Undecidability of LTL for Timed Petri Nets

Parosh Aziz Abdulla   Aletta Nylén

Department of Information Technology
Uppsala University, Sweden
{parosh, aletta}@it.uu.se

Abstract

We show undecidability of (action based) linear-time temporal logic (LTL) for timed Petri nets. This is to be contrasted with decidability of both the problem of checking safety properties for timed Petri nets, and the problem of checking LTL formulae for (untimed) Petri nets. The undecidability result is shown through a reduction from a similar problem for lossy counter machines [May00]. The reduction is a modification of the reduction used in [RGdFE99] for proving undecidability of reachability for timed Petri nets.

1 Introduction

In this paper we show undecidability of (action based) linear-time temporal logic (LTL) for *Timed Petri Nets (TPNs)*. The model of TPNs is a generalization of standard Petri nets in the sense that each token is equipped with a real-valued clock representing the age of the token. Furthermore each arc in the net is provided with an interval restricting the ages of tokens allowed to travel through the arc.

In fact, we show that it is already undecidable to check a certain fixed property expressible in LTL for TPNs. More precisely, we show undecidability of whether there exists a computation of the TPN along which a given transition is fired infinitely often. The undecidability result is shown through a reduction from a similar problem for *lossy counter machines* [May00]. The reduction is a modification of a reduction of a problem for (non-lossy) 2-counter machines, which was used to prove undecidability of reachability for a model similar to TPNs in [RGdFE99].

Our result should be contrasted with decidability of the following two related problems:

- Checking safety properties, or coverability, for TPNs [AN00, AN01]. More precisely, in [AN01] we show how to characterize the set of markings in a TPN from which a given upward closed set of markings is reachable.

- Checking LTL formulae for standard (untimed) Petri nets [Esp94]. In fact [BM99] shows that there is an effectively constructible representation of the set of markings satisfying a formula.

We recall that (even small fragments) of branching time logics are known to be undecidable for Petri nets [Esp97, May01]. Undecidability holds also for all state based temporal logics [Esp97].

Outline In the next section TPNs and the recurrent place problem for TPNs are introduced. In Section 3 we present LCMs and the recurrent state problem. In Section 4 we prove that LTL is undecidable for TPNs.

2 Timed Petri Nets

Timed Petri Nets (TPNs) are Petri nets where each token is equipped with a real-valued clock representing the age of the token. The firing conditions for transitions include the usual ones for Petri nets. Furthermore, each arc between a place and a transition is labeled with an interval. When a transition is fired, the tokens removed from the input places of the transition and the tokens added to the output places should have ages lying in the intervals of the corresponding arcs.

We let $\mathbb{N}$ and $\mathbb{R}^{\geq 0}$ denote the sets of natural numbers and nonnegative reals respectively. A *multiset* B over a set A is a mapping from A to $\mathbb{N}$ such that for $a \in A$, $B(a)$ is the number of occurrences of a in B. We define A^M to be the set of multisets over A. Sometimes we write multisets as lists, so e.g. $(2.4, 5.1, 5.1, 2.4, 2.4)$ represents a multiset B over $\mathbb{R}^{\geq 0}$ where $B(2.4) = 3$, $B(5.1) = 2$ and $B(x) = 0$ for $x \neq 2.4, 5.1$. We may also write B as $\left(2.4^3, 5.1^2\right)$. For multisets B_1 and B_2 over a set A, we say that $B_1 \leq B_2$ if $B_1(a) \leq B_2(a)$ for each $a \in A$. We define $B_1 + B_2$ to be the multiset B where $B(a) = B_1(a) + B_2(a)$, and (assuming $B_1 \leq B_2$) we define $B_2 - B_1$ to be the multiset B where $B(a) = B_2(a) - B_1(a)$, for each $a \in A$. We use $\emptyset$ to denote the empty multiset, i.e., $\emptyset(a) = 0$ for each $a \in A$.

We use a set *Intrv* of *intervals* of the form $[a : b]$, where $a \in \mathbb{N}$ and $b \in \mathbb{N} \cup \{\infty\}$. For $x \in \mathbb{R}^{\geq 0}$, we write $x \in [a : b]$ to denote that $a \leq x \leq b$.

A *timed Petri net* N is a tuple $(P, T, \mathit{In}, \mathit{Out})$ where P is a finite set of *places*, T is a finite set of *transitions* and $\mathit{In}, \mathit{Out} : T \times P \to \mathit{Intrv}^M$ describes the flow relation. If $\mathit{In}(t,p) \neq \emptyset$ ($\mathit{Out}(t,p) \neq \emptyset$) we say that p is an *input (output) place* of t.

A *marking* of N is a finite multiset over $P \times \mathbb{R}^{\geq 0}$. The marking M defines numbers and ages of the tokens in each place in the net. That is, $M(p, x)$ defines the number of tokens with age x in place p. For example if marking $M = ((p_1, 2.5), (p_1, 1.3), (p_2, 4.7), (p_2, 4.7))$, then there are two tokens in p_1 with ages 2.5 and 1.3 and two tokens each with age 4.7 in the place p_2 in M. For each place p we define $M(p)$ to be the multiset over $\mathbb{R}^{\geq 0}$, where $M(p)(x) = M(p, x)$.

Notice that untimed Petri nets are a special case in our model where all intervals are of the form $[0 : \infty]$.

We define two types of transition relations on markings. A *timed transition* increases the age of all tokens by the same real number. Formally $M_1 \longrightarrow_\delta M_2$ if M_1 is of the form $((p_1, x_1), \ldots, (p_n, x_n))$ and there is $\delta \in \mathbb{R}^{\geq 0}$ such that $M_2 = ((p_1, x_1 + \delta), \ldots, (p_n, x_n + \delta))$.

We define the set of *discrete transitions* $\longrightarrow_D$ as $\bigcup_{t \in T} \longrightarrow_t$, where $\longrightarrow_t$ represents the effect of firing the transition t. More precisely, we define $M_1 \longrightarrow_t M_2$ if, for each place p with $In(t,p) = (\mathcal{I}_1, \ldots, \mathcal{I}_m)$ and $Out(t,p) = (\mathcal{J}_1, \ldots, \mathcal{J}_n)$, there are multisets $B_1 = (x_1, \ldots, x_m)$ and $B_2 = (y_1, \ldots, y_n)$ over $\mathbb{R}^{\geq 0}$, such that the following holds.

- $B_1 \leq M_1(p)$.
- $x_i \in \mathcal{I}_i$, for $i : 1 \leq i \leq m$.
- $y_i \in \mathcal{J}_i$, for $i : 1 \leq i \leq n$.
- $M_2(p) = M_1(p) - B_1 + B_2$.

Intuitively, a transition t may be fired only if for each incoming arc to the transition, there is a token with the right age in the corresponding input place. This token will be removed from the input place when the transition is fired. Furthermore, for each outgoing arc a token with an age in the interval will be added to the output place. We define the relation $\longrightarrow$ to be $\longrightarrow_\delta \cup \longrightarrow_D$.

For a marking M_0 of a TPN N, an M_0*-computation* π of N is of the form $M_0, M_1, M_2, \ldots$, where $M_i \longrightarrow M_{i+1}$, for $i \geq 0$. We say that π *visits a place* p *infinitely often* if there are infinitely many i such that $M_i(p) \neq \emptyset$. The *recurrent place problem* for TPNs (RPP-TPN) is defined as follows.

Instance A TPN N, a marking M of N and a place p of N.

Question Is there an M-computation of N visiting p infinitely often?

In Theorem 4.1, we will show that RPP-TPN is undecidable.

3 Lossy Counter Machines

A *lossy counter machine (LCM)* is a tuple $L = (Q, C, \delta)$, where Q is a finite set of *states*, C is a finite set of counters and δ is a finite set of *transitions*. A transition is a triple of the form $(q_1, instr, q_2)$, where $q_1, q_2 \in Q$ and *instr* is an *instruction*. An instruction is of one of the following three forms

- $c+$ which increases the value of counter c by 1.
- $c-$ which decreases the value of counter c by 1.
- $c?$ which tests whether the value of counter c is equal to 0.

A *configuration* γ of L is of the form (q, Val), where $q \in Q$ and Val is a mapping from the set C of counters to the set $\mathbb{N}$ of natural numbers. We define a transition relation $\longrightarrow$ on the set of configurations such that $(q_1, Val_1) \longrightarrow (q_2, Val_2)$ iff one of the following conditions is satisfied:

1. $(q_1, c+, q_2) \in \delta$, $Val_2(c) = Val_1(c) + 1$ and $Val_2(c') = Val_1(c')$ if $c' \neq c$.
2. $(q_1, c-, q_2) \in \delta$, $Val_1(c) > 0$, $Val_2(c) = Val_1(c) - 1$ and $Val_2(c') = Val_1(c')$ if $c' \neq c$.
3. $(q_1, c?, q_2) \in \delta$, $Val_1(c) = 0$ and $Val_2 = Val_1$.
4. $q_2 = q_1$, $Val_2(c) = Val_1(c) - 1$ for some $c \in C$, and $Val_2(c') = Val_1(c')$ if $c' \neq c$.

For a configuration γ_0, a γ_0-*computation* π of L is of the form $\gamma_0, \gamma_1, \gamma_2, \ldots$, where $\gamma_i \longrightarrow \gamma_{i+1}$, for $i \geq 0$. For a state $q \in Q$, we say that π *visits* q *infinitely often* if there are infinitely many i such that γ_i is of the form (q, Val_i). The *recurrent state problem* for LCMs (RSP-LCM) is defined as follows.

Instance A LCM L, a configuration γ of L and a state q of L.

Question Is there a γ-computation of L visiting q infinitely often?

Theorem 3.1 (Mayr [May00]). *RSP-LCM is undecidable.*

4 Undecidability of LTL

In this section, we show undecidability of linear-time temporal logic (LTL) for TPNs. We do that by showing that RPP-TPN (see Section 2) is undecidable through a reduction from RSP-LCM (Section 3). It is straightforward to show that RPP-TPN is reducible to the problem of checking whether there is a computation of the TPN along which a given transition is fired infinitely often. It follows that (action based) LTL is also undecidable.

Theorem 4.1. *RPP-TPN is undecidable.*

We show this through a reduction from RSP-LCM.

Theorem 4.2. *RSP-LCM can be reduced to RPP-TPN.*

Proof. We say that an instance of RSP-LCM or RPP-TPN is a positive instance if there is a computation such that the state or place is visited infinitely often, that is if the answer to the question in the definition of the problem is yes. Given an instance of RSP-LCM we construct an instance of RPP-TPN such that one of them is a positive instance if and only if the other one is a positive instance. Suppose that we are given an instance of RSP-LCM, i.e., an LCM L, a configuration γ of L and a state q of L. We construct an equivalent instance of RPP-TPN, i.e., we derive a TPN N, a marking M of N, and a place p of

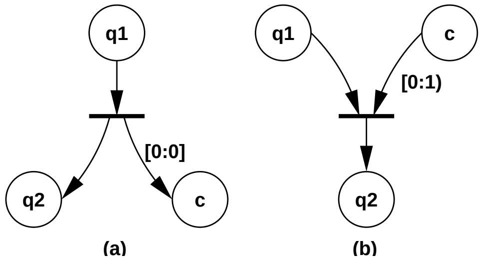

Figure 1: (a) Simulating the operation of increasing the counter. (b) Simulating the operation of decreasing the counter.

N, such that RPP-TPN has a positive answer if and only if RSP-LCM has a positive answer.

Suppose that LCM $L = (Q, C, \delta)$. We construct a corresponding TPN $N = (P, T, In, Out)$ as follows. For each state $q \in Q$ there is a place in P which we call place q. We use P_Q to denote to the set of places of N corresponding to states. Also, for each counter $c \in C$ there is a place in P which we call place c. We use P_C to denote to the set of places corresponding to counters. Intuitively, the state of L is defined in N by the element of P_Q which contains a token. (The TPN N satisfies the invariant that there is at most one place in P_Q which contains a token). The value of counter c in L is defined in N by the number of tokens in place c which have ages less than 1 (tokens which have ages more than 1 are considered to have been lost and do not affect the value of the counter). Losses in L are simulated either by making the age of the token at least 1, or by firing a special *loss* transition which can always remove tokens from the places in P_C. The flow relation corresponding to *In* and *Out* reflects these properties and is defined as follows.

- A transition $(q_1, c+, q_2)$ in δ is simulated by a transition in T which is of the form in Figure 1(a). The transition moves a token from place q_1 to place q_2 and adds a token to place c.

- A transition $(q_1, c-, q_2)$ in δ is simulated by a transition in T which is of the form in Figure 1(b). The transition moves a token from place q_1 to place q_2 and removes a token from place c.

- Transitions of the form $(q_1, c?, q_2)$ have the most complicated simulations. The construction is shown in Figure 2. We use two intermediate places r_1 and r_2. The transition t is first fired adding a token to each of the places r_1 and r_2. The token in r_2 will either stay in r_2 for exactly on time unit, or it will forever stay in place r_2 after which no tokens will ever reside in any place in P_Q. The idea is that we reset the value of counter c to 0, by making the ages of all tokens in place c at least 1. Observe that we simulate testing for 0 in L by resetting the counter in N. This is possible

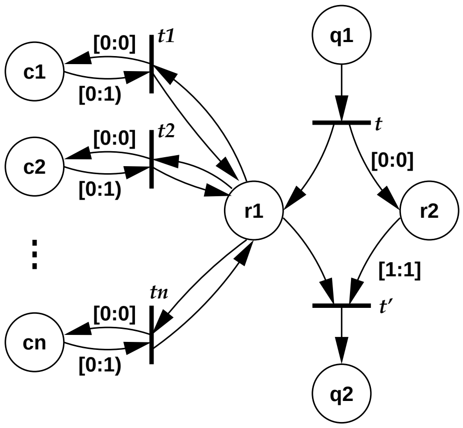

Figure 2: Simulating the operation of testing the value of the counter c.

since L is lossy and therefore it may choose to decrease the counter c to 0 each time c is tested for 0. Furthermore, in order to avoid resetting the values of the rest of the counters, we add, for each $c' \in C - \{c\}$ a new transition. In Figure 2, we assume that $C - \{c\} = \{c_1, \ldots, c_n\}$, and thus we add the transitions $t_1, \ldots, t_n$. The transition is used to refresh the ages of the tokens in the places in $P_C - \{c\}$. For instance, if a token in place c_1 is about to become 1 and thus become too old, the transition t_1 can be fired replacing the token by a new fresh token with age 0. When the transition t' is fired, the new control state will be q_2, and each token in place c will have an age which is at least 1. The resulting marking will therefore correspond to the counter c having the value 0. We also observe that the refreshing process for the rest of the counters will be stopped after firing t', since the token in r_1 will now be removed.

- For each place c in P_C there is a transition which we call $loss_c$ (Figure 3).

Consider a marking M of N and a configuration $\gamma = (q, Val)$ of L. We say that M is an *encoding* of γ if M contains one token in place q and the number of tokens with ages less than 1 in place c is equal to $Val(c)$. Furthermore, all other places in M are empty.

We derive N from L as described above. We define M to be the encoding of γ and define p to be q.

Consider a γ-computation $\pi = \gamma_0, \gamma_1, \gamma_2, \ldots$ of L. We show that there is a M-computation $\pi' = M_0, M_1, M_2, \ldots$ of N, such that for each i there is a $j \geq i$ where M_j is an encoding of γ_i. This implies that if π visits q infinitely often then π' visits p infinitely often. We use induction on i. The base case is trivial. We know that $\gamma_i \longrightarrow \gamma_{i+1}$. This means that we can derive γ_{i+1} from γ_i, using one of the four possible types of transitions described above. We explain only the least obvious case, namely when γ_{i+1} is derived from γ_i by testing the value

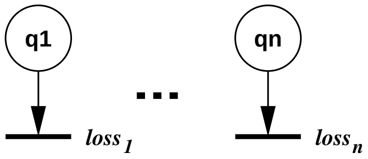

Figure 3: Simulating losses.

of a counter c for 0. The other cases can be explained in a similar manner. Let $\gamma_i = (q_i, Val_i)$. We know that $Val_i(c) = 0$. By induction hypothesis we know that there is a j such that M_j is a encoding of γ_i. This means that place q_i in M_j contains a token. From the construction described above (Figure 2) we know that we can fire a sequence of transitions, which result in moving the token from place q_i to place q_{i+1}, making the ages of all tokens in place c at least 1 and keeping the number of tokens in $P_C - \{c\}$ which have ages less than one. This means that the new marking will be a encoding of γ_{i+1}.

Suppose that there is an M-computation π of N visiting place q infinitely often. Let π be of the form $M_0, M_1, M_2, \ldots$. Consider the maximal subsequence $\pi' = M'_0, M'_1, M'_2, \ldots$ of π, where each M'_i is an encoding of some configuration of L. The sequence π' exists and is infinite since q is visited infinitely often. Let $\gamma_i = (q_i, Val_i)$ be the configuration which is encoded by M'_i. We show that $\gamma_i \xrightarrow{*} \gamma_{i+1}$. It follows immediately that there is a computation π'' of L visiting q infinitely often.

Since $M'_i \xrightarrow{*} M'_{i+1}$ we know that there are $M_0, M_1, \ldots, M_m$ such that $M_0 = M'_i$, $M_m = M'_{i+1}$ and $M_0 \longrightarrow M_1 \longrightarrow \cdots \longrightarrow M_m$. There are two cases. If $m = 1$, i.e., $M'_i \longrightarrow M'_{i+1}$, we know that M'_{i+1} can be derived from M'_i by firing a transition corresponding to one of those in Figure 1 or Figure 3. In this case the proof is straightforward. If $m > 1$, then M'_{i+1} is obtained from M'_i by firing transitions corresponding to those in Figure 2 (these are the only transitions in N which can make all places in P_Q empty and thus prevent the markings $M_1, \ldots, M_{m-1}$ from being encodings of configurations of L). This means that $(q_i, c?, q_{i+1})$ is a transition in L, for some counter c. From the construction of Figure 2, we know that all tokens in place c will eventually have ages which are at least 1. Furthermore, the ages of some of the tokens in $P_C - \{c\}$ may also exceed 1, since not all tokens need to be refreshed. We can derive γ_{i+1} from γ_i by first performing loss transitions corresponding to tokens which become too old followed by the transition $(q_i, c?, q_{i+1})$. $\square$

Acknowledgment

We would like to thank Richard Mayr for helpful discussions.

References

[AN00] Parosh Aziz Abdulla and Aletta Nylén. Better is better than well: On efficient verification of infinite-state systems. In *Proc. LICS' 00 16^{th} IEEE Int. Symp. on Logic in Computer Science*, pages 132–140, 2000.

[AN01] Parosh Aziz Abdulla and Aletta Nylén. Timed Petri nets and BQOs. In *Proc. 22^{th} Int. Conf. on Application and Theory of Petri Nets*, volume 2075 of *Lecture Notes in Computer Science*, pages 53 –70, 2001.

[BM99] A. Bouajjani and R. Mayr. Model checking lossy vector addition systems. In *Symp. on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 323–333, 1999.

[Esp94] J. Esparza. On the decidability of model checking for several mu-calculi and Petri nets. In *CAAP '94*, volume 787 of *Lecture Notes in Computer Science*, pages 115–129. Springer Verlag, 1994.

[Esp97] J. Esparza. Decidability of model checking for infinite-state concurrent systems. *Acta Informatica*, 34:85–107, 1997.

[May00] R. Mayr. Undecidable problems in unreliable computations. In *Theoretical Informatics (LATIN'2000)*, number 1776 in Lecture Notes in Computer Science, 2000.

[May01] R. Mayr. Decidability of model checking with the temporal logic EF. *Theoretical Computer Science*, 256:31–62, 2001.

[RGdFE99] V. Valero Ruiz, F. Cuartero Gomez, and D. de Frutos Escrig. On non-decidability of reachability for timed-arc Petri nets. In *Proceedings of the 8th Int. Workshop on Petri Net and Performance Models (PNPM'99)*, pages 188–196, 1999.