude. □

Let us now focus on the two other fragments of LTL. The case for $\mathscr{L}(\mathbf{GF})$ is straightforward:

Proposition 2 *$\mathscr{L}(\mathbf{GF})$ model checking of reset nets is undecidable.*

Proof. It is enough to consider that $\mathbf{GF}cov(M)$, which is a formula in $\mathscr{L}(\mathbf{GF})$, expresses the repeated coverability problem, which is undecidable for reset nets [2]. □

However, not only that fragment, but the weaker fragment $\mathscr{L}(\mathbf{F})$, which is decidable for P/T nets [12], is undecidable for reset nets. The following proof uses ideas from [12], in which $\mathscr{L}(\mathbf{F})$ model checking is reduced to reachability for P/T nets.

Proposition 3 *$\mathscr{L}(\mathbf{F})$ model checking of reset nets is undecidable.*

Proof. We reduce reachability, which is undecidable for reset nets [2], to model checking some formula in $\mathscr{L}(\mathbf{F})$. Let $N = (P,T,F,R)$ be a reset net and m a marking of N. We can compute the set of the least markings greater than m.[4] Indeed, that set is just $\{m_p \mid p \in P\}$, where m_p is given by $m_p(q) = m(q)$ for $q \neq p$ and $m_p(p) = m(p)+1$. For example, the set of the least markings greater than the marking depicted in the net in the left-hand side of Fig. 2 is $\{m_p, m_q, m_r\}$, where $m_p = \{p,p,r,r\}$, $m_q = \{p,q,r,r\}$ and $m_r = \{p,r,r,r\}$. Then, m is reachable in N iff there is a reachable marking m' that covers m, but does not cover any m_p, because this would imply that in each place p, m' has exactly $m(p)$ tokens (because it does not cover m_p). Therefore, m is reachable in N iff the formula $\mathbf{F}(cov(m) \wedge \bigwedge_{p \in P} \neg cov(m_p))$ is satisfied. □

The previous proof is based on obtaining a formula which expresses the reachability problem. In order to obtain this formula, there is an important property that reset nets satisfy: Given a marking m, we are able to compute a finite set of markings $S = \{m_1, \ldots, m_n\}$ such that for each $m_i \in S$, $m_i \supset m$ and if $m' \supset m$, then there is $m_i \in S$ such that $m' \supseteq m_i$. Intuitively, we can compute the finite set of "the smallest markings greater than m". As we can build that set, a marking m_f coincides with m if and only if m_f covers m and m_f does not cover any marking of S.

3.2 Model checking of ν-PN

We first recall from [21] how ν-PN can simulate reset nets (see Fig. 5, where the double arrow represents a reset arc). For each place p of a reset net N we consider a copy of it and a new place p' in the ν-PN N' we build. The main idea of the construction is to store in place p' a single token of the colour that we consider valid in the current marking for place p of N'. That is, the tokens in p of the colour of the token in p' will be considered valid, and the rest of the tokens in p will be considered garbage. For example, in the ν-PN of Fig. 5, there are two valid tokens in place r, because the colour of the token in r' is b.

The construction of N' guarantees that for each place p of N, the place p' of N' contains a single token at any time. The firing of any transition ensures that the token being used in the place p of N' coincides with that in p' (by labelling both arcs with the same variable x_p). Every time a transition resets a place p of N, the token in p' is replaced by a fresh one, so that no token remaining in place p of N' can be used from then on. For example, suppose we fire transition t in Fig. 5. Then, a new colour is put in r', and therefore the tokens of name b in r cannot be used anymore.

Therefore, this simulation can introduce some garbage tokens (those in p when p is reset). Given a marking m we define m' by arbitrarily choosing a different name $a_p \in Id$ for each $p \in P$, and taking $m'(p') = \{a_p\}$, and $m'(p) = \{a_p, \overset{m(p)}{\ldots}, a_p\}$. Then, if m_0 is the initial marking of N, N'

[4] In order theory that set is called the cover of m, though we prefer not to overload that term here.

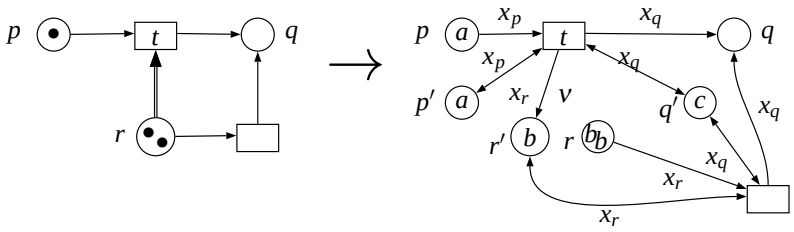

Figure 5: A reset net and the corresponding ν-PN. The double arrow represents a reset arc

with initial marking m'_0 simulates N. The previous simulation preserves all behavioral properties. Then, the following is a straightforward consequence of Prop. 1.

Corollary 1 *LTL_f model checking is undecidable for ν-PN.*

The previous simulation also preserves coverability. More precisely, if a marking m is coverable in the reset net N, then the marking m' of N' defined above is coverable too. The markings in the simulation may contain some garbage, which is created when we simulate the firing of a reset, because instead of removing all tokens of some place p, we change the name of the token in p', making all tokens in p become garbage. However, the presence of that garbage is irrelevant for coverability. In particular, we have the following.

Proposition 4 *Repeated coverability is undecidable for ν-PN.*

Proof. It is enough to consider that repeated coverability is undecidable for reset nets [2], and that the previous simulation preserves repeated coverability. We prove that m can be repeatedly covered from m_0 in N iff M' can be repeatedly covered from m'_0 in N'. Indeed, if m is repeatedly covered there is a run $m_0 \to^+ m_1 \to^+ m_2 \to^+ \ldots$ of N such that $m_i \geq m$ for all $i \geq 1$. By construction of N', there is a run $m'_0 \to^+ M_1 \to^+ M_2 \to^+ \ldots$ of N' such that $M_i \geq m'_i \geq m'$ for all $i \geq 1$, so m' is repeatedly covered. Moreover, M_i coincides with m'_i when considering only the "valid tokens" of M_i (and after possibly renaming the names carried by the tokens). The converse is analogous. □

Once we know that repeated coverability is undecidable, undecidability of $\mathscr{L}(\mathbf{GF})$ model checking is trivial.

Corollary 2 *$\mathscr{L}(\mathbf{GF})$ model checking of ν-PN is undecidable.*

Next, we see the undecidability of $\mathscr{L}(\mathbf{F})$ model checking for ν-PN.

Proposition 5 *$\mathscr{L}(\mathbf{F})$ model checking of ν-PN is undecidable.*

Proof. The proof is analogous to the one of the same result for reset nets. We reduce reachability in ν-PN, which is undecidable [21], to the model checking problem for some formula in $\mathscr{L}(\mathbf{F})$. Let $N = (P,T,F)$ be a ν-PN. Given a marking M, we can compute the finite set of the least markings greater than M. Indeed, given $p \in P$ and $c \in Id(M) \cup \{b\}$, with $b \notin Id(M)$, we define

M_{pc} given by $M_{pc}(q) = M(q)$ if $p \neq q$ and $M_{pc}(p) = M(p) + \{c\}$. Then, the set we are looking for is $\{M_{pc} \mid p \in P, c \in Id(M) \cup \{b\}\}$. For example, consider a net with two places p and q, and a marking M with one token a in p and empty in q. The set of the least markings greater than this marking is $\{M_{pa}, M_{pb}, M_{qa}, M_{qb}\}$, where $M_{pa}(p) = \{a,a\}$, $M_{pa}(q) = \emptyset$, $M_{pb}(p) = \{a,b\}$, $M_{pb}(q) = \emptyset$, $M_{qa}(p) = \{a\}$, $M_{qa}(q) = \{a\}$, $M_{qb}(p) = \{b\}$ and $M_{qb}(q) = \{a\}$. Therefore, M is reachable in N iff $N \models \mathbf{F}(cov(M) \wedge \bigwedge_{p,c} \neg cov(M_{pc}))$, and we conclude. □

4 A decidable fragment

In the previous section we have proved the undecidability of model checking of reset nets and ν-PN for some logics, whose model checking problem is known to be decidable for P/T nets. In this section we define a restriction of $\mathscr{L}(\mathbf{F})$, thus obtaining a fragment that is less expressive than all the logics considered here.

Definition 4 $\mathbf{F}_{cov}$ is the fragment of $\mathscr{L}(\mathbf{F})$ in which negation is not allowed.

In this logic we can express bounded repeated coverability. However, $\mathbf{F}_{cov}$ cannot express properties like $\neg cov(M)$. In particular, the formula $\mathbf{F}(cov(M) \wedge \bigwedge_{p \in P} \neg cov(M_p))$ which expresses reachability, is not a formula of $\mathbf{F}_{cov}$. As for $\mathscr{L}(\mathbf{F})$, we consider existential interpretation, so that a formula is satisfied if some maximal run starting in the initial marking satisfies it. We will see that $\mathbf{F}_{cov}$ model checking is decidable both for reset nets and for ν-PN. Later, we will consider $\forall\mathbf{F}_{cov}$, the version of $\mathbf{F}_{cov}$ with universal interpretation, and we will prove that $\forall\mathbf{F}_{cov}$ model checking is undecidable even for P/T nets.

Proposition 6 *$\mathbf{F}_{cov}$ model checking of reset nets is decidable.*

Proof. Let $N = (P,T,F,R)$ be a reset net and ϕ a formula in $\mathbf{F}_{cov}$. We proceed by induction on the nesting of operators $\mathbf{F}$ in ϕ. If ϕ is a boolean combination of formulae of the form $cov(m)$, it is trivial to decide whether ϕ is satisfied because multiset inclusion is decidable.

Let us suppose that we can check each formula with at most $n > 0$ nested $\mathbf{F}$ operators. Let ϕ be a boolean combination of formulae of the form $cov(m)$ and $\mathbf{F}\varphi$, where φ has at most n nested $\mathbf{F}$ operators, and let us see that we can decide whether each of those formulae $\mathbf{F}\varphi$ is satisfied (and hence, whether ϕ is satisfied). Using standard techniques, we can write $\mathbf{F}\varphi$ as $\mathbf{F}(\bigvee_i((\bigwedge_j cov(m_{ij})) \wedge (\bigwedge_k \mathbf{F}\varphi_{ik})))$, where φ_{ik} are formulae of $\mathbf{F}_{cov}$ with at most $n-1$ nested operators. That formula is equivalent to the formula $\bigvee_i \mathbf{F}((\bigwedge_j cov(m_{ij})) \wedge (\bigwedge_k \mathbf{F}\varphi_{ik}))$, that is, a disjunction of formulae of the form $\mathbf{F}(cov(m_1) \wedge \ldots \wedge cov(m_q) \wedge \mathbf{F}\varphi_1 \wedge \ldots \wedge \mathbf{F}\varphi_r)$, where q and r are not simultaneously zero, and each φ_k has at most $n-1$ nested operators. Let us distinguish the following two cases:
(a) If $q = 0$ the formula is of the form $\mathbf{F}(\mathbf{F}\varphi_1 \wedge \ldots \wedge \mathbf{F}\varphi_r)$, which is equivalent to $\mathbf{F}\varphi_1 \wedge \ldots \wedge \mathbf{F}\varphi_r$. Hence, we can apply the induction hypothesis to each $\mathbf{F}\varphi_k$ and we are done.
(b) If $q > 0$, we modify N, thus obtaining N', by adding transitions $t_1, \ldots, t_q, t_{q+1}$ and places $p_0, p_1, \ldots, p_q$ as follows. We add p_0 as precondition/postcondition of every transition in N. Moreover, t_i moves a token from p_{i-1} to p_i, provided m_i is covered. Finally, t_{q+1} can be fired only one time (for which we add a new place with a single token initially, as precondition of t_{q+1}),

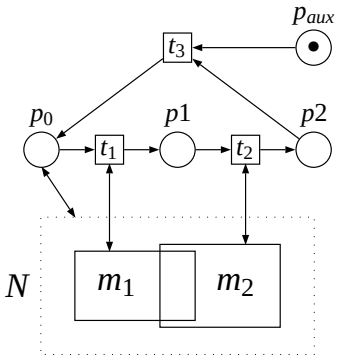

Figure 6: Construction of N' in Prop. 6

setting again a token in p_0, and having p_q as precondition and postcondition. This construction is represented in Fig. 6, for $q = 2$. Then, N' behaves as N, but when every m_i can be covered, it can sequentially fire $t_1, \ldots, t_q, t_{q+1}$. Hence, every m_i can be simultaneously covered in N iff p_q can be covered in N'. We consider the following two sub-cases:
(b.1) If $r = 0$ then the formula is of the form $\mathbf{F}(cov(m_1) \wedge \ldots \wedge cov(m_q))$ and hence equivalent to $\mathbf{F}cov(\{p_q\})$, which expresses a coverability problem, so that it can be decided.
(b.2) Consider now that $r > 0$. For any formula φ, we define φ' as follows:

- If $\varphi = cov(m)$ then $\varphi' = cov(m + \{p_q\})$.
- If $\varphi = \varphi_1 \wedge \varphi_2$ then $\varphi' = \varphi'_1 \wedge \varphi'_2$. Analogously, if $\varphi = \varphi_1 \vee \varphi_2$ then $\varphi' = \varphi'_1 \vee \varphi'_2$.
- If $\varphi = \mathbf{F}\varphi_1$ then $\varphi' = \mathbf{F}\varphi'_1$.

Then, $\mathbf{F}(cov(m_1) \wedge \ldots \wedge cov(m_q) \wedge \mathbf{F}\varphi_1 \wedge \ldots \wedge \mathbf{F}\varphi_r)$ holds in N iff $\mathbf{F}(\mathbf{F}\varphi'_1 \wedge \ldots \wedge \mathbf{F}\varphi'_r)$ holds in N'. Notice that the number of nested $\mathbf{F}$ operators is the same for φ_k and φ'_k. Then, by (a) we are done. □

The proof of the same result for ν-PNs is analogous to the previous one.[5]

Proposition 7 *$\mathbf{F}_{cov}$ model checking is decidable for ν-PN.*

Proof. The construction for ν-PN is the same as the previous one. The only difference is that the names in the markings of the formulae need to be handled correctly in N', by choosing a different variable for each name in a marking to label the arcs. □

Since in particular $\mathbf{F}_{cov}$ allows us to express coverability, which has a non primitive recursive complexity for reset nets [22] and for ν-PN [21], we have the following:

Proposition 8 *The complexity of $\mathbf{F}_{cov}$ model checking is non primitive recursive for reset nets and for ν-PN.*

[5] Actually, the same is true for any model that belongs to the class of Well Structured Transitions Systems, with fairly minor conditions.

To conclude, let us see that the version of $\mathbf{F}_{cov}$ with universal interpretation, that we denote by $\forall\mathbf{F}_{cov}$, is undecidable even for P/T nets. Intuitively, formulae in $\forall\mathbf{F}_{cov}$ are global properties of some trace, or equivalently, eventuality properties of every trace. For instance, $\mathbf{F}cov(M)$ expresses that every run starting from the initial marking eventually covers M.

We reduce control-state reachability for Two Counter Machines, which is undecidable [15]. A *Two Counter Machine* (TCM for short) is a tuple $C = (Q, \{c_1, c_2\}, Ins, q_0)$, where Q is a finite set of control states, c_1 and c_2 are the two counters, Ins is a set of instructions and $q_0 \in Q$ is the initial state. An instruction can be of the following three forms: $Inc(p,i,q)$, $Dec(p,i,q)$ or $Zero(p,i,q)$, where $p,q \in Q$ and $i \in \{1,2\}$, for the increasing of the counter c_i, the decreasing of c_i, or check for zero of c_i respectively. A configuration of C is given by a tuple $\langle q, c_1 = n_1, c_2 = n_2\rangle$, where $q \in Q$ is the current state, and $n_1, n_2 \in \mathbb{N}$ are the current values of the counters. The initial configuration is $\langle q_0, c_1 = 0, c_2 = 0\rangle$.

In a configuration $\langle p, c_1 = n_1, c_2 = n_2\rangle$, we may execute $Inc(p,i,q) \in Ins$, reaching $\langle q, c_1 = n'_1, c_2 = n'_2\rangle$, where $n'_i = n_i + 1$ and $n'_{3-i} = n_{3-i}$. If $n_i > 0$ we may execute $Dec(p,i,q) \in Ins$, reaching $\langle q, c_1 = n'_1, c_2 = n'_2\rangle$, where $n'_i = n_i - 1$ and $n'_{3-i} = n_{3-i}$. Finally, if $n_i = 0$, we can execute $Zero(p,i,q) \in Ins$, reaching $\langle q, c_1 = n_1, c_2 = n_2\rangle$. The control-state reachability problem consists in deciding, given $q \in Q$, whether a configuration of the form $\langle q, c_1 = n_1, c_2 = n_2\rangle$ is reachable. It is well-known that this is an undecidable problem [15].

Example 7 *Consider the TCM $C = (Q, \{c_1, c_2\}, Ins, p)$, with $Q = \{p, q, r\}$ and $Ins = \{Inc(p, c_1, q), Inc(q, c_2, p), Zero(p, c_2, r)\}$. In order to reach a configuration with state r, the first instruction to be executed must be $Zero(p, c_2, r)$. Otherwise, the two first executed instructions are $Inc(p, c_1, q)$ and $Inc(q, c_2, p)$, so we reach a configuration with $c_1 = c_2 = 1$. As there is not a Dec instruction in this machine, after executing this instructions we cannot reach a configuration with $c_2 = 0$ anymore, and therefore we cannot reach state r anymore.*

We consider only deterministic TCM, that is, TCM such that at each reachable configuration there is at most one instruction that can be executed. Moreover, without loss of generality we assume that if $Zero(p,i,q) \in Ins$ then there is no other instruction of the form $Inc(p',j,q)$, $Dec(p',j,q)$ or $Zero(p',j,q)$ in Ins, that is, q can only be reached by that instruction (defined as requirement †). Indeed, for each instruction $I = Zero(p,i,q) \in Ins$ we can add two states q_1, q_2, and replace I by $Zero(p,i,q_1)$, $Inc(q_1,i,q_2)$, $Dec(q_2,i,q)$.[6] The control-state reachability problem for deterministic TCM is still undecidable.

Proposition 9 *$\forall\mathbf{F}_{cov}$ model checking of P/T nets is undecidable.*

Proof. We reduce the control-state reachability problem for deterministic TCM to the model checking problem of a formula in $\forall\mathbf{F}_{cov}$. Let $C = (Q, \{c_1, c_2\}, Ins, q_0)$ be a deterministic TCM and $p_{end} \in Q$. We use the standard simulation of a TCM by means of a P/T net. We define $N = (Q \cup \{c1, c2\}, Ins, F)$, where:

- $F(p, Inc(p,i,q)) = 1$, $F(Inc(p,i,q), q) = 1$, and
  $F(Inc(p,i,q), c_i) = 1$ (a token is moved from p to q, and a token is added to c_i).

[6] If we allow instructions that do not modify the counter then it is enough to add a single state q_1 and an instruction changing the state from q_1 to q.

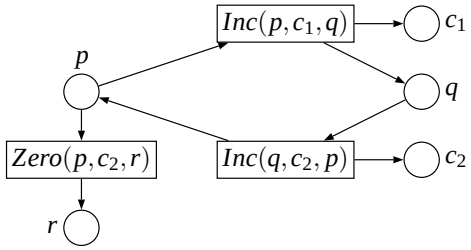

Figure 7: Construction of Prop. 9 for the TCM of Ex. 7

- $F(p, Dec(p,i,q)) = 1$, $F(Dec(p,i,q), q) = 1$, and $F(c_i, Dec(p,i,q)) = 1$ (a token is moved from p to q, and a token is removed from c_i).
- $F(p, Zero(p,i,q)) = 1$ and $F(Zero(p,i,q), q) = 1$ (a token is moved from p to q).

Moreover, $F(n,m) = 0$ elsewhere, and the initial marking of N is $\{q_0\}$. In N, the number of tokens in c_i represent the value of the counter c_i in C. Increasing and decreasing transitions are simulated faithfully. However, the simulation of a transition $Zero(p,i,q)$ can "cheat", whenever it is fired with tokens in c_i. In that case, notice that the marking $\{c_i, q\}$ can be covered. Moreover, because q cannot be reached using a different instruction (requirement (†) above), we know that if such marking is covered then the current simulation has cheated. Focus on Fig. 7, which represents the net built from the TCM of Ex. 7. Note that transition $Zero(p, c_2, r)$ can be fired even after firing $Inc(p, c_1, q)$ and $Inc(q, c_2, p)$, when c_2 is not empty. In this case, this simulation has cheated.

We consider $\varphi = \mathbf{F}(cov(p_{end}) \vee \bigvee_{m \in J} cov(m))$, where $J = \{\{c_i, q\} \mid Zero(p,i,q) \in Ins\}$. Notice that all the cheating runs satisfy φ. We prove that p_{end} can be reached in C if and only if $N \models \varphi$. For the if part, if C reaches p_{end} then the non-cheating run of N eventually covers p_{end}, so that it satisfies φ. Since cheating runs always satisfy φ, every run of N satisfies φ. Conversely, if C does not reach p_{end} then the non-cheating run of N does not satisfy φ. □

5 Conclusions and future work

Table 1 summarizes the results on model checking of P/T nets, reset nets and ν-PNs. In particular, in this work we have proved the undecidability of the fragments LTL_f, $\mathscr{L}(\mathbf{GF})$ and $\mathscr{L}(\mathbf{F})$ for reset nets and ν-PN.

We have defined $\mathbf{F}_{cov}$, a very simple restriction of LTL that does not allow negations, for which model checking of reset nets and ν-PN is decidable. Actually, we claim this is the case for any model in the class of Well Structured Transition Systems [10] under fairly minor conditions, since the model checking problem can be reduced to a finite number of coverability problems. Moreover, we have proved that if we require that every run starting from the initial marking satisfies a formula, then even for the simple case of $\mathbf{F}_{cov}$ and P/T nets, the corresponding model checking problem is undecidable.

Further study, in order to define more expressive logics for which the model checking problem is decidable, is needed. A possible direction in such study could be the definition of logics with atomic propositions that are more specific of the particular model. Such direction links with the so called Yen's logics for P/T nets. In the case of ν-PN, the corresponding logic should be able to express properties about the names in the marking.

Language theory was used to prove the difference of expressiveness between reset nets and ν-PNs in [20]. In this sense, it would certainly be interesting to find a logic which distinguishes between reset nets and ν-PNs.

Finally, we have proved that the complexity of $\mathbf{F}_{cov}$ model checking is non primitive recursive for reset nets and for ν-PN. However, it would be interesting to perform a finer complexity analysis.

Bibliography

[1] M. Ben-Ari, Z. Manna and A. Pnueli. *"The Temporal Logic of Branching Time"*. Acta Informatica 20, 207-226(1983).

[2] R. Bonnet. *"Theory of Well-Structured Transition Systems and Extended Vector-Addition Systems"*. Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France (2013).

[3] A. Bouajjani and R. Mayr. *"Model Checking Lossy Vector Addition Systems"*. International Symposium on Theoretical Aspects of Computer Science, LNCS vol. 1563, 323-333 (1999).

[4] E. M. Clarke, O. Grumberg, and D. A. Peled. *"Model Checking"*. MIT Press Cambridge(1999).

[5] G. Decker, M. Weske. *"Instance Isolation Analysis for Service-Oriented Architectures"*. Proceedings of the 2008 IEEE International Conference on Services Computing, 1, 249-256 (2008).

[6] J. Desel and W. Reisig. *Place/transition petri nets.* Lectures on Petri Nets I: Basic Models, LNCS vol. 1491, pp.122–173. Springer, 1998.

[7] C. Dufourd, A. Finkel, and Ph. Schnoebelen. *"Reset Nets Between Decidability and Undecidability"*. International Colloquium on Automata Languages and Programming, LNCS vol. 1443, 103-115 (1998).

[8] J. Esparza and M. Nielsen. *"Decidability Issues for Petri Nets"*. BRICS Report Series, RS-94-8 (1994).

[9] J. Esparza. *"On the decidability of model checking for several μ-calculi and Petri nets"*. Colloquium on Trees in Algebra and Programming, LNCS vol. 787, 115-129 (1994).

[10] A.Finkel, and P.Schnoebelen. *Well-Structured Transition Systems Everywhere!* Theoretical Computer Science 256(1-2):63-92 (2001).

[11] K. van Hee, A. Serebrenik, N. Sidorova and M. Voorhoeve. *Soundness of Resource-Constrained Workflow Nets.* Applications and Theory of Petri Nets, LNCS vol. 3536, 250-267 (2005)

[12] R. Howell, L. Rosier and H. Yen. *"A taxonomy of fairness and temporal logic problems for Petri nets"*. Theoretical Computer Science 82, 341-372 (1991).

[13] P. Jančar. *"Decidability of a Temporal Logic Problem for Petri Nets."*. Theoretical Computer Science 74, 71-93 (1990).

[14] C. Lakos, S. Christensen. *A General Systematic Approach to Arc Extensions for Coloured Petri Nets.* Applications and Theory of Petri Nets. LNCS vol. 815, pp. 338-357 (1994)

[15] M. L. Minsky. *"Computation: Finite and Infinite Machines"* Prentice-Hall (1967).

[16] OASIS Web Services Business Process Execution Language Version 2.0. OASIS Standard (2007). http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

[17] A. Pnueli. *"The Temporal Semantics of Concurrent Programs"* Theoretical Computer Science, 13, 1-20(1981).

[18] Klaus Reinhardt. *"Reachability in Petri Nets with Inhibitor Arcs"* Electr. Notes Theor. Comput. Sci. 223: 239-264 (2008).

[19] F. Rosa-Velardo and D. de Frutos-Escrig. Name creation vs. replication in Petri Net systems. *Fundamenta Informaticae* 88(3). IOS Press (2008) 329-356.

[20] F. Rosa-Velardo and G. Delzanno. *"Language-based Comparison of Nets with Black Tokens, Pure Names and Ordered Data."*. International Conference on Language and Automata Theory and Applications, LNCS vol. 6031, pp. 524-535. Springer (2010)

[21] F. Rosa-Velardo and D. de Frutos-Escrig. *Decidability and Complexity of Petri Nets with Unordered Data.* Theoretical Computer Science 412(34): 4439-4451 (2011)

[22] Ph. Schnoebelen. *"Revisiting Ackermann-Hardness for Lossy Counter Machines and Reset Petri Nets"*. Theoretical Computer Science, LNCS vol. 6281, pp. 616,628 (2010).