Robustness of Time Petri Nets under architectural constraints

Sundararaman Akshay, Loic Helouet, Claude Jard, Didier Lime, Olivier Henri Roux

Robustness of Time Petri Nets under architectural constraints ⋆

S. Akshay[1,2], Loïc Hélouët[1], Claude Jard[1,2], Didier Lime[3] and Olivier H. Roux[3]

[1] INRIA/IRISA Rennes, France
[2] ENS Cachan Bretagne, Rennes, France
[3] LUNAM Université, École Centrale de Nantes, IRCCyN (CNRS UMR 6597), Nantes, France

Abstract. This paper addresses robustness issues in Time Petri Nets (TPN) under constraints imposed by an external architecture. The main objective is to check whether a timed specification, given as a TPN behaves as expected when subject to additional time and scheduling constraints, specified by another TPN that constrains the specification via read arcs. Our robustness property says that the constrained net does not exhibit new timed or untimed behaviors. We show that this property is not always guaranteed but that checking for it is always decidable in 1-safe TPNs. We further show that checking if the set of untimed behaviors of the constrained and specification nets are the same is also decidable. Next we turn to the more powerful case of labeled 1-safe TPNs with silent transitions. We show that checking for the robustness property is undecidable even when restricted to 1-safe TPNs with injective labeling, and exhibit a sub-class of safe TPNs (with silent transitions) for which robustness is guaranteed by construction. We demonstrate the practical utility of this sub-class with a case-study and prove that it already lies close to the frontiers of intractability.

1 Introduction

Robustness is a key issue for the implementation of systems. Starting from a description of a system, one wants to ensure that the considered system can run as expected on a given architecture with resource constraints (e.g., processors, memory), scheduling schemes on machines implementing several components of the system, imprecision in clocks, possible failures and so on. Once a system is implemented on a given architecture, one may discover that it does not behave as expected: some specified behaviors are never met or unspecified behaviors appear. In this paper, we consider systems in which time and concurrency play an important role.

We address the problem of robust implementability of safe Petri nets. Precisely, we consider a Petri net model of a concurrent system, which is then constrained by another Petri net defining some implementation details (for example, the use of resources). We want to connect these two models in such a way that implementation features can only restrict the set of possible behaviors of the original model, and does not create new behaviors. Thus, if the implementation features can only restrict (but not enlarge) the set of original behaviors, we say the model is robust with respect to the implementation

⋆ This work was funded by the project ANR ImpRo (ANR-2010-BLAN-0317)

constraints. We consider these issues in the setting of Time Petri nets. Time Petri nets (TPNs) are Petri nets which transitions are equipped with timing constraints, given as intervals. As soon as a transition is enabled, a clock attached to this transition is reset and starts measuring time. A transition is then allowed to fire if it is enabled and if its clock's value lays within the time interval of the transition. When a TPN contains read arcs, places that are read can enable/disable a transition, but tokens from read places are not consumed at firing time. Figure 1-a is an example of TPN with read arcs. Transitions are represented as black rectangles, places as circles, flows as thick lines joining transitions ans places, and dotted lines represent read arcs. Transitions can be labeled by an observable letter, or unobservable, and constraints are represented as intervals labeling transitions. Note that in litterature, robustness in timed automata usually refers to invariance of behaviors under small time perturbations. We use the term "robustness" is a more general context: we consider preservation of specified behaviors when new architectural constraints (scheduling policies, resources, ...) are imposed.

We consider bipartite architectures: a specification of a distributed system is given as a TPN, called the *ground net* and the architectural constraints are specified by another TPN, called the *controller*. The controller net can read places of the ground net, but cannot consume tokens from the ground net, and vice versa. The net obtained by considering the ground net in the presence of the controller is called the *controlled net*. Though this problem resembles supervisory control, there are some important differences. Supervisory control is used to restrict the behaviors of a system in order to meet some (safety) property P. The input of the problem is the property P, a description of the system, and the output a controller that restricts system: the behavior of a system under control is a subset of the original specification satisfying P. In our setting, there is no property to ensure, but we want to preserve as much as possible the specified behaviors. We will show in the example below that architectural contraints may add behaviors to the specification. This situation can be particularly harmful, especially when the architecture changes for a system that has been running properly on a former architecture. New faults that were not expected may appear, even when the overall performance of the architecture improves. Detecting such situations is a difficult task that should be automated. The last difference with supervisory control is that we do not ask for synthesis of a controller. In our setting, the controller represents the architectural constraints, and is part of the input of the robustness problem. The question is then whether the ground net preserves its behaviors when controlled.

More specifically, we consider the following questions. We first ask if the untimed language of the controlled net is contained in the untimed language of the ground net. This problem is called *untimed robustness*. Next, we ask if the untimed language is exactly the same despite control, called the *untimed equivalence problem*. The last problem considered is *timed robustness*, which asks if the timed language of the controlled net is contained in the timed language of the ground net.

Let us consider the example of Figure 1-a. It contains a ground net $\mathcal{N}_1$, with four transitions a, a', b, b', and a controller $\mathcal{C}_1$, that acts as a global scheduler allowing firing of a or b. In $\mathcal{N}_1$, transitions a, a' and b, b' are independant. The net $\mathcal{N}_1$ **is not** timed robust w.r.t. the scheduling imposed by $\mathcal{C}_1$: in the controlled net, a can be fired at time 3 which is impossible in $\mathcal{N}_1$ alone. However, if we consider the restriction of $\mathcal{N}_1$ to b, b',

the resulting subnet is timed robust w.r.t $\mathcal{C}_1$. Figure 1-b shows a ground net $\mathcal{N}_2$ with four unobservable transitions, and one observable transition c. This transition can be fired at different dates, depending on wheter the first transition to fire is the left transition (with constraint $[1, 2]$) or the right transition (with constraint $[2, 3]$) below the initially marked place. The net $\mathcal{C}_2$ imposes that left and right transitions are not enabled at the same time, and switches the enabled transition from time to time. With the constraints imposed by $\mathcal{C}_2$, c is firable at date 5 in the controlled net but not at date 6 while it is firable at both dates 5 and 6 in $\mathcal{N}_2$ alone. This example is timed robust w.r.t $\mathcal{C}_2$, as it allows a subset of its original behaviors.

Our results are the following: The problem of checking untimed robustness for 1-safe TPNs is decidable. The timed variant of this problem is decidable for 1-safe TPNs, under the assumption that there are no ϵ transitions and the labeling of the ground net is injective. However, with arbitrary labeling and silent transitions this problem becomes undecidable. Further, even with injective labeling, timed robustness is undecidable as soon as the ground net contains silent transitions. We then show a natural relaxation on the way transitions are controlled and constrained, which ensures timed robustness of nets. In the untimed setting we also consider the stronger notion of equivalence of untimed languages and show that it is always decidable to check this property with or without silent transitions. The paper is organized as follows: Section 2 introduces our model of Petri nets, and the problems considered in the paper. Section 3 shows decidability of robustness in the untimed setting, or when nets are unlabelled. Section 4 shows that this problem becomes undecidable in the timed setting as soon as silent transitions are introduced. Section 5 shows conditions on ground nets and control schemes ensuring timed robustness. Section 6 provides a small case-study to show the relevance of our condition, before concluding with Section 7. Missing proofs can be found in appendix.

Several papers deal with control of Petri Nets where transitions are divided into untimed controllable and uncontrollable transitions. Among them, Holloway and Krogh [8] first proposed an efficient method to solve a control problem for a subclass of Petri Nets called *safe marked graph*. Concerning TPNs, [6] propose a method inspired by the approach of Maler [10]. The controller is synthesized as a feedback function over the state

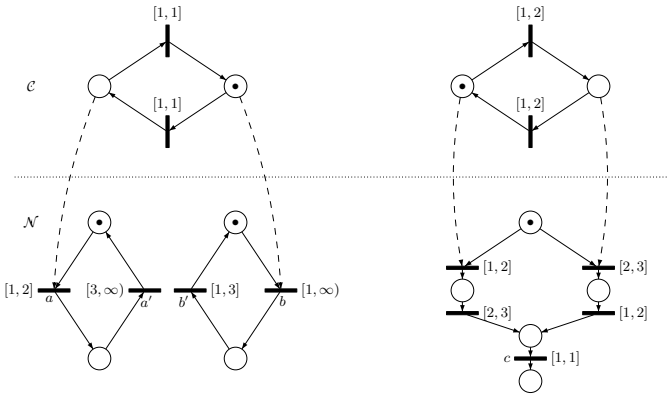

Fig. 1. Illustrative examples (a) and (b) - (unlabeled transitions depict silent moves)

space. However, in all these papers, the controller is given as a feedback law, and it is not possible to design a net model of the controlled system. To overcome this problem, [7] propose a solution using *monitors* to synthesise a Petri Net that models the closed-loop system. The method is extended to real time Supervisory Control in [11]. The supervisor uses enabling arcs (which are equivalent to read arcs) to enable or block a controllable transition. In [13], robustness is addressed in a weaker setting called *schedulability*: given an TPN N, the question is whether the untimed language of N, and the language of the underlying untimed net (i.e. without timing constraints) is the same. This problem is addressed for acyclic nets, or with restricted cyclic behaviors.

2 The model and the questions

Let $\mathbb{Q}^+, \mathbb{R}^+$ denote the set of non-negative rationals and reals respectively. Then, $\mathcal{I}$ denotes the set of time intervals, i.e., intervals in $\mathbb{R}^+$ with end points in $\mathbb{Q}^+ \cup \{+\infty\}$. An interval $I \in \mathcal{I}$ can be open (I^-, I^+), closed $[I^-, I^+]$, semi-open $(I^-, I^+], [I^-, I^+)$ or unbounded $[I^-, +\infty), (I^-, +\infty)$, where I^- and $I^+ \in \mathbb{Q}^+$.

2.1 Time Petri nets

Definition 1 (place/transition net with read arcs). *A time Petri net (TPN for short) with read arcs is a tuple $\mathcal{N} = (P, T, W, R, I)$ where P is a finite set of* places*, T is a finite set of* transitions*, with $P \cap T = \emptyset$, $W : (P \times T) \cup (T \times P) \to \{0, 1\}$ and $R : (P \times T) \to \{0, 1\}$ s.t., $W^{-1}(1) \cap R^{-1}(1) = \emptyset$ are* flow relations *and $I_s : T \to \mathcal{I}$ is a map from the transitions of $\mathcal{N}$ to time intervals $\mathcal{I}$.*

Every TPN can be seen as a union of an untimed Petri Net $N = (P, T, W, R)$ and of a timing function I. The untimed net N will be called the *underlying net* of $\mathcal{N}$.

Semantics. The net defines a bipartite directed graph with two kinds of edges: there exists a (consume) arc from x to y (drawn as a solid line) iff $W(x, y) = 1$ and there exists a (read) arc from x to y (drawn as a dashed line) iff $R(x, y) = 1$. For all $x \in P \cup T$, we define the following sets: ${}^\bullet x = \{y \in P \cup T \mid W(y, x) = 1\}$ and $x^\bullet = \{y \in P \cup T \mid W(x, y) = 1\}$. For all $x \in T$, we define ${}^\circ x = \{y \in P \mid R(y, x) = 1\}$. These definitions extend naturally to subsets by considering union of sets. A *marking* $m : P \to \mathbb{N}$ is a function such that (P, m) is a multiset. For all $p \in P$, $m(p)$ is the number of *tokens* in the place p. A transition $t \in T$ is said *enabled* by the marking m if $m(p) > 0$ for every place $p \in ({}^\bullet t \cup {}^\circ t)$. $\mathsf{en}(N, m)$ denotes the set of transitions of N enabled by m. The firing of an enabled transition t produces a new marking m' computed as $\forall p \in P, m'(p) = m(p) - W(t, p) + W(p, t)$. We fix a marking m^0 of N called its *initial marking*. We say that a transition t' is in conflict with a transition t iff $({}^\bullet t \cup {}^\circ t) \cap ({}^\bullet t') \neq \emptyset$ (firing t' consumes tokens that enable t).

The semantics of a TPN is usually given as a timed transition system (TTS) [9]. This model contains two kinds of transitions: continuous transitions when time passes and discrete transitions when a transition of the net fires. A transition t_k is said *newly enabled* by the firing of the firable transition t_i from the marking m, and denoted

$\uparrow \mathsf{en}(t_k, m, t_i)$, if the transition t_k is enabled by the new marking $(m \setminus {}^\bullet t_i) \cup t_i^\bullet$ but was not by $m \setminus ({}^\bullet t_i)$. We will denote by $\uparrow \mathsf{en}(m, t_i)$ the set of transitions newly enabled by the firing of t_i from m. A valuation is a map $\nu : T \to \mathbb{R}^+$ such that $\forall t \in T, \nu(t)$ is the time elapsed since t was last newly enabled. For $\delta \in \mathbb{R}^+$, $\nu + \delta$ denotes the valuation that associates $\nu(t) + \delta$ to every transition $t \in T$. Note that $\nu(t)$ is meaningful only if t is an enabled transition. $\mathbf{0}$ is the null valuation such that $\forall t, \mathbf{0}(t) = 0$.

The semantics of TPN $\mathcal{N}$ is defined as the TTS $(Q, q_0, \to)$ where a state of Q is a couple (m, ν) of a marking and valuation of $\mathcal{N}$, $q_0 = (m_0, \mathbf{0})$ and $\to \in (Q \times (T \cup \mathbb{R}^+) \times Q)$ is the transition relation describing continuous and discrete transitions. The continuous transition relation is defined $\forall \delta \in \mathbb{R}^+$ by:

$$(m, \nu) \xrightarrow{\delta} (m, \nu') \text{ iff } \nu' = \nu + \delta \text{ and } \forall t_k \in \mathsf{en}(m), \begin{cases} \nu'(t_k) \leq I_s^+(t_k) \text{ and } I_s(t_k) \text{is of the form } [a, b] \text{ or (a,b]} \\ \nu'(t_k) < I_s^+(t_k) \text{ and } I_s(t_k) \text{is of the form [a,b) or (a,b)} \end{cases}$$

Intuitively, time can progress iff letting time elapse does not violate the upper constraint $I_s^+(t)$ of any transition t. The discrete transition relation is defined $\forall t_i \in T$ by:

$$(m, \nu) \xrightarrow{t_i} (m', \nu') \text{ iff } \begin{cases} t_i \in \mathsf{en}(m), m' = (m \setminus {}^\bullet t_i) \cup t_i^\bullet \\ \nu(t_i) \in I_s(t_i), \\ \forall t_k, \nu'(t_k) = 0 \text{ if } \uparrow \mathsf{en}(t_k, m, t_i) \text{ and } \nu(t_k) \text{ otherwise.} \end{cases}$$

Intuitively, transition t_i can fire if it was enabled for a duration included in the time constraint $I_s(t)$. Firing t_i from m resets the clocks of newly enabled transitions.

A *run* of a TTS is a sequence of the form $p_1 \xrightarrow{\alpha_1} p_2 \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_n} p_n$ where $p_1 = q_0$, and for all $i \in \{2..n\}$, $(p_{i-1}, \alpha_i, p_i) \in \to$ and $\alpha_i = t_i \in T$ or $\alpha_i = \delta_i \in \mathbb{R}^+$. Each finite run defines a sequence over $(T \cup \mathbb{R}^+)^*$ from which we can obtain a *timed word over* T of the form $w = (t_1, d_1)(t_2, d_2) \dots (t_n, d_n)$ where each t_i is a transition and $d_i \in \mathbb{R}^+$ the time at which transition t_i is fired. More precisely, if the sequence of labels read by the run are of the form $\delta_0 \delta_1 \dots \delta_{k_1} t_1 \delta_{k_1+1} \delta_{k_1+2} \dots \delta_{k_2} t_2 \dots t_n$, then the timed word obtained is $(t_1, d_1) \dots (t_n, d_n)$ where $d_i = \sum_{0 \leq j \leq k_i} \delta_j$. We define a *dated run* of a TPN $\mathcal{N}$ as the sequence of the form $q_1 \xrightarrow{(d_1, t_1)} q_2 \dots \xrightarrow{(d_n, t_n)} q_n$, where d_i's are the dates as defined above and each q_i is the state reached after firing t_i at date d_i.

We denote by $\mathcal{L}_{tw}(\mathcal{N})$ the timed words over T generated by the above semantics. This will be called the timed (transition) language of $\mathcal{N}$. We denote by $\mathcal{L}_w(\mathcal{N})$ the untimed language of sequences of transitions obtained by projecting onto the first component. Furthermore, given a timed word w over T, if we consider a subset of transitions $X \subseteq T$, we can project w onto X to obtain a timed word over X. We will denote this projected language by $\mathcal{L}_{tw}(\mathcal{N})|_X$. For simplicity, we did not consider final states in our TTS, and hence define prefix-closed languages as is standard in Petri nets. Our results will still continue to hold with an appropriate definition of final states.

In this paper, we limit the study of robustness to TPNs where the underlying PN is *1-safe*, i.e., nets such that $\forall p \in P,\ m(p) \leq 1$, for all reachable markings m in the underlying PN. The reason for using a property of the underlying net is that deciding if an untimed PN is 1-safe is PSPACE-complete, whereas checking if a TPN is bounded is undecidable [12]. Reachability of a marking m in a safe net is also PSPACE-

complete [5]. For safe Petri nets a place contains either 0 or 1 token, hence we identify a marking m with the set of places p such that $m(p) = 1$.

2.2 The Control relation

Let us consider two safe Time Petri nets $\mathcal{N} = (P_\mathcal{N}, T_\mathcal{N}, W_\mathcal{N}, R_\mathcal{N}, I_\mathcal{N}, m^0_\mathcal{N})$ and $\mathcal{C} = (P_\mathcal{C}, T_\mathcal{C}, W_\mathcal{C}, R_\mathcal{C}, I_\mathcal{N}, m^0_\mathcal{C})$. $\mathcal{C}$ models time constraints and resources of an architecture. One can expect these constraints to restrict the behaviors of the original net (we will show however that this is not always the case), that is $\mathcal{C}$ could be seen as a controller. Rather than synchronizing the two nets (as is often done in supervisory control), we define a relation $R \subseteq (P_\mathcal{C} \times T_\mathcal{N}) \cup (P_\mathcal{N} \times T_\mathcal{C})$, connecting some places of $\mathcal{C}$ to some transitions of $\mathcal{N}$ and vice versa. The resulting net $\mathcal{N}^{(\mathcal{C},R)}$ is still a place/transition net defined by $\mathcal{N}^{(\mathcal{C},R)} = (P_\mathcal{N} \cup P_\mathcal{C}, T_\mathcal{N} \cup T_\mathcal{C}, W_\mathcal{N} \cup W_\mathcal{C}, R_\mathcal{N} \cup R_\mathcal{C} \cup R, I_\mathcal{N} \cup I_\mathcal{C}, m^0_\mathcal{N} \cup m^0_\mathcal{C})$. We call $\mathcal{N}$ the *ground net*, $\mathcal{C}$ the *controller net* and $\mathcal{N}^{(\mathcal{C},R)}$ the *controlled net*.

The reason for choosing this relation is two-fold. Firstly, the definition of control above preserves the formalism as the resulting structure is a time Petri net as well. This allows us to deal with a single formalism throughout the paper. Secondly, one can define several types of controllers. By allowing read arcs from the controller to the ground net only, we model blind controllers, whose states evolve independently of the ground net's state. The net in Figure 1(a) is an example of such a controlled net. Conversely, if read arcs are allowed from the ground net to the controller, controller's state changes depending on the current state of the ground net. For the sake of clarity, all examples in the paper have blind controllers, but both types of control are possible.

Our goal is to compare the behaviors of $\mathcal{N}$ with its behaviors when controlled by $\mathcal{C}$ under R, i.e., $\mathcal{N}^{(\mathcal{C},R)}$. Therefore, the language of (timed and untimed) transitions, i.e., $\mathcal{L}_{tw}(\mathcal{N}), \mathcal{L}_{tw}(\mathcal{C}), \mathcal{L}_w(\mathcal{N}), \mathcal{L}_w(\mathcal{C})$, are as usual but when talking about the language of the controlled net, we will always mean the language projected onto transitions of $\mathcal{N}$, i.e., $\mathcal{L}_{tw}(\mathcal{N}^{(\mathcal{C},R)})|_{T_\mathcal{N}}$ or $\mathcal{L}_w(\mathcal{N}^{(\mathcal{C},R)})|_{T_\mathcal{N}}$. Abusing notation, we will write $\mathcal{L}_{tw}(\mathcal{N}^{(\mathcal{C},R)})$ (similarly $\mathcal{L}_w(\mathcal{N}^{(\mathcal{C},R)})$) to denote their projections onto $T_\mathcal{N}$.

2.3 The robustness problem

We will now formally define and motivate the problems that we consider in this paper.

Definition 2. *Given 1-safe TPNs $\mathcal{N}$ and $\mathcal{C}$, and a set of read arcs $R \subseteq (P_\mathcal{C} \times T_\mathcal{N}) \cup (P_\mathcal{N} \times T_\mathcal{C})$, $\mathcal{N}$ is said to be* untimed robust *under $(\mathcal{C}, R)$ if $\mathcal{L}_w(\mathcal{N}^{\mathcal{C},R}) \subseteq \mathcal{L}_w(\mathcal{N})$.*

For time Petri nets, the first problem we consider is the *untimed robustness* problem, which asks whether a given TPN $\mathcal{N}$ is untimed robust under $(\mathcal{C}, R)$. This corresponds to checking whether the controlled net $\mathcal{N}^{(\mathcal{C},R)}$ only exhibits a subset of the (untimed) behaviors of the ground TPN $\mathcal{N}$. The second question addressed is the *untimed equivalence* problem, which asks if the untimed behaviors of the controlled net $\mathcal{N}^{(\mathcal{C},R)}$ and ground net $\mathcal{N}$ are the same, i.e., if $\mathcal{L}_w(\mathcal{N}^{\mathcal{C},R}) = \mathcal{L}_w(\mathcal{N})$. In fact these questions can already be asked for "untimed Petri nets", i.e., for Petri nets without the timing function I_s and we also provide results for this setting.

Note however that untimed robustness only says that every *untimed* behavior of the controlled net $\mathcal{N}^{(\mathcal{C},R)}$ is also exhibited by the ground net $\mathcal{N}$. However some *timed*

behaviors of the controlled net $\mathcal{N}^{(\mathcal{C},R)}$ may *not* be timed behaviors of the ground net $\mathcal{N}$. For obvious safety reasons, one may require that a controlled system does not allow new behaviors, timed or untimed. Thus, we would like to ensure or check that even when considering timed behaviors, the set of timed behaviors exhibited by the controlled net $\mathcal{N}^{(\mathcal{C},R)}$ is a subset of the set of timed behaviors exhibited by the ground net $\mathcal{N}$. We call this the *timed robustness property*.

Definition 3. *Given 1-safe TPNs $\mathcal{N}$ and $\mathcal{C}$, and a set of read arcs $R \subseteq (P_{\mathcal{C}} \times T_{\mathcal{N}}) \cup (P_{\mathcal{N}} \times T_{\mathcal{C}})$, $\mathcal{N}$ is said to be* timed robust *under $(\mathcal{C}, R)$ if $\mathcal{L}_{tw}(\mathcal{N}^{\mathcal{C},R}) \subseteq \mathcal{L}_{tw}(\mathcal{N})$.*

One can further ask if the timed behaviors are exactly the same, which means that the controller is useless. Brought back to our setting, it means that the architectural constraints do not affect the executions of the system, nor their timings. While untimed equivalence of unconstrained and constrained systems seems a reasonable notion, timed equivalence is rarely met, and hence seems a too restrictive requirement. We will see in Section 4 that introducing silent transitions gives a new meaning to these notions.

3 Controlling (time) Petri nets

Let us first consider untimed 1-safe Petri nets. Let N be an untimed net, and C be an untimed controller. We can observe that C can only restrict the behaviors of N, under *any* choice of R. Hence N is always untimed robust under (C, R). Furthermore one can effectively check if the controlled net has the same untimed language as the ground net, by building their marking graphs, and then checking inclusion. Thus, the robustness and equivalence problems are decidable for untimed nets.

Proposition 1. *Let N, C be two* untimed *1-safe Petri nets. Then,*

1. *For any $R \subseteq (P_C \times T_N) \cup (P_N \times T_C)$, N is untimed robust under (C, R).*
2. *For a fixed set of read arcs $R \subseteq (P_C \times T_N) \cup (P_N \times T_C)$ checking if $\mathcal{L}_w(N) = \mathcal{L}_w(N^{(C,R)})$ is PSPACE-complete.*

The proof of this proposition can be found in appendix. Part 1) comes from the fact that a controller only restrics the set of reachable markings. Part 2) comes after demonstration that it is sufficient to show inclusion $\mathcal{L}_w(N) \subseteq \mathcal{L}_w(N^{(C,R)})$, which can be done by exploration of the marking graph of the controlled net.

This property of untimed Petri nets has a counterpart for time Petri nets: let us consider *unconstrained* nets $\mathcal{N}$ and $\mathcal{C}$, i.e., such that $I_{\mathcal{N}}(t) = [0, \infty)$ for every $t \in T_{\mathcal{N}}$, and $I_{\mathcal{C}}(t) = [0, \infty)$ for every $t \in T_{\mathcal{C}}$. Let N and C be the underlying nets of $\mathcal{N}$ and $\mathcal{C}$. One can easily show that for any R, $\mathcal{L}_w(\mathcal{N}^{\mathcal{C},R}) \subseteq \mathcal{L}_w(\mathcal{N})$. As any timed word $w = (a_1, d_1) \dots (a_n, d_n)$ in $\mathcal{L}_{tw}(\mathcal{N}^{\mathcal{C},R})$ (resp. in $\mathcal{L}_{tw}(\mathcal{N})$) is such that $a_1 \dots a_n \in \mathcal{L}_w(N^{C,R})$ (resp. $\mathcal{L}_w(N)$) where each $d_1, \dots d_n$ can be arbitrary dates, we also have $\mathcal{L}_{tw}(\mathcal{N}^{\mathcal{C},R}) \subseteq \mathcal{L}_{tw}(\mathcal{N})$. Thus, unconstrained time Petri nets are also untimed robust.

The question for Time Petri Nets is whether the controlled TPN only restricts the set of behaviors of the original TPN. Unlike in the untimed case, in the timed setting the controlled TPN may exhibit more (in fact, different set of) behaviors than the ground TPN, because of the urgency requirement of TPNs. Consider the example in Figure 2.

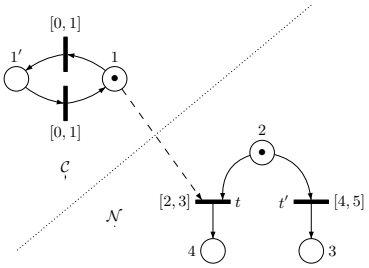

Fig. 2. An example of control of TPN through read-arcs leading to new behaviors

The ground net $\mathcal{N}$ always fires t in the absence of the controller $\mathcal{C}$ but in the presence of $\mathcal{C}$ with R as in the picture, transition t is never fired and t' is always fired. Thus set of (timed and untimed) behaviors of $\mathcal{N}$ and $\mathcal{N}^{(\mathcal{C},R)}$ are disjoint. Discrepancies between untimed languages can be checked using the state class graph construction [4, 9]. This gives the following theorem and its corollary, which proofs are in appendix.

Theorem 1. *For 1-safe TPNs, the untimed robustness problem is PSPACE-complete.*

Corollary 1. *For 1-safe TPNs, the untimed equivalence problem is PSPACE-complete.*

Next we consider timed robustness properties for TPNs. Then, we have

Theorem 2. *For 1-safe TPNs, the timed robustness problem is decidable.*

Proof (sketch). Let $\mathcal{N}$ and $\mathcal{C}$ be 1-safe TPNs, and R be a set of read arcs. We can check if $\mathcal{L}_{tw}(\mathcal{N}^{(\mathcal{C},R)}) \subseteq \mathcal{L}_{tw}(\mathcal{N})$ by using the state class timed automata construction from [9]. It is shown that from the state class graph construction of a 1-safe TPN, $\mathcal{N}$, we can build a deterministic timed automaton $\mathcal{A}$ over the alphabet $T_{\mathcal{N}}$, called the state class timed automaton, such that $\mathcal{L}_{tw}(\mathcal{N}) = \mathcal{L}_{tw}(\mathcal{A})$. As a result, $\mathcal{L}_{tw}(\mathcal{N})$ can be complemented and its complement is accepted by some timed automaton $\mathcal{A}'$, which is computed from $\mathcal{A}$ (see [1] for details of complementation of deterministic timed automata). On the other hand, the state class timed automaton $\mathcal{B}$ constructed from $\mathcal{N}^{(\mathcal{C},R)}$ is over the language $T_{\mathcal{N}} \cup T_{\mathcal{C}}$. By projecting this language onto $T_{\mathcal{N}}$, we obtain the timed (transition) language $\mathcal{L}_{tw}(\mathcal{N}^{(\mathcal{C},R)})$. We remark that the timed automaton corresponding to the projection, denoted $\mathcal{B}'$, can be easily obtained by replacing all transitions of $\mathcal{C}$ in the timed automaton $\mathcal{B}$ by ϵ-transitions [1,3]. Now we just check if $\mathcal{L}_{tw}(\mathcal{B}') \cap \mathcal{L}_{tw}(\mathcal{A}') = \emptyset$, which is decidable in PSPACE [1] (in the sizes of $\mathcal{A}'$ and $\mathcal{B}'$). □

4 Controlling TPNs with silent transitions

We now consider ground nets which may have silent or ϵ-transitions. The (timed and untimed) language of the ground net contains only sequences of observable (i.e., not ϵ) transitions and the robustness question asks if the controller introduces new timed behaviors with respect to this language of observable transitions. From a modeling perspective, robustness means that sequence of important actions remain unchanged with

architectural constraints, and hence this property should hold. Silent transitions can be used to model unimportant or unobservable transitions in the ground net. In this setting, it is natural to require that control does not add to the language of important/observable transitions, while it may allow new changes in other transitions.

An example of such a control is given in the introduction in Figure 1 (b). In that example, the ground net has a unique critical (visible) action c. All other transitions are left unlabeled and so we do not care if the timed or untimed behaviors on those transitions are different in the ground and controlled nets. Then the timed robustness problem asks if c can occur in the controlled net at a date when it was not allowed to occur in the ground net. A more practical example will be studied in detail in Section 6.

With this as motivation, we introduce the class of *ϵ-TPN*, which are TPNs where some transitions may be silent, i.e. labeled by ϵ. The behavior of such nets is deterministic except on silent actions: from a configuration, if a discrete transition that is not labeled ϵ is fired, then the net reaches a unique successor marking.

Definition 4. *Let Σ be a finite set of labels containing a special label ϵ.*

1. *An LTPN over Σ is a structure $(\mathcal{N}, \lambda)$ where $\mathcal{N}$ is a TPN and $\lambda : T_{\mathcal{N}} \to \Sigma$ is the labeling function.*
2. *An ϵ-TPN is an LTPN $(\mathcal{N}, \lambda)$ over Σ such that, for all $t \in T_{\mathcal{N}}$, if $\lambda(t) \neq \epsilon$ then $\lambda(t) \neq \lambda(t')$ for any $t' \neq t \in T_{\mathcal{N}}$.*

For an ϵ-TPN or LTPN $\mathcal{N}$, its *timed* (resp. *untimed*) *language* denoted $\mathcal{L}_{tw}(\mathcal{N}, \lambda)$ (resp. $\mathcal{L}_{w}(\mathcal{N}, \lambda)$) is the set of timed (resp. untimed) words over $\Sigma \setminus \{\epsilon\}$ generated by the timed (resp. untimed) transition system , by ignoring the ϵ labels. A TPN $\mathcal{N}$ from Definition 1 can be seen as the LTPN $(\mathcal{N}, \lambda)$ over Σ such that for all $t \in T_{\mathcal{N}}$, $\lambda(t) = t$, that is, λ is the identity map. An ϵ-TPN can be seen as an LTPN $(\mathcal{N}, \lambda)$ over $\Sigma = T_{\mathcal{N}} \cup \{\epsilon\}$ such that $\lambda(t) = t$ or $\lambda(t) = \epsilon$ for all $t \in T_{\mathcal{N}}$. In [2] it was shown that LTPNs are as powerful, language-wise, as timed automata. As a consequence, we have:

Proposition 2. *[2] The universality problem for timed automata reduces to the universality problem for LTPNs, and hence universality for LTPNs is undecidable.*

We are interested in the problem of *checking timed robustness*, i.e.,

Definition 5. *Given two ϵ-TPNs $(\mathcal{N}, \lambda)$ and $(\mathcal{C}, \lambda')$ over Σ and a set of read arcs R from $(P_{\mathcal{C}} \times T_{\mathcal{N}}) \cup (P_{\mathcal{N}} \times T_{\mathcal{C}})$,*

- *the controlled ϵ-TPN $(\mathcal{N}, \lambda)^{(\mathcal{C},R)}$ is defined as the ϵ-TPN $(\mathcal{N}^{(\mathcal{C},R)}, \lambda'')$ over Σ where $\lambda''(t) = \lambda(t)$ for $t \in T_{\mathcal{N}}$ and $\lambda''(t) = \epsilon$ for $t \in T_{\mathcal{C}}$.*
- *the timed robustness problem asks if $\mathcal{L}_{tw}((\mathcal{N}, \lambda)^{\mathcal{C},R}) \subseteq \mathcal{L}_{tw}(\mathcal{N})$.*

Note that the labels in $\mathcal{C}$ are ignored (i.e., replaced by ϵ), since robustness only compares labels of the ground nets. We remark that untimed robustness and even untimed equivalence are decidable for ϵ-TPNs and LTPNs, since Theorem 1 still holds in the presence of ϵ or labels (indeed, the state class graph built in the proof is an untimed object). We now consider timed robustness and show that this problem is undecidable for ϵ-TPNs and LTPNs.

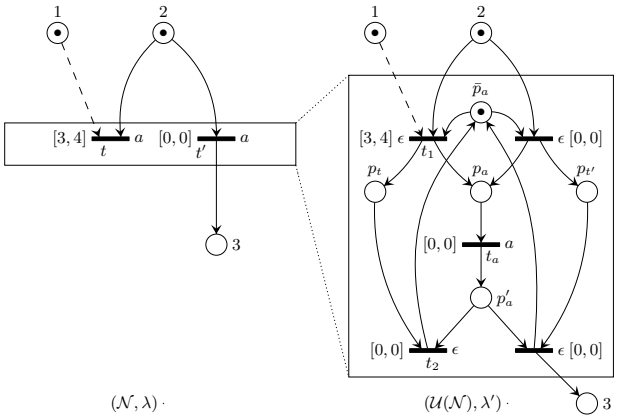

Fig. 3. Construction of a ϵ-TPN equivalent to a LTPN.

Theorem 3. *Checking timed robustness is undecidable for ϵ-TPNs (and LTPNs).*

The proof follows in three steps: First we show that LTPNs can be simulated by ϵ-TPNs. Thus, ϵ-TPNs are expressively as powerful as LTPNs. Then, we show that checking universality of a labeled net can be reduced to checking timed robustness of a related net. Finally, we use Proposition 2 above, which shows that checking universality of labeled nets is undecidable. Let us now prove the first step.

Lemma 1. *Given an LTPN $(\mathcal{N}, \lambda)$ over Σ, there exists a ϵ-TPN $(\mathcal{U}(\mathcal{N}), \lambda')$ over Σ such that $\mathcal{L}_{tw}(\mathcal{U}(\mathcal{N}), \lambda') = \mathcal{L}_{tw}(\mathcal{N}, \lambda)$.*

Proof. The construction is depicted in Figure 3. The idea is to have a unique transition t_a for each letter a which is urgent and will be fired for each transition labeled a in the original net, and use ϵ-transitions (and extra places) to capture the timing constraints on the different transitions (of the original net) labeled by a. Note that the place $\bar{p}_a$ is included in addition to ensure that the resulting net remains 1-safe.

Formally, given a LTPN $(\mathcal{N}, \lambda)$ over Σ, we construct the ϵ-TPN $(\mathcal{U}(\mathcal{N}), \lambda')$ as follows. We split each transition $t \in T_{\mathcal{N}}$ into two transitions t_1 and t_2 and also add a place p_t in $\mathcal{U}(\mathcal{N})$. Further for each action $a \in \Sigma$, such that $\lambda(\widehat{t}) = a$ for some transition $\widehat{t} \in T_{\mathcal{N}}$, we add three places $p_a, p'_a, \bar{p_a}$ and a transition t_a. Then

- we replace every incoming edge into t in $\mathcal{N}$, say (p, t) for some p, by the edge (p, t_1) in $\mathcal{U}(\mathcal{N})$.
- we replace every outgoing edge from t in $\mathcal{N}$, say (t, p') for some p', by the edge (t_2, p') in $\mathcal{U}(\mathcal{N})$.
- in $\mathcal{U}(\mathcal{N})$, we add edges from t_1 to p_t, from t_1 to p_a, from p_a to t_a,
- from t_a to p'_a and from p'_a to t_2. We also add an edge from $\bar{p_a}$ to each transition t_1 and from t_2 to $\bar{p_a}$ such that t is labeled by a. Note that this procedure is applied for each action a and every transition t labeled by a, so as a result, we can obtain a net with several outgoing edges from p'_a or incoming edges to p_a.
- Finally, for the timing constraints, we assign to each t_1 in $\mathcal{U}(\mathcal{N})$ the constraint $I(t)$ assigned to t in $\mathcal{N}$. All other transitions of $\mathcal{U}(\mathcal{N})$ are assigned the constraint $[0, 0]$, hence forcing them to be urgent.

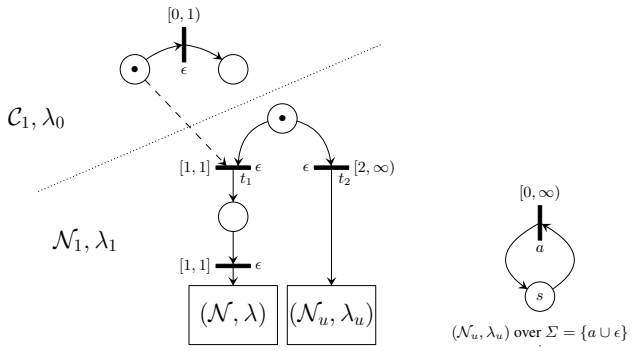

Fig. 4. Reducing checking universality of LTPN to checking robustness of a new ϵ-TPN

Then, λ' is defined by $\lambda'(t_a) = a$ for each t_a, i.e., the transition of $\mathcal{U}(\mathcal{N})$ that was added above for each $a \in \Sigma$, and $\lambda'(\widetilde{t}) = \epsilon$ for all other transitions $\widetilde{t}$ of $\mathcal{U}(\mathcal{N})$. By construction, $(\mathcal{U}(\mathcal{N}), \lambda')$ is uniquely labeled. Each transition t of $\mathcal{N}$ is simulated by a sequence of transitions t_1, t_a, t_2 (place $\bar{p_a}$ ensures atomicity of this sequence). Then we can easily show that $\mathcal{L}_{tw}(\mathcal{U}(\mathcal{N}), \lambda') = \mathcal{L}_{tw}(\mathcal{N}, \lambda)$. □

Next we show a reduction from universality for LTPNs to robustness for ϵ-TPNs.

Lemma 2. *The universality problem for LTPNs can be reduced to checking robustness of ϵ-TPNs.*

Proof. We use a gadget net $(\mathcal{N}_u, \lambda_u)$ which accepts the universal language of timed words over Σ. Such a net is shown in Figure 4 (right). The net depicted in the figure is only over the single discrete alphabet a, but we can by replicating it obtain the universal net over any finite alphabet. Now, as shown in Figure 4 (left), we construct a ground net $(\mathcal{N}_1, \lambda_1)$ which starts with a place and chooses between accepting the timed language of the LTPN $(\mathcal{N}, \lambda)$ and the universal language by using $(\mathcal{N}_u, \lambda_u)$.

Formally, this is defined by adding arcs from the last transition on the left (resp. right) side to the places in the initial marking of $\mathcal{N}$ (resp. $\mathcal{N}_u$). Now by adding disjoint time constraints $[1, 1]$ and $[2, \infty)$ on the transitions, we ensure that $(\mathcal{N}_1, \lambda_1)$ always chooses the left transition t_1 and hence, in the absence of controller, the language accepted is $L_1 = \{(w_1, d_1) \dots (w_n, d_n) \in (\Sigma \times \mathbb{R}^+)^* \mid (w_1, d_1 - 2) \cdots (w_n, d_n - 2) \in \mathcal{L}_{tw}(\mathcal{N}, \lambda)\}$ i.e., the timed language of $(\mathcal{N}, \lambda)$ delayed by 2. In the presence of the controller $(\mathcal{C}_1, \lambda_0)$, only transition t_2 can be fired (as t_1 is disabled by the controller) and hence, the language accepted is $L_2 = \{(w_1, d_1) \dots (w_n, d_n) \in (\Sigma \times \mathbb{R}^+)^* \mid (w_1, d_1 - 2) \cdots (w_n, d_n - 2) \in \mathcal{L}_{tw}(\mathcal{N}_u, \lambda_u)\}$ i.e., the universal language delayed by 2.

Then checking timed robustness corresponds to checking if $L_2 \subseteq L_1$, and checking $L_2 \subseteq L_1$ reduces to checking that $\mathcal{L}_{tw}(\mathcal{N}, \lambda)$ contains the universal language, or equivalently if $\mathcal{L}_{tw}(\mathcal{N}, \lambda)$ is universal, which is undecidable. Note that $(\mathcal{N}_1, \lambda_1)$ is not uniquely labeled since every action a definitely occurs in $(\mathcal{N}_u, \lambda_u)$ and may also occur more than once in $(\mathcal{N}, \lambda)$. Thus the above proof only shows that checking timed robustness for LTPNs is undecidable. But now, using Lemma 1, we can build the ϵ-TPN $(\mathcal{U}(\mathcal{N}_1), \lambda_1')$ over Σ, with the same timed language as $(\mathcal{N}_1, \lambda_1)$. Hence by the above argument checking robustness of ϵ-TPNs is also undecidable. □

Note that checking if $\mathcal{L}_{tw}(\mathcal{N}^{(\mathcal{C},R)}) = \mathcal{L}_{tw}(\mathcal{N})$, i.e., the timed language equivalence is a weaker notion in the context of ϵ-TPNs than in TPN (it only requires preserving timing for important observable actions), and hence could be relevant. For instance in Figure 1(b), we may want to check if c can occur in the controlled net at every date at which it can occur in the ground net (even if the other ϵ-transitions are perturbed). Unfortunately, we easily obtain the undecidability of this problem as an immediate corollary of the above theorem, and even in restricted settings (see Proposition 4 in Appendix).

5 Ensuring robustness in TPNs with silent transitions

The situation for ϵ-TPNs is unsatisfactory since checking timed robustness is undecidable. Hence, we are interested in restrictions that make this problem decidable, or ensuring that this property is met by construction. In this section, we will show that we can restrict the controlling set of read-arcs to ensure that a net is always timed robust. Indeed, it is natural to expect that a "good" controller never introduces new behaviors and we would like to ensure this.

Here, we consider the restriction in which all transitions of the ground nets that have controller places in their preset are not urgent, i.e., the time constraint on the transition is $[\alpha, \infty)$ or (α, ∞) for some $\alpha \in \mathbb{Q}^+$. We call such controlled nets *R-restricted* ϵ-TPNs. In this case we will show that R-restricted ϵ-TPNs are always timed robust (as in the case of untimed PNs shown in Proposition 1). That is,

Theorem 4. *Let $\mathcal{N}$ and $\mathcal{C}$ be two ϵ-TPNs, and R be a set of read arcs such that for every $(p,t) \in R \cap (P_{\mathcal{C}} \times T_{\mathcal{N}})$, $I_s(t)^+ = \infty$, then $\mathcal{L}_{tw}(\mathcal{N}^{(\mathcal{C},R)}) \subseteq \mathcal{L}_{tw}(\mathcal{N})$.*

Proof. We start with some notations. Let $q^{(\mathcal{C},R)}$ be a state of $\mathcal{N}^{(\mathcal{C},R)}$ and $\rho^{(\mathcal{C},R)}$ be a dated run of $\mathcal{N}^{(\mathcal{C},R)}$. We denote by $\check{p}_{\mathcal{N}}(q^{(\mathcal{C},R)})$ the projection of $q^{(\mathcal{C},R)}$ obtained as follows: we keep in the state description, only places of the ground net and clocks associated with uncontrollable transitions of the ground net. Note that the obtained state is described by the same variables as a state of $\mathcal{N}$ but a priori, it is not reachable in the ground net $\mathcal{N}$. Similarly, we denote by $\check{p}_{\mathcal{N}}(\rho^{(\mathcal{C},R)})$ the projection of a dated run of $\mathcal{N}^{(\mathcal{C},R)}$ onto the variables of $\mathcal{N}$ i.e. onto transitions of the ground net and states as defined above. Finally, we denote the last state of the dated run ρ by $last(\rho)$.

We will now prove that for all dated runs $\rho^{(\mathcal{C},R)}$ of $\mathcal{N}^{(\mathcal{C},R)}$, there exists a dated run ρ of $\mathcal{N}$ such that $\rho = \check{p}_{\mathcal{N}}(\rho^{(\mathcal{C},R)})$. The proof is done by induction on the number of transitions in the dated runs. The property obviously holds with no actions (same initial states: $q_0 = \check{p}_{\mathcal{N}}({q_0}^{(\mathcal{C},R)})$). Suppose it holds up to $n \geq 0$ and consider some run ${\rho'}^{(\mathcal{C},R)} = \rho^{(\mathcal{C},R)} \xrightarrow{(d,t)} {q_f}^{(\mathcal{C},R)}$ of $\mathcal{N}^{(\mathcal{C},R)}$ such that $\rho^{(\mathcal{C},R)}$ is of size n.

By the induction hypothesis, there exists ρ of $\mathcal{N}$ such that $\rho = \check{p}_{\mathcal{N}}(\rho^{(\mathcal{C},R)})$. Now consider transition t (occuring at date d): either $t \in T_{\mathcal{C}}$ is a transition of the controller and hence is silent in the controlled net by definition, so we can discard it, and $\check{p}_{\mathcal{N}}({\rho'}^{(\mathcal{C},R)}) = \rho$; or $t \in T_{\mathcal{N}}$ is a transition of the ground net and two cases may arise:

– either t is not controlled, then, two new cases may arise:

- no controlled transitions are in conflict with t in $\mathcal{N}^{(\mathcal{C},R)}$ and since $last(\rho) = \check{p}_{\mathcal{N}}\big(last(\rho^{(\mathcal{C},R)})\big)$, it can occur in the ground net at the same date d;
- a controlled transition t' is in conflict with t in $\mathcal{N}^{(\mathcal{C},R)}$ and the controller blocks t' and allows the firing of t. But then, by definition, we have $I_s(t')^+ = +\infty$. Thus, t' is not urgent in the ground net, i.e., it is always possible to delay it and hence fire t' at a date greater than d in the ground net. As a result t can be fired in the ground net at date d leading to a state $q_f = \check{p}_{\mathcal{N}}({q_f}^{(\mathcal{C},R)})$;

– or it is controlled, and then we have $I_s(t)^+ = +\infty$ and so $I_s(t) = [\alpha, \infty)$ (or (α, ∞), but this is handled similarly so we only consider the closed case) for some $\alpha \in \mathbb{Q}^+$. Then, by induction hypothesis the previous transition of the ground net and the controlled net were fired at the same date d'. Thus
  - if there is no transition in conflict with t in the ground net, then in the controlled net $\mathcal{N}^{(\mathcal{C},R)}$, for the run $last(\rho^{(\mathcal{C},R)}) \xrightarrow{(d,a)} {q_f}^{(\mathcal{C},R)}$, we are guaranteed that $d - d' \geq \alpha$. But in the ground net, t can be fired at any date $d'' \geq d' + \alpha$ (due to $I_s(t)^+ = \infty$) and so it is possible to fire t at date d leading to a state $q_f = \check{p}_{\mathcal{N}}({q_f}^{(\mathcal{C},R)})$ as before.
  - if there are transitions in conflict with t in the ground net, the problematic cases are when they either (i) disable t due to urgency or (ii) force t to be delayed by an arbitrary amount possibly greater than α (for instance, a conflicting transition may empty and refill the preset of t after α time units) in the ground net. But now any delay in firing of t forced on the ground net will also be forced on the controlled net. Thus, if t is either disabled or forced to be delayed beyond d in the ground net, then in the controlled net as well it will be disabled/ forced to delay beyond d which contradicts our assumption t was firable in $\mathcal{N}^{(\mathcal{C},R)}$ at date d. If not, then the delay forced in the controlled net will be (possibly) more than the delay forced in the ground net and hence t is firable at date d in the controlled net implies (due to $I_s(t)^+ = \infty$) that t is firable at date d in the ground net.

Then there exists a run $\rho' = \rho \xrightarrow{(d,a)} q_f$ of $\mathcal{N}$ such that $q_f = \check{p}_{\mathcal{N}}({q_f}^{(\mathcal{C},R)})$ which concludes the induction. □

Note that while timed robustness is ensured for nets and control schemes that fulfill conditions of theorem 4, timed equivalence remains undecidable for such nets. The condition in Theorem 4 is quite restrictive but relaxing it rapidly leads to undecidability:

Proposition 3. *The timed robustness problem is undecidable for ϵ-TPNs with at least one read arc from a place of the controller to any transition t of the ground net such that $I_s(t)^+ \neq \infty$.*

Proof. The proof of Theorem 3 actually gives the result if t is a silent transition. Now, if t is a non-silent transition, then that proof does not work off-the-shelf anymore and we need to modify the construction of Fig. 4. The resulting net is shown in Fig.5.

As before, $(\mathcal{N}, \lambda)$ is any LTPN on some alphabet Σ and $(\mathcal{N}_u, \lambda_u)$ is an ϵ-TPN universal on Σ. Apart from those components, $(\mathcal{N}_1, \lambda_1)$ contains only one non-silent transition ($a \notin \Sigma$). This transition furthermore has a controller place in its preset and its

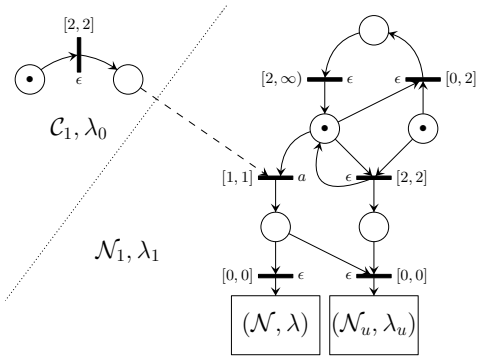

Fig. 5. Reducing universality to rubustness in an ϵ-TPN

time interval has a finite upper bound. So, using Lemma 1, $\mathcal{N}_1$ and $\mathcal{N}_1^{(\mathcal{C},R)}$ can indeed be transformed into ϵ-TPNs satisfying our relaxed condition.

Form the intial configuration, transition a can fire exactly at date 1. The two ϵ transition at the top of the ground net simulate an arbitrary delay greater than two, which can occur only once before firing a. Hence, the timed language of the ground net is the empty word plus the set of all the words of the form $(a, x)w$ with $x = 1$ or $x \geq 3$ and w is either the empty word or any timed word in $\mathcal{L}_{tw}(\mathcal{N}, \lambda)$ delayed by x time units.

Similarly, in the controlled net, a can only fire at a date greater than 2. So, the timed language of the controlled net is the empty word plus the set of all the words of the form $(a, x)w$ with $x \geq 3$ and w is either the empty word or any timed word in $\mathcal{L}_{tw}(\mathcal{N}, \lambda)$ delayed by x time units, or of the form $(a, 3)w'$ where w' is either the empty word or any timed word in $\mathcal{L}_{tw}(\mathcal{N}_u, \lambda_u)$ delayed by 3 time units. Thus, the net is *timed robust* iff $\mathcal{L}_{tw}(\mathcal{N}_u, \lambda_u) \subseteq \mathcal{L}_{tw}(\mathcal{N}, \lambda)$, i.e., $(\mathcal{N}, \lambda)$ is universal. □

6 A small case study

We consider a heater-cooler system depicted in Figure 6. This system improves the hardness of a particular material by first heating and then cooling it. The heater-cooler is equipped with two sensors: $Toohot$ is raised when the heater reaches its maximal temperature. If it occurs, the heating stops automatically. $Cold$ is raised when the temperature is cold enough in the cooling stage. If it occurs, the cooler stops automatically. The heater-cooler starts in the $heating$ state and the operator can push the $StartCooling$ button if the constraints of the system allow it.

We assume architectural constraints imposing that the $StartCooling$ action is not allowed after 20 t.u. in the heating stage, and also disallowed before the date 120 t.u. if the $toohot$ sensor has been raised. The constraints are encoded as a controller $\mathcal{C}$, and read arcs as shown in Figure 6.

We can show that $\mathcal{L}_w(\mathcal{N}^{\mathcal{C},R}) = \mathcal{L}_w(\mathcal{N})$. Hence, $\mathcal{N}$ is untimed robust and even untimed equivalent under $(\mathcal{C}, R)$. The net $\mathcal{N}$ is not an ϵ-TPN, but can be converted to an ϵ-TPN (by Lemma 1). The resulting net is R-restricted, so according to Theorem 4, we have $\mathcal{L}_{tw}(\mathcal{N}^{(\mathcal{C},R)}) \subseteq \mathcal{L}_{tw}(\mathcal{N})$ and then $\mathcal{N}$ is timed robust under $(\mathcal{C}, R)$.

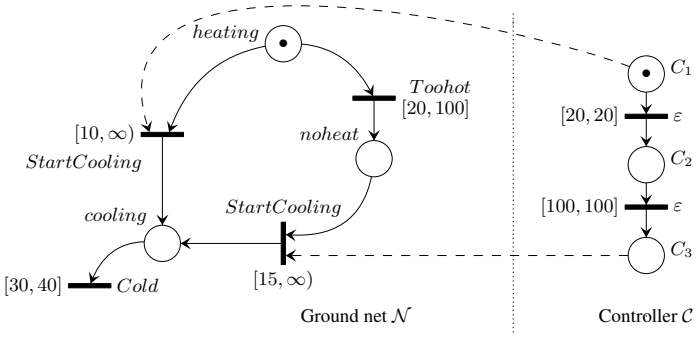

Fig. 6. Case Study

7 Conclusion and discussion

We have defined and studied notions of timed and untimed robustness as well as untimed equivalence for time Petri nets. We are interested in whether we can check or/and guarantee these properties for timed and untimed behaviors. We summarize the results obtained in the table below.

	TPN	R-restricted ϵ-TPN	ϵ-TPN	LTPN
Untimed robustness	Pc (thm 1)	G (thm 4)	Pc	Pc
Untimed equivalence	Pc (cor 1)	Pc	Pc	Pc
Timed robustness	D (thm 2)	G (thm 4)	U (thm 3)	U (thm 3)

U stands for undecidable, D for decidable, Pc for PSPACE-complete, and G for guaranteed.

Overall, with injective labels and no ϵ, robustness is decidable. We think that timed robustness of TPN is EXPSPACE-complete, but this need to be proved. However from a modeling perspective it is important to allow silent transitions. With silent transitions, untimed properties are still tractable, but timed properties become hard to check. To overcome this problem, we proposed a sufficient condition to guarantee timed robustness which we showed is already at the border of undecidability. To show its practical relevance, we designed a small case-study. We also show that untimed equivalence is easily decidable in all the cases. As for timed equivalence, this property is undecidable in most cases. This is not really a surprise nor a limitation, as asking preservation of timed behavior under architectural constraints is a rather strong requirement.

As further discussion, we remark that other criteria can be used for comparing the controlled and ground nets such as (timed) bisimulation or weak bisimulation. While this would be an interesting avenue to explore, a priori, they seem to be more restrictive and hence less viable from a modeling perspective. Possible extensions could be to define tractable subclasses of nets, for instance by considering semantic properties of the net rather that syntactic conditions ensuring decidability. It also seems possible to consider robustness of nets *up to some small delay*. Formally, we can fix a delay as a small positive number δ, and define $\mathcal{L}^{\delta}_{tw}(\mathcal{N}) = \{(w_1,t_1)\dots(w_n,t_n) \mid \exists (w_1,t'_1)\dots(w_n,t'_n) \in \mathcal{L}_{tw}(\mathcal{N}), \forall i \in 1\dots n, |t'_i - t_i| \leq \delta\}$. Then a possible extension of the definitions is to consider δ-robustness under $\mathcal{C}, R$ as the timed inclusion $\mathcal{L}_{tw}(\mathcal{N}^{(\mathcal{C},R)}) \subseteq \mathcal{L}^{\delta}_{tw}(\mathcal{N})$.

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux. Comparison of the expressiveness of timed automata and time Petri nets. In Paul Pettersson and Wang Yi, editors, *3rd International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS 2005)*, volume 3829 of *Lecture Notes in Computer Science*, pages 211–225, Uppsala, Sweden, September 2005. Springer-Verlag.
3. Beatrice Berard, Antoine Petit, Volker Diekert, and Paul Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fundam. Inform.*, 36(2–3):145–182, 1998.
4. Bernard Berthomieu and Michel Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE transactions on software engineering*, 17(3):259–273, March 1991.
5. A. Cheng, J. Esparza, and J. Palsberg. Complexity results for 1-safe nets. *Theoretical Computer Science*, 147(1-2):117–136, 1995.
6. Guillaume Gardey, Olivier (F.) Roux, and Olivier (H.) Roux. Safety control synthesis for time Petri nets. In *8th International Workshop on Discrete Event Systems (WODES'06)*, pages 222–228, Ann Arbor, USA, July 2006. IEEE Computer Society Press.
7. A. Giua, F. DiCesare, and M. Silva. Petri net supervisors for generalized mutual exclusion constraints. In *Proc. 12th IFAC World Congress*, pages 267–270, Sidney, Australia, jul 1993.
8. L. E. Holloway and B. H. Krogh. Synthesis of feedback control logic for a class of controlled Petri nets. *IEEE Trans. on Automatic Control*, 35(5):514–523, may 1990.
9. Didier Lime and Olivier (H.) Roux. Model checking of time Petri nets using the state class timed automaton. *Journal of Discrete Events Dynamic Systems - Theory and Applications (DEDS)*, 16(2):179–205, 2006.
10. Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems. In E.W. Mayr and C. Puech, editors, *Proc. STACS '95*, number 900 in LNCS, pages 229–242. Springer–Verlag, 1995.
11. M. Uzam, A.H. Jones, and I. Yucel. Using a Petri-net-based approach for the real-time supervisory control of an experimental manufacturing system. *Journal of Electrical Engineering and Computer Sciences*, 10(1):85–110, 2002.
12. V. Valero, D. Frutos-Escrig, and F. Cuartero. On non-decidability of reachability for timed-arc Petri nets. In *Proc. 8th International Workshop on Petri Nets and Performance Models (PNPM 99)*, 1999.
13. Dianxiang Xu, Xudong He He, and Yi Deng. Compositional schedulability analysis of real-time systems using time Petri nets. *IEEE Transactions on Software Engineering*, 28(10):984 – 996, 2002.

A Appendix

A.1 Controlling Untimed Petri nets

Here we consider untimed robustness and equivalence in the context of untimed 1-safe Petri nets. Let us denote by N an untimed net, by C an untimed controller, and so on. We can first observe that C may only restrict the behaviors of N, under *any* choice of R, thus the net is always untimed robust. Further, we can effectively check if the controlled net has the same untimed language as the ground net, i.e., the untimed equivalence problem is decidable.

Proposition 1. *Let N, C be two* untimed *1-safe Petri nets. Then,*

1. *For any $R \subseteq (P_C \times T_N)$, N is untimed robust under (C, R).*
2. *For a fixed set of read arcs $R \subseteq (P_C \times T_N)$, checking if N is untimed equivalent to $N^{(C,R)}$ is PSPACE-complete.*

Proof. (1) Observe that the net N is robust under (C, R) iff for all reachable markings $m \subseteq (P_N \cup P_C)$ of $N^{(C,R)}$, $m \setminus P_C$ is a reachable marking of N and $(\mathsf{en}(N^{(C,R)}, m) \setminus T_C) \subseteq \mathsf{en}(N, m \setminus P_C)$. But for a given N, C and $R \subseteq (P_C \times T_N)$, for any $m \subseteq (P_N \cup P_C)$, we have: $(\mathsf{en}(N^{(C,R)}, m) \setminus T_C) \subseteq \mathsf{en}(N, m \setminus P_C)$ since $t \in T_N \wedge t \in \mathsf{en}(N^{(C,R)}, m) \Rightarrow (\forall p \in {}^\circ t \cup t^\bullet, m(p) > 0) \Rightarrow (\forall p \in {}^\circ t \cup t^\bullet, (m \setminus P_C)(p) > 0) \Rightarrow t \in \mathsf{en}(N, m \setminus P_C)$. Now, if m is a reachable marking of $N^{(C,R)}$ then there is a sequence of transitions of $N^{(C,R)}$ leading to m, which means from the above property there is a sequence of transitions of N leading to $m \setminus P_C$. This completes the proof of (1).

(2) First note that since N is always robust under C, R, N is equivalent to $N^{(C,R)}$ if and only if, $\mathcal{L}_w(N) \subseteq \mathcal{L}_w(N^{(C,R)})$. One can check inclusion by exploration of the marking graph of the controlled nets.

We adapt the proof of reachability complexity from [5] to our setting. First, one can notice that N is not equivalent under (C, R) iff one can find a marking m of $N^{(C,R)}$ and a transition t such that t is not fireable from m in $N^{(C,R)}$, t is fireable from $m \setminus P_C$ in N (we will a pair (m, t) of markings and transitions *potential witnesses*), and there is no marking m' reachable from m by using only transitions of C such that t is fireable from m'. A pair (m, t) is a potential witness iff ${}^\bullet(t) \cup {}^\circ(t) \not\subseteq m$, and ${}^\bullet(t) \cup ({}^\circ(t) \cap P_N) \subseteq m$. We can adopt the following strategy. Start from the initial marking m_0, and maintain a boolean rc, and two counters cg and cc, initially set to 0. For each reached marking, do the following. Choose a transition t. If t is fireable from m, then increment cg, and test if cg is greater that $2^{P_C \cup P_N}$. If this is the case, then we have explored a path containing loops, i.e. we have explored twice the same marking and we can stop. If t is not fireable, then check if it is a potential witness. If this is the case, then memorize m set cc to 0, and do an exploration of the marking graph of C starting from $m \cap P_C$ and using controller's transitions only to find a marking m' allowing t. Slightly adapting the result of [5] (which finds a marking and not a set of markings), this can be done in PSPACE, more precisely using $2|P_C|$ bits. If no such marking is found, then our potential witness (m, t) shows that the controlled net is not equivalent. If a marking m' is found, then t is not fireable, and we can consider another transition from m. If m is a deadlock marking, then we have found no witness for non-equivalence and we can stop. So, we have an exploration algorithm that uses $2|P_C \cup P_N| + 2|P_C|$ bits to run, and which is then in PSPACE. For the hardness part, we use again a result from [5], which shows that coverability is PSPACE-Complete. Coverability of a marking m by a net N is satisfied iff one can find a marking $m' \supseteq m$ of N that is reachable from m_0. Let us consider this marking m. We can build from N and m a new net N', by adding a transition t_{fail} that has as preset ${}^\bullet(t_{fail}) = \{p \in P \mid m(p) = 1\}$. Obviously, from any marking that covers m, transition t_{fail} is fireable. Now, we can append a controller C_{fail} with no transition and a single empty place p_{fail} to N, and impose as read arc $r = (p_{fail}, t_{fail}$. In the controlled net $N'^{\{r\}, C_{fail}}$, transition t_{fail} never fires. So, $N'^{\{r\}, C_{fail}}$ is not equivalent if and only if m is coverable by N. As the complement of PSPACE problems is also PSPACE, we have that equivalence in untimed setting is PSPACE-complete. □

Theorem 1. *For 1-safe TPNs, the untimed robustness problem is PSPACE-complete.*

Proof. The proof follows from the state class graph construction [4, 9]. The state class graph $SCG(\mathcal{N})$ is a finite (untimed) transition system describing a regular language. For every safe TPN $\mathcal{N}$, we have $\mathcal{L}_w(SCG(\mathcal{N})) = \mathcal{L}_w(\mathcal{N})$.As result, we obtain that $\mathcal{L}_w(\mathcal{N})$ and $\mathcal{L}_w(\mathcal{N}^{(\mathcal{C},R)})$ are regular. As checking inclusion of two regular languages is decidable (and PSPACE-complete), untimed robustness is also decidable. A state of the state class graph consists of a marking and of a finite set of constraints of the form $x_t - x'_t \leq k$, that can be remembered with a finite number of bits, hence exploring the state class graph of a controlled net can be done in PSPACE, as in the proof of proposition 1 (but with more bits needed to encode states). The hardness reduction from a coverability problem is almost the same. We can append to a place p of net $\mathcal{N}$ two transitions t_1, t_2 to obtain a new net $\mathcal{N}'$. Then we can design a controller $\mathcal{C}$ such that $\mathcal{L}_w(\mathcal{N}') = \{w.t_1 \mid w \in \mathcal{L}_w(\mathcal{N})\}$ and $\mathcal{L}_w(\mathcal{N}^{(\mathcal{C},R)}) = \{w.t_1 \mid w \in \mathcal{L}_w(\mathcal{N})\}$. Hence, $\mathcal{N}'$ is robust iff p is not coverable in $\mathcal{N}$.

Corollary 1. *For 1-safe TPNs, the untimed equivalence problem is PSPACE-complete.*

Proof. Equivalence consists in checking inclusion in both ways, so the proof of theorem 1 can be easily adapted to show decidability and complexity.

A.2 Ensuring robustness for restricted ϵ-TPNs

Theorem 4. *Let $\mathcal{N}$ and $\mathcal{C}$ be two ϵ-TPNs, and R be a set of read arcs such that for every $(p,t) \in R \cap (P_{\mathcal{C}} \times T_{\mathcal{N}})$, $I_s(t)^+ = \infty$, then $\mathcal{L}_{tw}(\mathcal{N}^{(\mathcal{C},R)}) \subseteq \mathcal{L}_{tw}(\mathcal{N})$.*

Proof. We start with some notations. Let $q^{(\mathcal{C},R)}$ be a state of $\mathcal{N}^{(\mathcal{C},R)}$ and $\rho^{(\mathcal{C},R)}$ be a dated run of $\mathcal{N}^{(\mathcal{C},R)}$. We denote by $\check{p}_{\mathcal{N}}(q^{(\mathcal{C},R)})$ the projection of $q^{(\mathcal{C},R)}$ obtained as follows: we keep in the state description, only places of the ground net and clocks associated with uncontrollable transitions of the ground net. Note that the obtained state is described by the same variables as a state of $\mathcal{N}$ but a priori, it is not reachable in the ground net $\mathcal{N}$. Similarly, we denote by $\check{p}_{\mathcal{N}}(\rho^{(\mathcal{C},R)})$ the projection of a dated run of $\mathcal{N}^{(\mathcal{C},R)}$ onto the variables of $\mathcal{N}$ i.e. onto transitions of the ground net and states as defined above. Finally, we denote the last state of the dated run ρ by $last(\rho)$.

We will now prove that for all dated runs $\rho^{(\mathcal{C},R)}$ of $\mathcal{N}^{(\mathcal{C},R)}$, there exists a dated run ρ of $\mathcal{N}$ such that $\rho = \check{p}_{\mathcal{N}}(\rho^{(\mathcal{C},R)})$. The proof is done by induction on the number of transitions in the dated runs. The property obviously holds with no actions (same initial states: $q_0 = \check{p}_{\mathcal{N}}({q_0}^{(\mathcal{C},R)})$). Suppose it holds up to $n \geq 0$ and consider some run ${\rho'}^{(\mathcal{C},R)} = \rho^{(\mathcal{C},R)} \xrightarrow{(d,t)} {q_f}^{(\mathcal{C},R)}$ of $\mathcal{N}^{(\mathcal{C},R)}$ such that $\rho^{(\mathcal{C},R)}$ is of size n.

By the induction hypothesis, there exists ρ of $\mathcal{N}$ such that $\rho = \check{p}_{\mathcal{N}}(\rho^{(\mathcal{C},R)})$. Now consider transition t (occuring at date d): either $t \in T_{\mathcal{C}}$ is a transition of the controller and hence is silent in the controlled net by definition, so we can discard it, and $\check{p}_{\mathcal{N}}({\rho'}^{(\mathcal{C},R)}) = \rho$; or $t \in T_{\mathcal{N}}$ is a transition of the ground net and two cases may arise:

– either t is not controlled, then, two new cases may arise:

- no controlled transitions are in conflict with t in $\mathcal{N}^{(\mathcal{C},R)}$ and since $last(\rho) = \check{p}_{\mathcal{N}}\big(last(\rho^{(\mathcal{C},R)})\big)$, it can occur in the ground net at the same date d;
- a controlled transition t' is in conflict with t in $\mathcal{N}^{(\mathcal{C},R)}$ and the controller blocks t' and allows the firing of t. But then, by definition, we have $I_s(t')^+ = +\infty$. Thus, t' is not urgent in the ground net, i.e., it is always possible to delay it and hence fire t' at a date greater than d in the ground net. As a result t can be fired in the ground net at date d leading to a state $q_f = \check{p}_{\mathcal{N}}({q_f}^{(\mathcal{C},R)})$;

– or it is controlled, and then we have $I_s(t)^+ = +\infty$ and so $I_s(t) = [\alpha, \infty)$ (or (α, ∞), but this is handled similarly so we only consider the closed case) for some $\alpha \in \mathbb{Q}^+$. Then, by induction hypothesis the previous transition of the ground net and the controlled net were fired at the same date d'. Thus
  - if there is no transition in conflict with t in the ground net, then in the controlled net $\mathcal{N}^{(\mathcal{C},R)}$, for the run $last(\rho^{(\mathcal{C},R)}) \xrightarrow{(d,a)} {q_f}^{(\mathcal{C},R)}$, we are guaranteed that $d - d' \geq \alpha$. But in the ground net, t can be fired at any date $d'' \geq d' + \alpha$ (due to $I_s(t)^+ = \infty$) and so it is possible to fire t at date d leading to a state $q_f = \check{p}_{\mathcal{N}}({q_f}^{(\mathcal{C},R)})$ as before.
  - if there are transitions in conflict with t in the ground net, the problematic cases are when they either (i) disable t due to urgency or (ii) force t to be delayed by an arbitrary amount possibly greater than α (for instance, a conflicting transition may empty and refill the preset of t after α time units) in the ground net. But now any delay in firing of t forced on the ground net will also be forced on the controlled net. Thus, if t is either disabled or forced to be delayed beyond d in the ground net, then in the controlled net as well it will be disabled/ forced to delay beyond d which contradicts our assumption t was firable in $\mathcal{N}^{(\mathcal{C},R)}$ at date d. If not, then the delay forced in the controlled net will be (possibly) more than the delay forced in the ground net and hence t is firable at date d in the controlled net implies (due to $I_s(t)^+ = \infty$) that t is firable at date d in the ground net.

Then there exists a run $\rho' = \rho \xrightarrow{(d,a)} q_f$ of $\mathcal{N}$ such that $q_f = \check{p}_{\mathcal{N}}({q_f}^{(\mathcal{C},R)})$ which concludes the induction. □

Note that while timed robustness of a net is ensured for nets and control schemes that fulfill conditions of theorem 4, timed equivalence remains undecidable for such nets.

Proposition 4. *Let $\mathcal{N}$ and $\mathcal{C}$ be two ϵ-TPNs, and R be a set of read arcs such that for every $(p,t) \in R \cap (P_{\mathcal{C}} \times T_{\mathcal{N}})$, $I_s(t)^+ = \infty$, then checking whether $\mathcal{L}_{tw}(\mathcal{N}^{(\mathcal{C},R)}) = \mathcal{L}_{tw}(\mathcal{N})$ is undecidable.*

Proof. Consider the net of Figure 3, and replace every time constraint in the ground net by $[0, \infty)$. Exchange $(\mathcal{N}, \lambda)$ and $(\mathcal{N}_u, \lambda_u)$, and remove the token in the initial marking of the controller (that is, the transition with a controller's place in its preset will never fire in the controlled net). Call this new net $\mathcal{N}_2$. Note that $\mathcal{N}_2$ fulfills the conditions of theorem 4. Then the language of $\mathcal{N}_2$ is $\mathcal{L}_{tw}(\mathcal{N}_2) = \mathcal{L}_{tw}(\mathcal{N}_u) \cup \mathcal{L}_{tw}(\mathcal{N})$, that is, it is the universal language. The language of the controlled net $\mathcal{N}_2^{C,R}$ is $\mathcal{L}_{tw}(\mathcal{N}_2^{C,R}) =$

$\{(w_1, d_1) \dots (w_n, d_n) \mid \exists (w_1, d'_1) \dots (w_n, d'_n) \in \mathcal{L}_{tw}(\mathcal{N}), \forall i \in 1..n d'_i \leq d_i + 2\}$. Comparing $\mathcal{L}_{tw}(\mathcal{N}_2)$ and $\mathcal{L}_{tw}(\mathcal{N}_2^{C,R})$ then amounts to checking that $\mathcal{L}_{tw}(\mathcal{N})$ is the universal language. □

Note that undecidability of timed equivalence holds event if controlled transitions have $[0, \infty)$ constraints, or if controlled transitions are not ϵ transitions.