Combining Free choice and Time in Petri Nets

S. Akshay[a], Loïc Hélouët[b], Ramchandra Phawade[a]

[a] *Department of CSE, IIT Bombay*
{akshayss,ramchandra}@cse.iitb.ac.in
[b] *INRIA Rennes*
loic.helouet@inria.fr

Abstract

Time Petri nets (TPNs) are a classical extension of Petri nets with timing constraints attached to transitions, for which most verification problems are undecidable. We consider TPNs under a strong semantics with multiple enablings of transitions. We focus on a structural subclass of unbounded TPNs, where the underlying untimed net is free choice, and show that it enjoys nice properties in the timed setting under a multi-server semantics. In particular, we show that the questions of firability (whether a chosen transition can fire), and termination (whether the net has a non-terminating run) are decidable for this class. Next, we consider the problem of robustness under guard enlargement and guard shrinking, i.e., whether a given property is preserved even if the system is implemented on an architecture with imprecise time measurement. For unbounded free choice TPNs with a multi-server semantics, we show decidability of robustness of firability and of termination under both guard enlargement and shrinking.

Keywords: Petri nets, Time, Free-choice, Robustness

---

1. Introduction

Modern systems are composed of several distributed components that work in real-time to satisfy a given specification. This makes them difficult to reason about manually and encourages the use of formal methods to analyze them automatically. This in turn requires the development of models that capture all the features of a system and still allow efficient algorithms for analysis. Further, to bring formal models closer to real world implementations, it is important to design robust models, i.e., models that preserve their behavior or at least some important properties under imprecise time measurement.

In this paper, we consider Petri nets extended with time constraints. These models have been used for modeling real-time distributed systems, but for timed variants of Petri nets, many basic problems are usually undecidable or algorithmically intractable. Our goal is to consider structural restrictions which allow us to model features such as unbounded resources as well as time-deadlines while remaining within the realm of decidability and satisfying some robustness properties.

Time Petri nets (TPNs) [1] are a classical extension of Petri nets in which time intervals are attached to transitions and constrain the time that can elapse between the enabling of a transition and its firing date. In such models, the basic verification problems considered include: **reachability**, i.e., whether a particular marking (or a configuration) can be reached in the net; **termination**, i.e., whether there exists an infinite run in the net; **boundedness**, whether there is a bound on the number of tokens in the reachable markings; and **firability**, i.e., whether a given transition is firable in some execution of the net.

It turns out that all these basic problems are in general undecidable [2] for TPNs, though they are decidable for the untimed version of Petri nets. The main reason is that TPNs are usually equipped with an urgent semantics: when the time elapsed since enabling of a transition reaches the maximal value of its interval, a transition of the net *has to fire*. This semantics breaks monotony (the behaviors allowed from a marking, and from a larger marking can be completely different). Indeed, with a TPN, one can easily encode a two-counter machine, yielding undecidability of most of the verification problems (see [2, 3] for such an encoding). Decidability can be obtained by restricting to the *subclass* of *bounded TPNs*, for which the number of tokens in all places of accessible markings is bounded by some constant. Another way to obtain decidability is to weaken the semantics [3], or restrict the use of urgency [4].

Another important problem in this setting, is the question of **robustness**. Robustness can be defined as the preservation of properties of systems that are subject to imprecision of time measurement. The main motivation for considering this is that formal models usually have an idealized representation of time, and assume an unrealizable precision in measurement of time, that cannot be guaranteed by real implementations. Robustness has been studied extensively for timed automata since [5], and more recently for TPNs [6], but decidability results are only obtained in a bounded-resource setting.

The definition of the semantics plays an important role both to define the expressive power of a model, and to obtaining decidability results. When considering unbounded nets, where multiple (and a possibly unbounded number of) tokens may be present at every place, one has to decide whether transitions should be considered as multiply-enabled, and if so, fix a policy to handle multiple instances of enabling. And this becomes even more complicated when real-time constraints are considered. Indeed, several possible variants for the multiple enabling semantics have been considered, as discussed in [7]. In this paper, we fix one of the variants and consider TPNs equipped with a multi-enabling urgent semantics, which allows to start measuring elapsed time from every occurrence of a transition enabling. This feature is particularly interesting: combined with urgency, it allows for instance to model maximal latency in communication channels. We adopt a semantics where time is measured at each transition's enabling, and with urgency, i.e. a discrete transition firing *has to occur* if a transition has been enabled for a duration that equals the upper bound in the time interval attached to it. Obviously, with this semantics, counter machines can still be encoded, and undecidability follows in general.

We focus on a structural restriction on TPNs, which restricts the underlying net of the given TPN to be free-choice, and call such nets *Free-choice TPNs*. Free-choice Petri nets have been extensively studied in the untimed setting [8] and have several nice properties from a decidability and a complexity-theoretic point of view. In this class of nets, all occurrences of transitions that have a common place in their preset are enabled at the same instant. Such transitions are said to belong to a *cluster* of transitions. Thus, with this restriction, a transition can only prevent transitions from the same cluster to fire, and hence only constrain firing times of transitions in its cluster. Further, we disallow forcing of instantaneous occurrence of infinite behaviors, that we call (forced) 0-delay firing sequences. This can easily be ensured by another structural restriction forbidding transitions or even loops in TPNs labeled with $[0, 0]$ constraints.

Our main results are the following: we show that for a free-choice TPN $\mathcal{N}$ under the multiple-enabling urgent semantics, and in the absence of 0-delay firing sequences, the problem of fireability of a transition and of termination are both decidable. The main idea is to show that, after some pre-processing, we can reduce these problems to corresponding problems on the underlying untimed free-choice PN. More precisely, we are able to show that every partially-ordered execution of the underlying untimed PN can be simulated by $\mathcal{N}$, i.e., it is the untimed prefix of a timed execution of $\mathcal{N}$. To formalize this argument we introduce definitions of (untimed and timed) causal processes for unbounded TPNs, which is another contribution of the paper.

Finally, we address several robustness questions. The problem of robustness for TPNs has previously been considered in [6], but showed decidability only for bounded classes of nets. We show that the problem of robustness of firability with respect to guard enlargement, i.e., whether there exists a $\Delta > 0$ such that enlarging all guards of a TPN by Δ preserves the set of fireable transitions, is decidable for free-choice TPNs without 0-delay firing sequences. We also consider the same question for guard shrinking, i.e. existence of a $\triangledown > 0$ such that shortening the guards of a TPN by $\triangledown$ preserves the set of fireable transitions. We show that this problem is also decidable. Finally, we consider robustness of termination (whether there exists an infinite run) w.r.t. guard enlargement or shrinking, and show that this question is also decidable. Up to our knowledge, this is the first decidability result on robustness for a class of unbounded TPNs.

Related work. Verification, unfolding, and extensions of Petri nets with time have been considered in many works. Another way to integrate time to Petri nets is to attach time to tokens, constraints to arcs, and allow firing of a transition iff all constraints attached to transitions are satisfied by at least one token in each place of its preset. This variant, called Timed-arc Petri nets, enjoys decidability of coverability [9], but cannot impose urgent firing, which is a key issue in real-time systems. In TPNs, [3] propose a weak semantics for TPNs, where clocks may continue to evolve even if a transition does not fire urgently. With this semantics, TPNs have the same expressive power as untimed Petri nets, again due to lack of urgency, which is not the case in our model.

Recently, [4] has considered variants of time and timed-arc Petri nets with urgency (TPNUs), where decidability of reachability and coverability is obtained by restricting urgency to transitions consuming tokens only from bounded places. This way, encoding of counter machines is not straightforward, and some problems that are undecidable in general for time or timed-arc Petri nets become decidable. The free-choice assumption in this paper is orthogonal to this approach and it would be interesting to see how it affects decidability for TPNUs.

Partial-order semantics have been considered in the timed setting: [10] defines a notion of timed process for timed-arc Petri nets and [11] gives a semantics to timed-arc nets with an algebra of concatenable weighted pomsets. However, processes and unfoldings for TPNs have received less attention. An initial proposal in [12] was used in [13] to define equivalences among time Petri nets. Unfolding and processes were refined by [14] to obtain symbolic unfoldings for *safe* Petri nets. The closest work to ours is [15], where processes are defined to reason about the class of free choice *safe* TPNs. However, this work does not consider unbounded nets, and focuses more on semantic variants w.r.t. time-progress than on decidability or robustness issues.

This paper is an extended version of a former work [16]. With respect to this first contribution, it contains all the proofs in full details, and an extended comparison between the single and multiple-server semantics of TPNs. It also establishes new results on robustness of firability and termination, that are now considered with respect to guard enlargement and shrinking. The paper is organized as follows: Section 2 introduces notations and defines a class of TPNs with multi-enabling semantics. Section 3 defines processes for these nets. Section 4 introduces the subclass of Free-choice TPNs and relates properties of untimed and timed processes. In Section 5, this relation is used to prove decidability of fireability and termination for FC-TPNs and, in Section 6 to address robustness of fireability and termination to guard enlargement and shrinking. Section 7 discusses the assumptions needed to obtain decidability, and issues related to decidability of other problems in FC-nets, before conclusion.

2. Multi-enabledness in TPNs

Let Σ be a finite alphabet, Σ^* be the set of finite words over Σ. For a pair of words $w, w' \in \Sigma^*$, we will write $w \leq w'$ iff $w' = w.v$ for some $v \in \Sigma^*$. Let $\mathbb{N}, \mathbb{Q}, \mathbb{R}_{\geq 0}$, respectively, denote the sets of naturals, rationals, and non-negative real numbers. An interval I of $\mathbb{R}_{\geq 0}$ is a $\mathbb{Q}_{\geq 0}$-interval iff its left endpoint belongs to $\mathbb{Q}_{\geq 0}$ and right endpoint belongs to $\mathbb{Q}_{\geq 0} \cup \{\infty\}$. An interval is *closed* if it of the form $[\alpha, \beta]$, and *open* otherwise. Let $\mathcal{I}$ denote the set of $\mathbb{Q}_{\geq 0}$-intervals of $\mathbb{R}_{\geq 0}$. For a set X of (clock) variables, a valuation v for X is a mapping $v : X \to \mathbb{R}_{\geq 0}$. Let $v_0(X)$ be the valuation which assigns value 0 to each clock in X. For any $d \in \mathbb{R}_{\geq 0}$, the valuation $v + d$ is : $\forall x \in X,\ (v + d)(x) = v(x) + d$.

A Petri net (PN) $\mathcal{U}$ is a tuple $\mathcal{U} = (P, T, F)$, where P is a set of places, $T = \{t_1, t_2, \dots, t_K\}$ is a set of transitions, and $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation. A marking M is a function $P \to \mathbb{N}$ that assigns a number of tokens to each place. We let M_0 denote an initial marking for a net. For any $x \in P \cup T$ (called a *node*) let ${}^\bullet x = \{y \in P \cup T \mid (y, x) \text{ in } F\}$ and $x^\bullet = \{y \in P \cup T \mid (x, y) \text{ in } F\}$. For a transition t, we will call ${}^\bullet t$ the preset of t, and $t^\bullet$ the *postset* of t. The semantics of a Petri net is defined as usual: starting from a marking M, a transition t can be fired if for every $p \in {}^\bullet t, M(p) > 0$. This firing results in new marking M' such that $M'(p) = M(p) - |{}^\bullet t \cap \{p\}| + |t^\bullet \cap \{p\}|$. We denote a discrete move of a Petri net from M to M' using transition t by $M \xrightarrow{t} M'$, and we write $M \to M'$ when there exists a discrete move from M to M'. The language $\mathit{Lang}(\mathcal{U}, M_0)$ of a Petri net $\mathcal{U}$ is the set of words of the form $t_1 \dots t_n$ such that $M_0 \xrightarrow{t_1} M_1 \dots \xrightarrow{t_n} M_n$. $\mathit{Reach}(\mathcal{U}, M_0)$ denotes the set of reachable markings that can be obtained via an arbitrary number of moves starting from M_0. We say that a Petri net is safe if, for every marking $M \in \mathit{Reach}(\mathcal{U}, M_0)$, for every place $p \in P$, $M(p) \leq 1$. Let x be any node of a net. The *cluster* of x (denoted by $[x]$) is a minimal set of nodes of $P \cup T$ satisfying following conditions:

i) $x \in [x]$,

ii) if a place p is in $[x]$, then $p^\bullet \subseteq [x]$, and

iii) if a transition t is in $[x]$, then ${}^\bullet t \subseteq [x]$.

Intuitively, two transitions belong to the same *cluster* if they share a place in their preset, and two places belong to the same cluster if they share a transition in their postset. The notion of cluster is interesting, as it defines transitions that may compete for the same resources. The notion of cluster of a element and the flow relation of a Petri net $\mathcal{U}$ allows for the definition of a partition of elements from $P \cup T$. The subsets in this partition are called the *clusters* of $\mathcal{U}$.

2.1. Time Petri nets

Definition 1. *A time Petri net (TPN) $\mathcal{N}$ is a tuple $(\mathcal{U}, M_0, I)$ where $\mathcal{U} = (P, T, F)$ is the underlying net, M_0 is an initial marking, and $I : T \to \mathcal{I}(\mathbb{Q}_{\geq 0})$ associates with each transition a firing interval.*

We denote by $eft(t)$ and $lft(t)$ the lower and upper bound of interval $I(t)$. For a TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$ let $Untime(\mathcal{N}) = (\mathcal{U}, M_0)$ denote the underlying net i.e., with no timing constraints for any transitions.

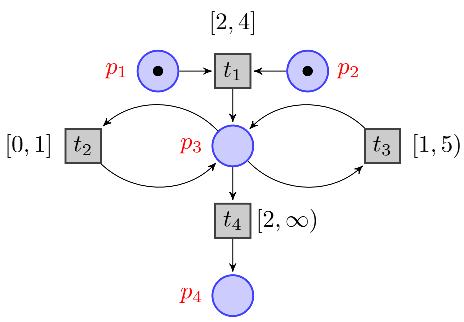

Figure 1: An example TPN

The clusters of a TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$ are the clusters of the underlying untimed net $\mathcal{U}$. The example of Figure 1 is a TPN. The time intervals attached to transition t_1, t_2, t_3, t_4 are respectively $[2, 4], [0, 1], [1, 5), [2, \infty)$. Note that intervals attached to transitions t_3 and t_4 are open. This net contains three clusters: $C_1 = \{p_1, p_2, t_1\}$ and $C_2 = \{p_3, t_2, t_3, t_4\}$ and $C_3 = \{p_4\}$.

So far, we have defined the syntax of TPNs, but their semantics is not obvious from the notation. In fact, several semantics for TPNs exist in the literature (see for instance [1, 17, 7, 3]). A frequently used semantics for TPNs is a *single server* semantics [1]: it attaches a clock x_t to each enabled transition t. This clock is initialized and starts measuring time as soon as transition t becomes enabled. Other semantics consider every occurrence of enabling of transitions and memorize the time elapsed since every enabling. This is called a multi-enabling semantics. In the next sections, we define formally these two frequently used semantics, namely the single server semantics of [1], and a multiserver semantics with a FIFO policy, which is a standard setting (see e.g., [7]).

2.2. Single server semantics for TPNs

A frequently used semantics for TPNs is a *clock-on-transition* and *single server semantics*, that can be found for instance in [1, 17, 3]. This semantics associates a clock x_t to every transition t of the considered TPN. Formally, given a TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$ with $\mathcal{U} = (P, T, F)$, we define a set of clocks $X_\mathcal{N} = \{x_t \mid t \in T\}$. A valuation ν is a map that associates a real value to every clock in $X_\mathcal{N}$. Intuitively, the value $\nu(x_t)$ defines the time elapsed since enabling of transitions t. Given a real value d, we denote by $\nu + d$ the valuation that associates to every clock x_t value $\nu(x_t) + d$. A *configuration* of $\mathcal{N}$ is a pair (M, ν) where M is a marking, and ν is a valuation. A transition is *enabled* in (M, ν) iff $M(p) > 1$ for every place $p \in {}^\bullet t$. It is *fireable* if $\nu(t) \in I(t)$. It is *urgent* if $\nu(t) = lft(t)$. From a given configuration (M, ν) , one can let $\delta \in \mathbb{R}$ time units elapse iff $\nu(t) + \delta < lft(t)$ for every transition t enabled in (M, ν), that is if no transition is urgent. Every fireable transition t in configuration (M, ν) can fire, leading to a configuration (M', ν'). A transition t_i is *newly enabled* by firing of t if t_i is enabled in M' and either $t_i \neq t$ and t_i is not enabled in $M \setminus {}^\bullet t$, or $t_i = t$. The *single server semantics* of a TPN can be formally defined as moves from a configuration (M, ν) to a configuration (M', ν'). These moves can be timed or discrete moves:

- **timed moves** are of the form $(M, \nu) \xrightarrow{d} (M, \nu + d)$, with $d \in \mathbb{R}$. Such move is allowed if for all $t, \nu(x_t) + d \leq lft(t)$ (or $\nu(x_t) + d < lft(t)$ if $I(t)$ is an open interval of the form $[a, b)$ or (a, b)), that is if no transition becomes urgent while letting d time units elapse

- **discrete moves** are moves of the form $(M, \nu) \xrightarrow{t} (M', \nu')$, where $M'(p) = M(p) \setminus {}^\bullet t + t^\bullet$ for every p in P, and for every $t_i \in T$, $\nu'(x_{t_i}) = 0$ if t_i is newly enabled by firing of t, and $\nu'(x_{t_i}) = \nu(x_{t_i})$ otherwise.

Consider the example of Figure 2. This net defines a set of timed words of the form $(t_1, d_1).\big((t_1, d_2) \dots (t_1, d_k)\big)||(t_2, d'_1)\big((t_2, d'_2) \dots (t_2, d'_k)\big)$ where $||$ is the usual shuffle of words, and that satisfies $\forall i \in 1..k, d_i \leq d'_i$, $\forall i \in 2..k, d_i - d_{i-1} \leq 1$, $d'_1 - d_1 = 1$ and $\forall i \in 2..k, d'_i - d'_{i-1} = 1$. Informally, t_1 produces tokens that are consumed by t_2. Tokens are produced at a rate of more than one token per time unit, but every instance of t_2 must occur exactly 1 time unit after the preceding instance of t_2, except for the first one, that occurs exactly one time unit after the first occurrence of t_1.

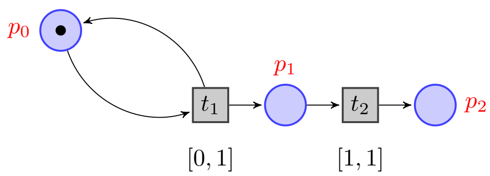

Figure 2: A TPN that discriminates between single and multi-server semantics

2.3. Multi-enabling semantics for TPNs

Let us now describe the semantics of Time Petri nets with multi-enabledness. In a multi-enabling semantics, when a marking associates to each place in the preset of a transition a number of tokens that is several times the number of tokens needed for the transition to fire, then this transition is considered as enabled *several times*, i.e. several occurrences of this transition are waiting to fire. Defining a multi-enabling semantics needs to address carefully which instances of enabled transitions can fire, what happens when an instance of a transition fires, in terms of disabling and creation of other instances. Several policies are possible as discussed in [7, 18, 19]. In the rest of the paper, we adopt a semantics where the oldest instances of transitions fire first, are subject to urgency requirements, and the oldest instances are also the disabled ones when a conflict occurs. We formalize this below.

Definition 2. *Let M be a marking of a TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$. A transition $t \in T$ is* k-enabled *at marking M, for $k > 0$, if for all $p \in {}^\bullet t, M(p) \geq k$ and there exists a place $p \in {}^\bullet t$ such that $M(p) = k$. In this case, k is called the* enabling degree *of t at marking M, denoted $deg(M, t)$.*

Therefore, from a marking M, and for each transition t, there are exactly $deg(M, t)$ instances of transition t which are enabled. A configuration of a time Petri net under multi-server semantics associates a clock with each instance of a transition that has been enabled. This is called *threshold semantics* in [7]. Time can elapse if no urgency is violated, and an occurrence of a transition t is allowed to fire if it has been enabled for a duration that lays within the interval attached to t. The notion of configuration in a multi-enabled semantics differs from that used in single-server semantics, as clocks used in a single-server semantics can only remember *one* elapsed duration per transition.

Formally, a *configuration* in a multi-server semantics is a pair $C = (M, enab)$, where M is a marking, and $enab$ is a partial function $enab : T \to (\mathbb{R}_{\geq 0})^*$. We denote by $enab(t)_i$ the i^{th} entry of $enab(t)$, and by $dom(enab)$ the set of transitions that have at least one enabled instance, i.e. such that $enab(t) \neq \epsilon$. For a marking M, $t \in T$, $enab(t)$ is defined iff there exists $k > 0$ such that t is k-enabled at M. We further require that the length of vector $enab(t)$ is exactly $deg(M, t)$, and if $1 \leq i < j \leq deg(M, t)$, then $enab(t)_i \geq enab(t)_j$. Intuitively, in a configuration $C = (M, enab)$ each enabled transition t is associated with a vector $enab(t)$ of decreasing real values, that record the time elapsed since each instance of t was newly enabled. With this convention, the first enabled instance of a transition t (that must have the maximal clock value) is the first entry $enab(t)_1$ of the vector. For a value $\delta \in \mathbb{R}$, we will denote by $enab(t) + \delta$ the function that associates $enab(t)_i + \delta$ to the i^{th} occurrence of t. The initial configuration is a configuration $C_0 = (M_0, enab_0)$ such that for every $t \in T$ we have $enab_0(t) = 0^{deg(M,t)}$, i.e. $enab_0(t)$ associates $deg(M_0, t)$ entries to each enabled occurrence of t and initializes their clocks.

The semantics of TPNs under multi-server semantics differs from the single server semantics. In particular, one has to remember for each enabling instance of a transition the age of this enabling, and define which instances of other transitions are competing for the same resources. As for single server, the multi-server semantics will be decomposed into timed and discrete moves. Discrete moves fire the oldest instance of some transition t, disables conflicting instances of other transitions, and enable new ones. The set of *conflicting transitions* and the set of *newly enabled* transitions are computed as follows: Let t and t' be two transitions enabled in M and let $k = deg(M, t)$ and $k' = deg(M', t')$ be their enabling degrees at marking M. Transition t is in conflict with transition t' in M iff in the intermediate marking $M'' = M \setminus {}^{\bullet}t$ computed when firing t, the enabling degree of t' is decreased w.r.t M, i.e if firing t consumes one token from at least a place place p with marking $M(p) = deg(M, t')$ in the preset of t'. So if the enabling degree of t' at M is k' and its enabling degree at $M'' = M - {}^{\bullet}t$ is $k' - 1$ then one enabled instances of t' is disabled when firing t to move from M to M''. We disable transitions according to the First Enabled First Disabled (FEFD) policy: disabling the oldest instance of a transition t' simply consists in removing $enab(t')_1$ from $enab(t')$. We will denote by $cnfl(M, t)$ the set of enabled transitions that are in conflict with t at marking M. Let $newenab(M, t)$ denote the set of newly enabled instances in the marking reached from M after firing oldest instance of transition t. If a transition t' has k' enabled instances at M'' and $k' + 1$-enabled instances M' as defined in the above paragraph, then firing t create one new enabled instance of t'. Note that as we consider intermediate markings, firing t can disable the oldest instance of some transition t', and at the same time create a new enabled instance of t'.

With these policies in place, we are ready to define formally the multi-server semantics of TPNs. Let $C = (M, enab)$ and $C' = (M', enab')$ be configurations. A move from C to C' can be either a timed move (that simply lets time elapse), or a discrete move, that represents a transition firing. A transition t is *urgent* in a configuration $C = (M, enab)$ iff $enab(t)_1 = lft(t)$.

A **timed move** from a configuration $C = (M, enab)$ consists in letting δ time units elapse, i.e. move to a new configuration $C' = (M, enab + \delta)$. Such move is allowed only if it does not violate urgency. Formally, a timed move of $\delta > 0$ time units from a configuration $(M, enab)$ to a transition $(M', enab')$ is denoted by $(M, enab) \xrightarrow{\delta} (M', enab')$, and is allowed iff

- $M' = M$,
- for every $t \in dom(enab)$, $enab(t)_1 + \delta \leq lft(t)$ if $I(T)$ is closed on its right endpoint, (or $enab(t)_1 + \delta < lft(t)$ if $I(t)$ is not closed on its right endpoint).

Note that urgency disallows time elapsing: as soon as a transition t is urgent, i.e. $enab(t)_1 = lft(t)$, one can not increase its clock value, and has to fire t or a conflicting transition t' that discards the first enabled instance of t before elapsing time.

A **discrete move** consists of firing an enabled instance of a transition t which clock lays in interval $I(t)$. When firing transitions, we will use the *First Enabled First Fired (FEFF)* policy, that is the instance of transition t fired is the one with the highest value of time elapsed, i.e., $enab(t)_1$. A standard way to address time in TPNs semantics is to start measuring time for a transition as soon as this transition is newly enabled. However, as highlighted in [3], there are several possible interpretations of new enabledness.

A frequently used semantics is the *intermediate semantics*, which considers intermediate markings, i.e. when firing a transition t, we consider the marking $M'' = M - {}^\bullet t$ obtained after removing the tokens in ${}^\bullet t$ from M, and comparing the set of enabled transitions in M'' with those enabled after production of tokens in the places of $t^\bullet$. Let transition t be k-enabled at marking M for $k > 0$, and $enab(t)_1 \in I(t)$. Then, an instance of t can fire, and we obtain a new marking M' via an intermediate marking M'' as follows:

$M''(p) = M(p) - 1$ if p in ${}^\bullet t$ else $M''(p) = M(p)$.

Then we define marking M' as $M'(p) = M''(p) + 1$ if p in $t^\bullet$ and $M'(p) = M''(p)$ otherwise.

Firing an instance of a transition t changes the enabling degree of several transitions (and not only of t). We will say that a transition is *newly enabled* by firing of t from M iff its enabling degree is larger in M' than in M''. Then newly enabled transitions are attached an additional clock initially set to 0 in *enab*. For transitions which enabling degree decreases in M'' (i.e. that are in conflict with t), the first clock in *enab* is discarded.

Thus, formally, a *discrete move* that fires an instance of transition t from a configuration $(M, enab)$ to reach a configuration $(M', enab')$ is denoted by $(M, enab) \xrightarrow{t} (M', enab')$, and is allowed iff:

- $t \in dom(enab)$, and $enab(t)_1 \in I(t)$
- for every $p \in P$, $M'(p) = M(p) - {}^\bullet t(p) + t^\bullet(p)$,
- $enab'$ is computed as follows. Let $enab(t) = (v_1, \dots v_k)$, and let $enab(t') = (v'_1, \dots v'_{k'})$ for every $t' \in dom(enab(M'))$.

  We first define $enab''$ as $enab''(t) = (v_2, \dots v_k)$ or ϵ if $deg(M, t) = 1$. Then for every $t' \in T \setminus \{t\}$,

$$enab''(t') = \begin{cases} (v'_2, \dots v_{k'}) \text{ if } t' \in dom(enab) \cap cnfl(M,t), \\ \quad \text{and } deg(M,t') > 1 \\ \epsilon \text{ if } t' \in dom(enab) \cap cnfl(M,t), \\ \quad \text{and } deg(M,t') = 1 \\ enab(t') \text{ otherwise.} \end{cases}$$

  Last, for every $t_i \in T$, we set:

$$enab'(t_i) = \begin{cases} enab''(t_i).0 \text{ if } deg(M',t_i) = deg(M'',t_i) + 1 \\ \quad \text{and } t_i \in dom(enab''), \\ \\ enab''(t_i) \text{ if } deg(M',t_i) = deg(M'',t_i) \\ \quad \text{and } t_i \in dom(enab''), \\ \\ (0) \text{ if } t_i \notin dom(enab''), \text{ and } deg(M',t_i) = 1 \\ \\ \epsilon \text{ otherwise.} \end{cases}$$

Intuitively, when an instance of transition t is fired at configuration $(M, enab)$, for every transition t_i which is in conflict with t, the first enabling instance of t_i is removed from the $enab(t_i)$ list, i.e. we remove from $enab(t_i)$ the value representing the age of the oldest enabling instance of t_i to obtain $enab''$. Similarly, moving tokens when firing t creates new enabling instances of some transitions. The clock attached to each new instance of some newly enabled transition t_j is set to 0 and this instance is inserted at the end of vector $enab''(t_j)$ to obtain $enab'$. Note that the timing information attached to a transition instance is not reset in the time period starting from the moment it is inserted into the *enab* list to the moment it is removed from the list (either after being fired or disabled).

Consider again the example net of Figure 2. Under the multi-server semantics, this net defines a set of timed words of the form :
$(t_1, d_1).\big((t_1, d_2) \dots (t_1, d_k) || (t_2, d'_1).(t_2, d'_2) \dots (t_2, d'_k)\big)$ where $\forall i \in 1..k, d'_i - d_i = 1$ and $\forall i \in 2..k, d_i - d_{i-1} \leq 1$. Informally, t_1 produces an arbitrary number of tokens every time unit. Each token produced is consumed by t_2 exactly 1 time units after its production. Clearly, such a language cannot be encoded with a TPN under single server semantics, as this needs memorizing time elapsed since the creation of each token, and single

server semantics associates only one clock to every transition. Conversely, the set of timed words for this net under the single server semantics is not contained in the language of a TPN under multi-server semantics, as this semantics cannot remember the time elapsed between two consecutive firings of the same transition as soon as multiple instances of this transition are enabled. Thus, the single and multi-server semantics define incomparable timed languages.

2.4. Sequences, timed languages, and diverging behaviors

TPNs can be seen as defining timed languages over the alphabet of transitions. Regardless of the used semantics, we will write $C \longrightarrow_{\mathcal{N}} C'$ when there exists a move from C to C' allowed by net $\mathcal{N}$. A *timed firing sequence* of $\mathcal{N}$ starting from configuration q_1 is a sequence of timed and discrete moves $\rho = q_1 \xrightarrow{\alpha_1} q_2 \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_{n-1}} q_n$ where α_i is either a transition of T ($q_i \xrightarrow{\alpha_i} q_{i+1}$ is a discrete move) or a value from $\mathbb{R}_{\geq 0}$ ($q_i \xrightarrow{\alpha_i} q_{i+1}$ is a timed move). A configuration C is reachable iff there exists a firing sequence from the initial configuration to C. Let $\mathit{Reach}(\mathcal{N}, C_0)$ denote the set of reachable configurations of a TPN $\mathcal{N}$, starting from configuration C_0. For a configuration $C = (M, \mathit{enab})$ in the multi server semantics, or $C = (M, v)$ in a single server semantics, we denote by $\mathit{Untime}(C) = M$ the *untiming* of configuration C. In a similar way, we let $\mathcal{R}(\mathcal{N}, C_0) = \mathit{Untime}(\mathit{Reach}(\mathcal{N}, C_0))$ be the set of (untimed) reachable markings, i.e., $\mathcal{R}(\mathcal{N}, C_0) = \{M \mid \exists (M, v) \in \mathit{Reach}(\mathcal{N}, C_0)\}$ in the single server semantics, and $\mathcal{R}(\mathcal{N}, C_0) = \{M \mid \exists (M, \mathit{enab}) \in \mathit{Reach}(\mathcal{N}, C_0)\}$ in the multi server semantics. Note that Reach and Untime operations are not commutative and $\mathit{Reach}(\mathit{Untime}(\mathcal{N}))$ can be different from $\mathit{Untime}(\mathit{Reach}(\mathcal{N}, C_0))$ for a TPN $\mathcal{N}$. This remark holds for both semantics. A TPN $\mathcal{N}$ is *bounded* if $\mathit{Untime}(\mathit{Reach}(\mathcal{N}))$ is finite, and *safe* if for every configuration in $\mathit{Reach}(\mathcal{N}, C_0)$, the marking part M of reached configurations is such that $M(p) \leq 1$ for every $p \in P$.

A *timed word* over T is a word of $(T \times \mathbb{R}_{\geq 0})$. Let $\rho = q_1 \xrightarrow{\alpha_1} q_2 \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_{n-1}} q_n$ be a firing sequence starting from q_1, and let $i_1, \dots i_k$ denote the indexes of discrete moves in ρ. The timed word associated with ρ is the word $w = (t_1, d_1) \dots (t_k, d_k)$, where each t_j is transition α_{i_j}, $d_1 = \sum_{j<i_1} \alpha_j$, and for every $m > 1$, $d_m = d_{m-1} + \sum_{i_{m-1}<j<i_m} \alpha_j$. A timed word w is *enabled* at configuration C if there exists a timed firing sequence ρ starting from C, and w is the timed word associated with this sequence. The untiming of a timed word $w = (t_1, d_1) \dots (t_n, d_n)$ is the word $\mathit{Untime}(w) = t_1 \dots t_n$.

For a TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$, let $\mathit{Lang}(\mathcal{N})$ denote the set of all timed words enabled at initial configuration C_0, and $\mathit{ULang}(\mathcal{N})$ be defined as $\mathit{Untime}(\mathit{Lang}(\mathcal{N}))$, i.e., the set of words obtained by removing the timing component from words of $\mathit{Lang}(\mathcal{N})$. Observe that $\mathit{ULang}(\mathcal{N})$ can be different from $\mathit{Lang}(\mathit{Untime}(\mathcal{N}))$.

Proposition 1. *The sets of timed languages defined by TPNs under the single server and multi-server semantics differ, except for the class of safe TPNs.*

Proof: The example of Figure 2 shows a Petri net which timed language under the single server semantics is not the language of a net under the multi-server semantics, and conversely. For the second part of the proposition, it suffices to remark that in the single-server semantics, x_t remembers the same information as $\mathit{enab}(t)_1$ in the multi-server semantics, to show that single and multi-server semantics of every safe net are in fact timed bisimilar. □

Definition 3. *A* forced 0-delay firing sequence *(0-delay firing sequence for short) of $\mathcal{N}$ is a sequence from $(Q \times T)^\infty$ such that $q_{i-1} \xrightarrow{\alpha_i} q_i$ where α_i is a transition of T with $eft(\alpha_i) = lft(\alpha_i) = 0$.*

In 0-delay firing sequences, transitions are enabled at each configuration and fired immediately without letting any other transition of the net fire. One can consider that TPNs with 0-delay firing sequence are ill-formed, as they allow diverging behaviors that take no time. A correct specification should not allow such behaviors. Note that forbidding 0-delay firing sequences does not prevent infinite sequences of 0-duration. For example, a TPN in which each transition has $eft(t) = 0$ and $lft(t) > 0$ can have infinite sequences of 0-duration.

3. Processes of untimed and timed nets

We now define a partial-order semantics for timed nets with multi-enabledness using processes. These notions will be central to reason about TPNs and their properties. The notion of time causal processes for TPNs have been introduced by [12], and later used by [14] to study a truly concurrent semantics for TPNs. First, we recall the definitions in the untimed setting.

Definition 4 (causal net). *A causal net $ON = (B, E, G)$ is a finitary (each node has finite number of predecessors) acyclic net with B as a set of conditions, E as set of events, flow relation $G \subseteq B \times E \cup E \times B$, such that $E = \{e \mid (e, b) \in G\} \cup \{e \mid (b, e) \in G\}$, and for any condition b of B we have $|\{e \mid (e, b) \in G\}| \leq 1$ and $|\{e \mid (b, e) \in G\}| \leq 1$.*

When causal nets are used to give a partial order semantics to Petri nets, events represent occurrences of transitions firings, and conditions occurrences of places getting filled with tokens. The last condition in the definition states that a token in a place is produced by a single transition firing, and is consumed by a single transition. As conditions have a single successor, causal nets are *conflict free*. They are hence well suited to represent executions of nets. Let $\prec = G^+$ and $\preceq = G^*$. We define $e\downarrow$ as the downward closure $e\downarrow = \{f \in E \mid f \preceq e\}$. A set $E' \subseteq E$ is downward closed iff $E'\downarrow = E'$. We define $ON^\bullet = \{b \in B \mid b^\bullet = \emptyset\}$ and ${}^\bullet ON = \{b \in B \mid {}^\bullet b = \emptyset\}$; Note that if we are looking at finite causal nets then $ON^\bullet$ is always defined. For any downward-closed subset E' of E we define $Cut(E') = \{E'^\bullet \cup {}^\bullet ON\} \setminus {}^\bullet E'$. As conditions are occurrences of places, $Cut(E')$ can be interpreted as the marking of $\mathcal{N}$ reached after executing E'.

Definition 5 (causal process). *Given a PN $\mathcal{U}$, a causal process of $\mathcal{U}$ is a pair $U = (ON, \pi)$ where, π is mapping from $B \cup E \to P \cup T$, satisfying following conditions:*

1. *$\pi(B) \subseteq P$ and $\pi(E) \subseteq T$,*
2. *$\pi\downarrow_{e^\bullet}$ is a bijection between sets $e^\bullet$ and $\pi(e)^\bullet$ (where π_X denotes the restriction of map π to elements of X),*
3. *$\pi\downarrow_{{}^\bullet e}$ is a bijection between sets ${}^\bullet e$ and ${}^\bullet\pi(e)$,*
4. *For each p, $M_0(p) = |\{b \in {}^\bullet ON \mid \pi(b) = p\}|$.*

To relate transitions of a PN and events of a causal net, we will consider that events are of the form $e = (X, t)$, where $\pi(e) = t$ and $X = {}^\bullet e$ i.e., event e corresponds to firing of transition t and X is a set of conditions consumed when firing t. We will also let conditions be of the form $b = (e, p)$, where e is the event that creates condition b, and $p = \pi(b)$ is the place that is represented by condition b. We will write $place(b)$ to denote such place in the original net. We let $posb(ON)$ denote the set of possible events which could be added to ON and is defined as $posb(ON) = \{e = (X, t) \mid X \subseteq B \wedge \pi(X) = {}^\bullet t \wedge \forall x \in X,\ x^\bullet = \emptyset\}$. Following these definitions, one can inductively build processes to capture the semantics of (even unbounded) Petri nets. The construction is as follows : we start from initial causal process $U_0 = (ON_0, \pi_0)$, where $ON_0 = (B_0, E_0)$ is an occurrence net. B_0 is a multiset of conditions computed from M_0 that associates a condition of the form $b = (\perp, p)$ to every marked place in M_0, E_0 is an empty set of events, and π_0 maps every condition $(\perp, p)$ to place p. A causal processes U_{i+1} is built inductively from U_i by choosing an event from $posb(ON_i)$ and adding it to U_i with the obvious extension of map π_i. Obviously, causal processes capture the semantics of Petri nets, event in an unbounded setting. Causal processes can be seen as Petri nets, and hence the notions of enabled transition, markings, and languages apply. Given a causal process U for an Petri net $\mathcal{U}$, we define as $Lang(U)$ the set of words of the form $w = t_1 \dots . t_k$ such that there exists a sequence of transitions $w' = e_1 \dots e_k$ in U starting from marking B_0 and $w = \pi(w')$. Using the above definitions and by a straightforward induction, we have the following proposition:

Proposition 2. *Let $\mathcal{U}$ be any untimed net. For any word $w \in Lang(\mathcal{U}, M_0)$, there exists a causal process U of $\mathcal{U}$, such that $w \in Lang(U)$.*

Now we define the notions of time causal net, and time causal process for Time Petri nets with multi-enabling semantics. Similar notions appear in [12] for Time Petri nets.

Definition 6 (time causal net). *A* time causal net *is a tuple $ON = (B, E, G, \tau)$ where (B, E, G) is a causal net, and $\tau : E \to \mathbb{R}_{\geq 0}$ is a timing function such that $eG^+e' \Rightarrow \tau(e) \leq \tau(e')$.*

For a time causal net $ON = (B, E, G, \tau)$, we define $Untime(ON)$ as the net (B, E, G) i.e., ON without its timing function.

Definition 7 (Prefixes). *Given two untimed causal nets $ON = (B, E, G)$ and $ON' = (B', E', G')$, ON' is said to be a* prefix *of ON, denoted by $ON' \leq ON$ if $B' \subseteq B$, $E' \subseteq E$, $G' = G \cap (B' \times E' \cup E' \times B')$, where E' is finite and downward closed subset of E.*

Given two time causal nets $ON = (B, E, G, \tau)$ and $ON = (B', E', G', \tau')$, ON' is a prefix of ON iff $(B', E', G') \leq (B, E, G)$ and E' is timely sound*, i.e., for all e' in E' we have $\tau'(e') \leq \tau(e')$.*

Definition 8 (time causal process). *A* time causal process *of a TPN $\mathcal{N}$ is a pair $N = (ON, \pi)$ where $ON = (B, E, G, \tau)$ is a timed causal net, and π is a mapping from $B \cup E \to P \cup T$, satisfying conditions 1.-4. of a causal process, and:*

5. *for every event $e = (X, t)$, $\min_{x \in X}\{\tau(e) - \tau({}^\bullet x)\} \in I(\pi(e))$, i.e. the time elapsed between enabling of the occurrence of t represented by e and its firing belongs to $I(t)$.*
6. *if there exists $X \subseteq B$ such that $\forall x, x' \in X x \not\leq x'$, and a transition t such that $Place(X) = {}^\bullet t$ then either*
   - *$(X, t) \in E$, or*
   - *there exists $e' = (X', t') \in E$ such that $X' \cap X \neq \emptyset$ and $\tau(e') - \max(\tau({}^\bullet X')) \leq lft(t)$, or*
   - *$\max(\tau(E)) - \max(\tau({}^\bullet X)) < lft(t)$.*

   This last condition means that if a transition was urgent before the date of the last event in ON, then it belongs to the time causal net, or was not appended due to firing of a conflicting transition.

For a time causal process (ON, π) we define $Untime(ON, \pi)$ as $(Untime(ON), \pi)$. As for Petri Nets, for a timed causal net $ON = (B, E, G, \tau)$, we denote by $posb(ON)$ (and we define similarly) the set of events that can be appended to ON (regardless of timing considerations). Abusing our notation, for a condition $b = (e, p)$ we will define $\tau(b)$ as $\tau(b) = \tau(e)$.

As for Petri nets, we can show that time causal processes faithfully describe the semantics of Time Petri nets. Given a time causal process $N = (ON, \pi)$, where $ON = (B, E, G, \tau)$, a timed word of N is a timed word $w = (e_1, d_1) \dots e_{|E|}, d_{|E|}$ such that $e_1 \dots e_{|E|}$ is a linearization of $Untime(ON)$, $d_i = \tau(e_i)$. Note that as w is a timed word, this means in addition that for every $i < j$, we have $d_i < d_j$. We denote by $Lang(N)$ the set of timed words of time causal process N. Note that there exist some words $w \in Lang(Untime(N))$ such that w is not the untiming of a word in $Lang(N)$. We have:

Proposition 3. *Let $\mathcal{P}$ be the set of time causal processes of a time Petri net $\mathcal{N}$. Then, $Lang(\mathcal{N}) = \bigcup_{N \in \mathcal{P}} Lang(N)$*

Proof: We will proceed by induction on the length of words and processes. Let us define the property $P(i)$: for every word of length i, we have $\{w \in Lang(\mathcal{N}) \mid |w| = i\} = \bigcup_{N \in \mathcal{P}, |N| = i} Lang(N)$. Obviously, this property holds for $i = 0$, as the empty word ϵ of length 0 is a word of any process starting from initial conditions. Let us assume that $P(i)$ holds up to n. For a given timed word, there exists a unique firing sequence $C_0 \dots C_n$, where C_n is the configuration reached immediately after execution of event e_n. Let $ON = (B, E, G, \tau)$ be a time causal net of size n such that $w \in Lang(ON)$. At least one such net exists, as $P(i)$ holds up to n. Now suppose that w can be extended to a word of size $n + 1$, i.e. that C_n allows execution of an additional event e_{n+1} at date d_{n+1}, that is an occurrence of some transition t. There is a connection between clocks in configurations and dates in timed causal nets: every occurrence of a clock x_t^i attached to the i^{th} occurrence of a transition t is created when a new complete occurrence of ${}^\bullet t$ is created. For event e_{n+1}, the date of creation of such clock is a date $d_k \leq d_{n+1}$ of occurrence of some event e_k, that appended enough new tokens in ${}^\bullet t$ to increase the degree of t. When e_{n+1} fires, it consumes tokens from a marking, that are maximal conditions in ON. So, there is a correspondence between clocks instances and conditions in time causal nets: $ON^\bullet$ is an union of conditions that contains occurrences of places $p_1, ..p_j \in {}^\bullet t$, and such that $\max\{\tau(p_1), \ldots, \tau(p_j)\} = d_k$. Note that d_k is not necessarily the maximal date attached to a condition in ON, but is maximal in ${}^\bullet e_{n+1} \cap ON^\bullet$. Now if e_{n+1}, representing an occurrence of t can occur at date d_{n+1} starting from $C_n = (ON^\bullet, enab)$ then there exists a clock $x_t^i \in enab$ and a delay δ such that x_t^i is the oldest remaining clock created for an instance of t in C_n and $x_t^i + \delta = d_{n+1}$, and $\delta \in I(t)$. Recall that x_t^i was initialized at date d_k. Hence, there exists a set of conditions X in $ON^\bullet$ such that the places appearing in X are exactly ${}^\bullet t$, $d_{n+1} - max(\tau(X)) = \delta \in I(t)$, and hence the process ON' obtained by adding $\{e_{n+1} = (X, t)\}$ to E and $\{(e_{n+1}, p) \mid p \in t^\bullet\}$ to B and setting $\tau(e_{n+1}) = d_{n+1}$ is a process of $\mathcal{N}$, as $d_{n+1} - d_k \in I(t)$, we have $\tau(e_{n+1}) - \tau(e_k) = \min_{x \in X}(\tau(e_{n+1}) - \tau({}^\bullet x)) \in I(t)$. One can prove similarly that if a new event e_{n+1} can be appended to ON, then this event is allowed from C_n after elapsing $\tau(e_{n+1}) - \max(\tau(E))$ time units, and hence $w.(e_{n+1}, \tau(e_{n+1}))$ is a word of $\mathcal{N}$ of size $n + 1$. $\square$

Example 1. *Consider the FC-TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$ shown in Figure 14. Nets ON_1 and ON_2 in Figure 3 are causal nets of Untime($\mathcal{N}$) starting from M_0. One can notice that $ON_1 \leq ON_2$.*

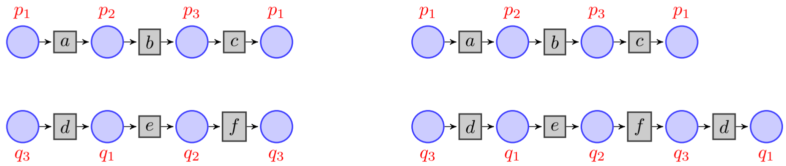
Figure 3: Untimed causal nets ON_1 and ON_2

Figure 4 below illustrates timed causal nets. Nets ON_3 and ON_4 in Figure are timed causal nets of $\mathcal{N} = (\mathcal{U}, M_0, I)$. In this Figure, we have $ON_3 \leq ON_4$. Let us now compare ON_3 and ON_4 with (untimed) causal processes of Figure 3: we have $ON_1 \leq$ Untime(ON_3) and $ON_2 \leq$ Untime(ON_4).

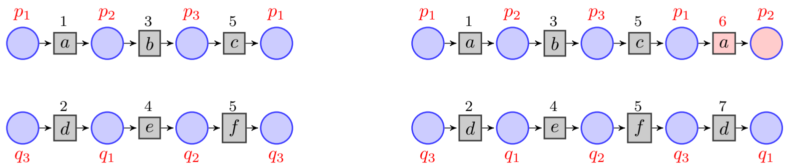
Figure 4: Timed causal net ON_3 and ON_4

4. Free Choice Time Petri Nets

Time Petri nets with urgent semantics [1] are rather expressive. They can be used to model distributed timed systems with unbounded resources. Unsurprisingly, most problems, particularly those listed below are undecidable for this model:

- **Fireability:** Given a TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$ and a transition t, is there a configuration $(M, enab) \in Reach(\mathcal{N}, C_0)$ such that t is fireable at $(M, enab)$.
- **Termination:** Given a TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$, does it have a non-terminating or infinite run?
- **Coverability:** Given a TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$ and a marking M, is there a marking $M' \in Reach(\mathcal{N}, M_0)$ such that $M' \geq M$?
- **Reachability:** Given a TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$ and a marking M, decide if $M \in Reach(\mathcal{N}, M_0)$.
- **Boundedness:** Given a PN $\mathcal{N} = (\mathcal{U}, M_0, I)$ decide if $Reach(\mathcal{N}, M_0)$ is finite.

To obtain decidability, one often considers bounded TPNs, where the number of tokens in places cannot grow arbitrarily. In this case, TPNs can be translated into finite timed automata (see for instance [20]). As a consequence, all properties decidable on timed automata are decidable for bounded TPNs. However, bounded TPNs cannot represent systems with unbounded resources. Furthermore, it is undecidable in general whether a TPN is bounded. One usually has to rely on a priori known bounds on place contents, or restrict to the class of nets such that $Untime(\mathcal{N})$ is bounded.

Considering multi-server semantics in unbounded nets does not change decidability issues in general: undecidability originates from the fact that one can encode a counter machine: counters are encoded with places, and zero test with urgent transitions (one can find such an encoding in [3]). However, we will show that multi-server semantics, beyond the practical fact that it allows encoding of queues with latency, also contain non-trivial decidable subclasses. Hence, in the rest of the paper, we will mainly focus on TPNs under multi-server semantics. In the rest of the paper, we consider a structural restriction of TPNs, which is based on the untimed underlying PN, namely free-choice. This is a standard restriction in the untimed setting, that does not impose boundedness of nets, and allows for efficient algorithms (see [8]). In this section and next, we show the interesting properties it engenders in TPNs and subsequently we will show how it affects their robustness under guard enlargement and shrinking.

Definition 9 (Free choice PN and Free choice TPN). *A Petri net $\mathcal{U} = (P, T, F)$ is called* (extended) free choice, denoted FC-PN, *if for any pair of transitions t and t' in T: ${}^\bullet t \cap {}^\bullet t' \neq \emptyset \implies {}^\bullet t = {}^\bullet t'$. A TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$ is called a* free choice TPN *(FC-TPN for short), if its underlying untimed net $Untime(\mathcal{N}) = \mathcal{U}$ is free choice.*

4.1. Pruning a TPN while preserving reachability

As mentioned above, the firability and termination problems are undecidable for TPNs in general. In next section we show that they are decidable for FC-TPNs under multi-server semantics. As a first step, given an FC-TPN $\mathcal{N}$, whose underlying net is free choice, we construct another FC-TPN $Prune(\mathcal{N})$ in which we remove transitions from a cluster if they can never be fired (due to the lower bound of their time constraint) and tighten timing constraints. Note that we do not delete *all* dead transitions from the net, but remove only transitions for which we can decide locally, by considering their clusters, that they will never fire. Let us illustrate this with an example.

Example 2. *Consider the FC-TPN $\mathcal{N}$ in Figure 5. Consider transition b, and its cluster $[b]$. One can notice from the definition of FC-TPNs that all transitions from the same cluster have a new instance created as soon as any transition from the same cluster has a new enabling instance. Note also that in this example it is clear that transition c will never be fired: in a configuration $(M, enab)$, every enabling instance of c is created at the same time as another instance of b and d, and hence we have $enab(c) = enab(b) = enab(d)$. Let $enab(c) = enab(b) = r_1 \dots r_k$. Transition b has to fire or be disabled at $lft(b) - r_1$, $lft(b) - r_2$, $\dots$. If b fires or is disabled, then the oldest instance of c is also disabled. As we have $eft(c) > lft(b)$ every i^{th} instance of b will fire or be disabled before the i^{th} instance of c is allowed to fire. Hence one can safely remove transition c without changing the semantics of the net.*

Similarly, in the cluster $[e]$, we cannot say for sure that some transition will be never fired, but only that the maximum amount of time that an enabling instance of a transition in $[e]$ can let elapse after its creation is 2 time units. Hence, we can safely set 2 as upper bound for intervals $I(e)$ and $I(f)$ without changing the semantics of the net. Note that, in fact, neither transition f nor e is firable in this net, but we cannot decide it just by looking at the clusters. Indeed in order to decide if e and f are firable, we have to study the behaviour of the net. Hence our pruning operation does not modify this. Thus, after removing transition c from its cluster, modifying flow relation accordingly, and changing the upper bounds of remaining transitions, we obtain the free choice net in Figure 6. One can see that $Reach(\mathcal{N}, M_0) = Reach(Prune(\mathcal{N}), M_0)$. Therefore, we also get that $Lang(N, M_0) = Lang(Prune(N), M_0)$.

Formally, we can define pruning as follows:

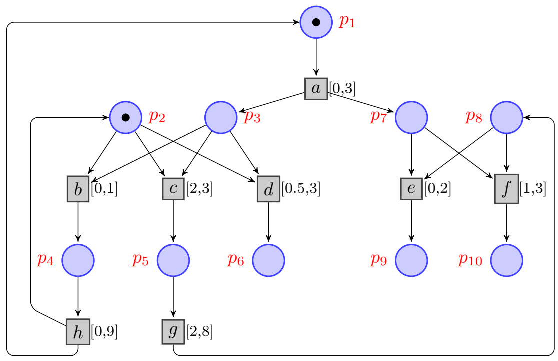

Figure 5: a FC-TPN $\mathcal{N}$

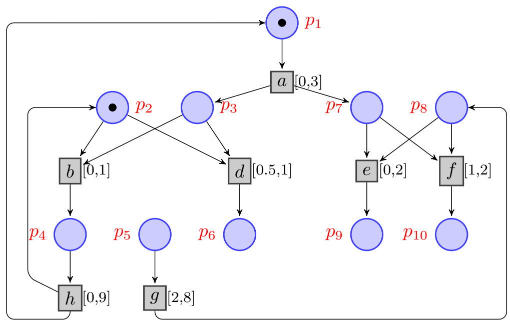

Figure 6: The pruned net $Prune(\mathcal{N})$

Definition 10 (pruned cluster and pruned net). *Given an FC-TPN* $\mathcal{N} = (\mathcal{U} = (P, T, F), M_0, I))$, *we define* pruned cluster *of a transition* t *as* $Prune([t]) = \{t' \in [t] \mid eft(t') \leq \min_{t'' \in [t]}(lft(t''))\}$. *The pruning of a FC-TPN* $\mathcal{N}$ *is the* pruned net $Prune(\mathcal{N}) = (\mathcal{U}' = (P, T', F'), M_0, I')$, *where:*

- $T' = \bigcup_{t \in T} Prune([t]) \cap T$,
- $F' = F \cap \big((P \times T') \cup (T' \times P)\big)$
- *For each transition* t *in* T', *we define* $I'(t) = \big(eft(t), \min_{t' \in [t]} lft(t')\big)$.

Lemma 1 below shows that pruning away unfirable transitions from clusters of a FC-TPN, does not modify its semantics. More precisely, the LTS obtained for a pruned FC-TPN is isomorphic to the LTS for the original net. This is not surprising and has already been considered in [15], where pruning is called "normalization" and is used to reason about free-choice but safe untimed nets.

Lemma 1 (Pruning Lemma). *Let* $\mathcal{N}$ *be a FC-TPN, and* $\mathcal{N}' = Prune(\mathcal{N})$. *Then,* $\big(Reach(\mathcal{N}, C_0), \longrightarrow_{\mathcal{N}}\big)$ *is isomorphic to* $\big(Reach(\mathcal{N}', C_0), \longrightarrow_{\mathcal{N}'}\big)$.

Proof: We can build a relation $\mathcal{R}$ from configurations of $\mathcal{N}$ to configurations of $\mathcal{N}'$, and show that this relation is an isomorphism. A configuration is a pair $\mathcal{C} = (M, enab)$ where M is a marking, and $enab$ assign a finite sequence of real values $enab(t)$ (clock values) to every transition of the net. For every configuration $\mathcal{C}'$ of $\mathcal{N}'$, $enab(t)$ is a map that attaches to every transition t a set of real values $r_1^t, r_2^t, \dots r_{deg(t)}^t$. One can notice that in configurations of free choice nets, all transitions from a cluster are newly enabled at the same date, and hence are attached the same valuations.

Let us now consider a transition t' that was pruned out. This transition comes from a cluster $[t']$, that contains a set of transitions $T(t') = t_{1,T(t')}, \dots, t_{k,T(t')}, t'$ of size greater than 1. Let $\mathcal{R}()$ be the relation that associates to every configuration $C' = (M', enab')$ from $\mathcal{N}'$ the configuration $\mathcal{R}(C') = (M', enab)$, i.e. with identical marking M', and such that $enab(t) = enab'(t)$ if t is not a pruned transition, and $enab(t) = enab'(t_{1,T(t')})$ otherwise. This relation $\mathcal{R}$ is reversible, and $\mathcal{R}^{-1}(C)$ is simply obtained using a restriction of $enab$ to unpruned transitions. Roughly speaking, $\mathcal{R}$ copies values of clocks attached to any unpruned transition and attaches them to pruned transitions in the cluster to obtain a configuration of $\mathcal{N}$. For a particular set of clocks $enab(t) = \{r_1^t, r_2^t, \ldots r_{deg(t)}^t\}$ and a real value $\delta \in \mathbb{R}$, we will denote by $enab(t) + \delta$ the set $\{r_1^t + \delta, r_2^t + \delta, \ldots r_{deg(t)}^t + \delta\}$.

We can now prove that $\mathcal{R}$ is an isomorphism between $\big(Reach(\mathcal{N}, C_0), \longrightarrow_{\mathcal{N}}\big)$ and $\big(Reach(\mathcal{N}', C_0), \longrightarrow_{\mathcal{N}'}\big)$. First of all, let C_0' be the initial configuration of $\mathcal{N}'$ and C_0 be the initial configuration of $\mathcal{N}$. We obviously have $\mathcal{R}(C_0') = C_0$. Now, we have to prove that for every timed or discrete transition $C_1' \longrightarrow_{\mathcal{N}'} C_2'$ and every configuration C_1 such that $\mathcal{R}(C_1') = C_1$, we have $C_1 \longrightarrow_{\mathcal{N}} C_2$ and $C_2 = \mathcal{R}(C_2')$.

- $C_1' \xrightarrow{\delta}_{\mathcal{N}'} C_2'$. We have $enab_2'(t) = enab_1'(t) + \delta$ for every transition t, and δ violates no urgency, i.e. for every transition t, the maximal value r in $enab_1'(t)$ is such that $r + \delta \leq lft(t)$. Obviously, in $C_1 = \mathcal{R}C_1'$, values of clocks are unchanged for unpruned transitions. As for pruned transitions the latest firing time is greater than the minimal latest firing time of transition in the same cluster, then a timed move of δ from C_1 violates no urgency, and is also allowed in C_1. Elapsing time from C_1 results in a new configuration $C_2 = (M_1', enab_1 + \delta)$. One can easily show that $C_2 = \mathcal{R}(C_2')$.

- $C_1' \xrightarrow{t}_{\mathcal{N}'} C_2'$. This transition t can fire from C_1' if $enab'(t)_1 \in [eft(t), lft(t)]$, and the marking M_1 associated with C_1' is greater than $pre(t)$. This transition can also fire from C_1 as $\mathcal{R}$ does not change markings nor clocks of unpruned transitions (and t is necessarily an unpruned transition), and as the time interval $I(t)$ attached to a transition in $\mathcal{N}$ contain the interval $I'(t)$ attached to t in $\mathcal{N}'$. The effect of firing t on markings is the same from C_1' and C_1', i.e. markings M_2' and M_2 are identical in C_2' and C_2. Let us now consider the clock part. Firing t removes the first clock value from $enab_1'(t_i)$ (resp from $enab_1(t_i)$) for every transition t_i such that $pre(t_i) = pre(t)$, i.e., transition from the same cluster as t, and adds a clock with value 0 to every transition which degree is modified w.r.t the intermediate marking. In C_2' and C_2 modified clock values are identically updated for unpruned transitions in $enab_2'$ and $enab_2$, and clock values in $enab_2$ for pruned transitions remains copies of clock values for transitions in their cluster. Hence, we still have $C_2 = \mathcal{R}(C_2')$

Let us now prove that $\mathcal{N}$ does not allow additional transitions.

- Suppose $C_1 \xrightarrow{\delta}_{\mathcal{N}} C_2$ and not $C_1' \xrightarrow{\delta}_{\mathcal{N}'} C_2'$ or $C_2 \neq \mathcal{R}(C_2')$. Note that for a chosen δ allowed in a configuration, the next configuration reached after elapsing δ time units is deterministically chosen. If δ can fire, then obviously $C_2 = \mathcal{R}(C_2')$. So the remaining possibility is that elapsing δ time units is not allowed from C_1'. However, this is impossible, as C_1 has more transitions than C_1' for each cluster, identical clocks attached to unpruned transitions but imposes the same constraint on possible values of δ: requiring $enabt1 + \delta \leq \min\{lft(t') \mid t' \in [t]\}$ for every t is the same constraint as $enab't1 + \delta \leq \min\{lft'(t') \mid t' \in [t]\}$.

- Suppose $C_1 \xrightarrow{t}_{\mathcal{N}} C_2$ and not $C_1' \xrightarrow{t}_{\mathcal{N}'} C_2'$ or $C_2 \neq \mathcal{R}(C_2')$. Transition t is allowed in configuration $C_1 = (M_1, enab_1)$ iff $M_1 \geq pre(t)$, and $\max(enab_1(t)_1) \in [eft(t), lft(t)]$. As we have $C_1' = (M_1, enab_1')$, the first condition is met, and as t can only be an unpruned transition, $enab_1'(t)_1 = enab_1(t)_1$, so firing t is also enabled in C_1'. It hence remains to show that the (unique) pair of configurations C_2, C_2' obtained after firing t from C_1 and C_1' is still in $\mathcal{R}$. As the marking part of configurations is the same in C_1 and C_1', it is also the same in C_2 and C_2', and it remains to compare the clock part. The clocks attached to transitions by $enab_2(t)$ (resp. $enab_2'(t)$) are updated iff the degree of t is modified either from M_1 (resp. M_1') to the intermediate marking $M_1 \setminus pre(t)$ (resp. $M_1' \setminus pre(t)$) or from the intermediate marking to the target marking M_2 (resp M_2'). For transitions with decreased degree in the intermediate marking, we have $enab1_{temp}(t) = enab(t) \setminus max(enab(t))$, and for other transitions $enab1_{temp}(t) = enab(t)$ (ans similarly for $enab_1', enab'1_{temp}$). For transitions which degree increases w.r.t temporary marking, we have $enab_2(t) = enab1_{temp} \uplus \{0\}$, and for remaining ones, $enab_2(t) = enab1_{temp}$. This applies similarly for $enab'_2(t)$ w.r.t $enab'1_{tmp}$. In $enab_2(t)$, the value of clocks is identical for all transitions from the same cluster, and hence in particular for pruned transitions. Hence, after firing a transition t, $enab'_2(t)$ is still a restriction of $enab_2(t)$ to unpruned transitions, and $C_2 = \mathcal{R}(C'_2)$ (contradiction).

Hence, $\mathcal{R}$ is a bijective relation from $Reach(\mathcal{N}')$ to $Reach(\mathcal{N})$ that preserves moves, i.e. it is an isomorphism from $\big(Reach(\mathcal{N}, C_0), \longrightarrow_{\mathcal{N}}\big)$ to
$\big(Reach(\mathcal{N}', C_0), \longrightarrow_{\mathcal{N}'}\big)$. □

An immediate consequence of Lemma 1 is that $Lang(\mathcal{N}) = Lang(Prune(\mathcal{N}))$. Note that this lemma holds only under the free-choice assumption. Indeed, in a standard TPN, one can disable the most urgent transition from a cluster without disabling other instances of transitions in this cluster. In the example of Figure 7, for instance, transition t_2 and t_3 belong to the same cluster, and pruning would remove t_3. However, in the unpruned net, transition t_2 can be disabled by firing of t_1, which allows t_3 to fire later. Hence, for this non-FC-TPN, $Lang(\mathcal{N}) \neq Lang(Prune(\mathcal{N}))$.

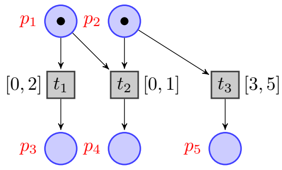

Figure 7: Pruning only works for FC-TPNs

4.2. Simulating runs in the timed and untimed FC-TPN

In this section, we prove our main theorem, which relates time causal processes of pruned FC-TPNs with untimed causal processes of their untimed nets. In the next section, we will use this theorem to show decidability of the firability and termination problems.

Theorem 1 (Inclusion of untimed prefixes). *Let $\mathcal{N} = (\mathcal{U}, M_0, I)$ be a pruned FC-TPN (without forced 0-delay time firing sequences) and let $\mathcal{U}' = Untime(\mathcal{N})$. Let U' be an (untimed) causal process of $\mathcal{U}'$. Then there exists a time causal process N of $\mathcal{N}$ such that $U' \leq Untime(N)$.*

Proof: We are given U' a causal process of $\mathcal{U}'$. We will iteratively construct a pair ρ_i, σ_i, where ρ_i is a causal process of $\mathcal{U}'$, σ_i a time causal process of $\mathcal{N}$, and such that ρ_i is a prefix of $Untime(\sigma_i)$. The construction ends at some $n \in \mathbb{N}$ such that $\rho_n = U'$. At that stage of the algorithm, σ_n is a time causal process such that $U' \leq Untime(\sigma_n)$ which is the desired result. For this, we maintain the following invariants at every intermediate step, i.e., for all $0 \leq i \leq n$:

(I1) ρ_i is a prefix of U' and

(I2) ρ_i is a prefix of $Untime(\sigma_i)$

(I3) either $|\rho_i| + 1 = |\rho_{i+1}|$ or $|\sigma_i| + 1 = |\sigma_{i+1}|$

(I4) $e \in posb(\rho_i)$ and $e \notin \sigma_i$ implies that $e \in posb(\sigma_i)$

(I5) $eG_i^+e' \Rightarrow \tau(e) \leq \tau(e')$, (with G_i^+ the flow relation of σ_i).

The first two conditions have been explained above. Condition $(I3)$ ensures that the algorithm progresses at every iteration, either in ρ_i or σ_i. Condition $(I4)$ says that if an event e is enabled in the untimed causal process ρ_i and has not yet been fired in σ_i, then it must be enabled at σ_i. Note that due to urgency, it might be the case that e is not yet firable in σ_i. Finally, Condition $(I5)$ ensures that the time stamps that we add in σ_i are consistent with its causal order.

We start by defining, for a time causal process ON, the maximal firing date for an event $e = (X, t) \in posb(ON)$ as $mfd(ON, e) = \max_{x \in X}(\tau(x)) + lft(t)$. This represents the maximal date that can be attached to event $e = (X, t)$ when it is appended to ON. Further, we use $td(ON, e)$ to denote the difference between $mfd(ON, e)$ and the maximal date attached to an event in ON. Note that this value can be negative if the maximal date in ON is greater than $mfd(ON, e)$, i.e., e has already been fired in ON. This represents the time that has to elapse before event e becomes urgent, or that has elapsed after e. Finally, for a (time) process ON and an event e, we denote by $ON \cup \{e\}$ the (time) process obtained by appending event e to ON, when it is possible to do so. Algorithm 1 gives a procedure that starts from an untimed causal process U' and computes time-stamps and additional events needed to obtain a timed causal process N'.

Algorithm 1: Causal-net-to-time-causal-net

Input:

- $\mathcal{N} = (\mathcal{U}, M_0, I)$ a pruned FC-TPN without 0-delay firing sequences,
- $U' = (B', E', G')$ a (untimed) causal process of $\mathcal{U}' = Untime(\mathcal{N})$

Output: N a timed causal process of $\mathcal{N}$ such that $U' \leq Untime(N)$

`// For all` i`,` ρ_i `is a untimed causal process`

`//` σ_i `is a timed causal process`

1 Initializations:
2 $CLK := 0, \rho_0 := (B_0, \emptyset, \emptyset), \sigma_0 := (B_0, \emptyset, \emptyset, \emptyset)$;
3 **while** $\rho_i \neq U'$ **do**
4   Choose an event $e = (X_e, t_e)$ from $posb(\rho_i) \cap U'$;
5   **if** $e \in \sigma_i$ **then**
6     $\rho_{i+1} \leftarrow \rho_i \cup \{e\}$; $\sigma_{i+1} \leftarrow \sigma_i$;
7   **else**
8     $min = \min_{e' \in posb(\sigma_i)} mfd(\sigma_i, e')$;
9     $S_i = \{e' \in posb(\sigma_i) \mid mfd(\sigma_i, e') = min \text{ and } min \neq \infty\}$;
10   **if** $S' = S_i \cap E' \neq \emptyset$ **then**
11   Pick an event $e_t = (X, t)$ from S';
12   **else**
13   **if** $S_i = \emptyset$ **then**
14         $CLK' = CLK + max\{0, td(\sigma_i, e)\}$;
15         $\tau(e) = CLK'$;
16         $\sigma_{i+1} \leftarrow \sigma_i \cup \{e\}$; $\rho_{i+1} \leftarrow \rho_i$;
17         $GOTO$ Step-23;
18   **else**
19   Pick an event $e_t = (X, t)$ from S_i;
20     $CLK' = CLK + lft(t) - td(\sigma_i, e_t)$;
21     $\tau(e_t) = CLK'$ `// adding time stamp`
22     $\sigma_{i+1} \leftarrow \sigma_i \cup \{e_t\}$; $\rho_{i+1} \leftarrow \rho_i$;
23   $i \leftarrow i + 1$;

Let us now explain the algorithm. At initialization, B_0 is the set of conditions corresponding to marked places in M_0, and CLK, a real valued variable which stores the time-elapsed till now, is set to 0. ρ_i and σ_i are respectively, untimed and time causal processes at i-th iteration. S_i is the set of events that are the most urgent instances of transitions in σ_i.

The idea is that, at each iteration, we pick (in line 4) an event e enabled in the current ρ_i, which would grow ρ_i to eventually reach the untimed causal process U'. If this event has already been fired in σ_i, then we just add it to ρ_i and go to the next iteration.

Else, we try to fire it in σ_i. However, to do so, we first compute S_i the set of all urgent transitions in σ_i (line 8–9). If there is an urgent transition instance e_t whose corresponding event is also in U', then we pick it (in line 11) and fire it, i.e., add it to σ_i and update time information correctly (line 20-23). This guarantees that σ_i has grown at this iteration so we go the next iteration. On the other hand, if there is no urgent transition in S_i which is also in U', we check if there is no urgent transition at all, i.e., S_i is empty. In this case, we elapse time till e can fire and fire it as soon as possible (line 14), updating clocks appropriately.

Finally, if there is some urgent transition in S_i but this transition is not in U', then we fire it as late as possible (line 20–23). The fact that this does not change the enabling of e (due to conflicts) is proved by our invariant $I4$.

If invariants $I1, I2, I3, I4$ and $I5$ are satisfied for all iterations then Algorithm 1 is correct.

First, it is easy to see that invariants $I1, I2$ are preserved. They hold at the beginning and if we assume that they hold at the end of i-th iteration of while loop, then in the $(i+1)$-th iteration, we have: if we exit the iteration in step 22 then it means that $\rho_{i+1} = \rho_i$ and $\sigma_{i+1} = \sigma_i \cup \{e\}$ and by induction hypothesis, we have $\rho_i \le U'$, and $\rho_i \le \sigma_i$. Hence $\rho_{i+1} \le U'$ and $\rho_i = \rho_{i+1} \le \sigma_i \le \sigma_{i+1}$. Otherwise we have exited the iteration in step 6 and it means that $\sigma_{i+1} = \sigma_i$ and $\rho_{i+1} = \rho_i \cup \{e\}$. So we have $e \in posb(\rho_i)$ and $e \in U' \setminus \rho_i$ and $\sigma_i = \sigma_{i-1} \cup \{e\}$. Hence $\rho_{i+1} \le U'$ and $\rho_i \cup \{e\} \le \sigma_i \le \sigma_{i+1}$. Similarly, assuming $I4$ holds at the previous iteration, it is easy to see that $I3$ will hold at each iteration as either ρ_i or σ_i grows. And we can also check that our time-stamps are indeed consistent with the causality imposed by the flow relation.

Lemma 2. *In Algorithm 1, all invariants are preserved at end of each iteration of while loop.*

Proof: Proof is given by induction on number of iterations. We note that each iteration must end either in 6, 17 or 22 (before incrementing i and going to the next iteration).

(I1) **(ρ_i is a prefix of U')** It is true at the beginning of loop i.e., base step of $i = 0$. Assume that it is true at the end of i^{th} iteration of the while loop. Now we have to prove that this holds at the end of the $(i+1)^{th}$ iteration. If we exit the iteration in step 17 or step 22, then it means that $\rho_{i+1} = \rho_i$ and by induction hypothesis, we have $\rho_i \le U'$ hence $\rho_{i+1} \le U'$. Otherwise we have exited the iteration in step 6 and it means that $\sigma_{i+1} = \sigma_i$ and $\rho_{i+1} = \rho_i \cup \{e\}$. So we have $e \in posb(\rho_i)$ and $e \in U' \setminus \rho_i$. Hence $\rho_{i+1} \le U'$.

(I2) **(ρ_i is a prefix of *Untime*(σ_i))** It is true at the beginning of loop i.e., base step of $i = 0$, since $\rho_0 = \sigma_0$. Assume that is true at the end of i^{th} iteration of the while loop. Now we have to prove that this holds at the end of the $(i+1)^{th}$ iteration. If we exit the iteration in step 17 or step 22 then it means that $\rho_{i+1} = \rho_i$ and $\sigma_{i+1} = \sigma_i \cup \{e_i\}$ for $e_i = e$ or $e_i = e_t$. By induction hypothesis we have that $\rho_i \le Untime(\sigma_i)$. Hence we have $\rho_i = \rho_{i+1} \le \sigma_i \le \sigma_{i+1}$. Otherwise we have exited the iteration in step 6 and it means that $\sigma_{i+1} = \sigma_i$ and $\rho_{i+1} = \rho_i \cup \{e\}$. So we had $\sigma_i = \sigma_{i-1} \cup \{e\}$ in the previous (i.e., i-th) iteration (or even before that!) and therefore $\rho_i \cup \{e\} \le Untime(\sigma_i) \le Untime(\sigma_{i+1})$.

(I3) $|\rho_i| + 1 = |\rho_{i+1}|$ **or** $|\sigma_i| + 1 = |\sigma_{i+1}|$ At the beginning of loop i.e., at $i = 0$, we have $\rho_0 = \sigma_0$, and $|\rho_0| = |\sigma_0| = 0$. At the first iteration, we necessarilty begin the loop discovering that any event in $posb(\rho_0) \notin \sigma_0$. Hence, an event is appended to σ_0 to obtain σ_1 and $\rho_1 = \rho_0$. So invariant $I3$ holds for $i = 0$. Assume that $I3$ holds up to i-th iteration of while loop. Now we have to prove that it holds at the end of $(i+1)^{th}$ iteration. If we exit the iteration in step 17 or step 22 then it means that $\rho_{i+1} = \rho_i$ and $\sigma_{i+1} = \sigma_i \cup \{e_i\}$ for $e_i = e$ or $e_i = e_t$. To perform this step of growing σ_{i+1} we need to add the event e corresponding to an instance of some transition in $\mathcal{N}$. This transition must be enabled at σ_i, which is guaranteed by Claim 23 given in $I4$ (proof given below). Hence we have that $|\sigma_i|+1 = |\sigma_{i+1}|$. Otherwise we have exited the iteration in step 6 and this means that $\sigma_{i+1} = \sigma_i$ and $\rho_{i+1} = \rho_i \cup \{e\}$. Therefore we have that $|\rho_i| + 1 = |\rho_{i+1}|$.

(I4) For all events $e \in posb(\rho_i)$ **and** $e \notin \sigma_i$**, implies** $e \in posb(\sigma_i)$**.** Again, recall that for ease of writing, we slightly abuse notation here; when we say that an event of an untimed causal process belongs (or does not) to a time causal process, we implicitly mean that it belongs to the *untiming* of the time causal process and so on.

The invariant holds at the beginning of loop i.e., at base step $i = 0$. Indeed, all enabled transitions in marking M_0 correspond to transitions enabled in the initial configuration of $\mathcal{N}$, and as $\mathcal{N}$ is pruned, all of them can fire if a sufficiently large delay elapses. Assume that it is true at the end of i^{th} iteration of while loop. Now we have to prove that this holds at the end of the $(i+1)^{th}$ iteration. So we enter the $i+1^{th}$ iteration with i, ρ_i, σ_i and exit with values $i+1, \rho_{i+1}, \sigma_{i+1}$. Let $I4^{i+1}$ denote the invariant $I4$ at the $i+1^{th}$ iteration and $I4^i$ denote the invariant $I4$ at the i^{th} iteration, which are explicitly stated below:

$$I4^{i+1} : f \in posb(\rho_{i+1}) \text{ and } f \notin \sigma_{i+1} \implies f \in posb(\sigma_{i+1}) \; I4^i : f \in posb(\rho_i) \text{ and } f \notin \sigma_i \implies f \in posb(\sigma_i).$$

Let f be some event of $posb(\rho_{i+1})$ (we use f as a generic event to not confuse with e, the event picked in step 4). We have two cases:

– *Case(exit by step 6).* This means that $\sigma_{i+1} = \sigma_i$ and $\rho_{i+1} = \rho_i \cup \{e_i\}$ for some event e_i. To execute this step, $e_i \in posb(\rho_i)$ is a pre-requisite, and $e_i \notin posb(\rho_{i+1})$. Hence $f \neq e_i$. Then either f was newly enabled after adding e_i or f was already enabled before adding e_i.

- First consider the case in which f was newly enabled after adding e_i to ρ_i. Assume that it satisfies premises of $I4^{i+1}$. Now we have to prove that $f \in posb(\sigma_{i+1})$ to prove that invariant $I4$ is preserved. Since f was enabled at ρ_{i+1} and not at ρ_i, event $f \notin posb(\rho_i)$. Therefore, it trivially satisfies property $I4^i$. Hence, we get that $f \in posb(\sigma_i)$. Now as i^{th} iteration of the while loop exited at step 6, we have $\sigma_{i+1} = \sigma_i$ and hence $posb(\sigma_{i+1}) = posb(\sigma_i)$. Therefore we obtain $f \in posb(\sigma_{i+1})$.
- Now consider the case in which f was already enabled before adding e_i to ρ_i. Assume that it satisfies premise of $I4^{i+1}$, i.e. $f \notin \sigma_{i+1}$. Now we have to prove that $f \in posb(\sigma_{i+1})$. Since it satisfies premises of $I4^{i+1}$ we have $f \notin \sigma_{i+1}$, implying $f \notin \sigma_i$. Now, as $I4$ holds up to step i, i.e. $I4^i$ holds, and as we know that $f \in posb(\rho_i)$ we get that $f \in posb(\sigma_i)$. Now when we exit at step 6, we have that $\sigma_{i+1} = \sigma_i$ and hence $posb(\sigma_{i+1}) = posb(\sigma_i)$. Therefore we get that $f \in posb(\sigma_{i+1})$.

– *Case(exit by step 22 or step 17)*: If we exit the iteration in either step, then it means that $\rho_{i+1} = \rho_i$ and $\sigma_{i+1} = \sigma_i \cup \{e_i\}$ for some event e_i (in case of exit by step 17, $e_i = e$ of that iteration). In both cases, there are two cases to consider: $f = e_i$ or $f \neq e_i$, i.e., whether f is the event which is used to grow σ_i to get σ_{i+1} or not.

- First, we prove this invariant for $f \neq e_i$ satisfying premise of $I4^{i+1}$. After adding e_i to σ_i, we cannot have event f newly enabled i.e., $f \in posb(\rho_{i+1})$ and $f \notin posb(\rho_i)$ because $\rho_i = \rho_{i+1}$. So we consider the case when f was already enabled i.e., $f \in posb(\rho_{i+1})$ and $f \in posb(\rho_i)$ also. Since f satisfies premise of $I4^{i+1}$ we have that $f \notin \sigma_{i+1}$, implying $f \notin \sigma_i$ as $\sigma_i \leq \sigma_{i+1}$. So we have both clauses of $I4^i$ satisfied for event f. Therefore we have that $f \in posb(\sigma_i)$. Let t be the transition corresponding to event e_i and let s be the transition corresponding to event f. Let $M(\sigma_i)$ denote the marking reached after execution of σ_i. Let t_u and s_u be the most urgent transition instances of t and s respectively at $M(\sigma_i)$. Both these transition instances were firable at marking $M(\sigma_i)$. So when σ_{i+1} was obtained by adding event e_i to σ_i, correspondingly we have $M(\sigma_i) \xrightarrow{t} M(\sigma_{i+1})$. Now if s and t were independent–i.e., have no common place in their presets– in the net (and it does not matter if s_u and t_u were equally urgent) transition s_u is still enabled at marking $M(\sigma_{i+1})$ and firable, implying that its event $f \in posb(\sigma_{i+1})$ which was what we wanted to prove. Otherwise s and t are in conflict–i.e., have a common place in their preset– in the net, and both s_u and t_u were enabled at marking $M(\sigma_i)$, which means that both the corresponding events e_i and f were in conflict too. But $\rho_i \leq N'$, and we had e_i and f from N' and in $posb(\rho_i)$, so we have $\rho_i \cup \{f, e'\} \leq N'$, but N' being a causal net, we could have only one of these events in it. So this case never arises.

– Now, on the other hand, if $f = e_i$, event e_i is added to σ_i, then we have $e \in \sigma_{i+1}$ which falsifies a clause in the premise of $I4^{i+1}$ hence trivially satisfying it. But for this, we need to crucially show that we can indeed add e_i, i.e., we can fire t the transition corresponding to event e_i in the net $\mathcal{N}$ (it has not got disabled by firing of any other urgent transitions). We show this below which completes the proof.

In the untimed net, at a marking, multiple enabled instances of a transition are indistinguishable from each other. One such instance of transition t gave rise to event e_i in the untimed causal net. To get the matching event in the timed causal net, it is sufficient to fire some instance of t in the timed net. Since transition instances of same transition are not in conflict with each other, and by FEFF policy, all we need to prove is that the first enabled transition instance of t is firable at $M(\sigma_i)$. Let t_u be this transition instance.

Claim A. Transition t_u it is firable at marking $M(\sigma_i)$
Proof: Assume the contrary, i.e., transition instance t_u is not firable at $M(\sigma_i)$. Inductively, let us assume that this is the first time that such transition which was not firable, while building time causal net N, from the beginning where initial conditions were same and set of enabled events same for both causal nets N' and N. Let $e_i \bigtriangledown$ denote the set of predecessors for e_i in causal net N'; it contains all events and conditions needed for event e_i to occur. As we are in $i+1$-th iteration, $e_i \in posb(\rho_i)$ and since we are exiting this iteration by step 22 or 17, it means $e_i \notin \sigma_i$. By induction we have invariant $I2$ true at i-th iteration so $\rho_i \leq \sigma_i$, implying $e_i \notin \rho_i$, so $e_i \bigtriangledown \setminus \{e_i\} \leq \rho_i$, which gives us $e_i \bigtriangledown \setminus \{e_i\} \leq \sigma_i$. Therefore all the places in the preset ${}^\bullet t$ are marked at the configuration with marking component $M(\sigma_i)$. Now the only possible reason for which transition instance t_u is not firable in the timed net at this configuration, is that there exists some instance t'_u of another transition t' which shares a pre-place with t (and hence by FEFD policy, transition instance t'_u is in conflict with t_u), and more urgent than it, i.e., at $M(\sigma_i)$, we have $lft(t') - v(t'_u) < lft(t) - v(t_u)$. Note that if ${}^\bullet t \cap {}^\bullet t' = \emptyset$ and $lft(t') - v(t'_u) < lft(t) - v(t_u)$, then we could have fired t'_u first and then t_u; so we can assume that all such transitions are fired before. As the net is free choice we have ${}^\bullet t = {}^\bullet t'$ and therefore, transition t_u is enabled at marking $M(\sigma_i)$. Since $\mathcal{N}$ is a pruned net, the cluster $[t]$ is pruned, and so $lft(t) = lft(t')$, which is a contradiction even in the case when $lft(t) = \infty$. So it must be the case that $e_i \in posb(\sigma_i)$ in time causal net N. Note that this step of the proof essentially relies on the assumptions that $\mathcal{N}$ is free-choice and pruned, and also uses the particularities of the multi-server semantics. □

(I5) $eG_i^+e' \implies \tau(e) \leq \tau(e')$. In the base case of $i = 0$, timed causal net σ_0 has no events and so the invariant is trivially satisfied. Assume that induction holds up to the $(i-1)$-th iteration. That is when we enter with σ_{i-1} and ρ_{i-1} in the loop and come out with σ_i and ρ_i. So we have

$$(I5)^i : \ eG_i^+e' \implies \tau(e) \leq \tau(e')$$

Now for the inductive step, assume that we are in i^{th} iteration and eG_{i+1}^+e'. If events e and e' both are in σ_i then by $(I5)^i$ we have that $\tau(e) \leq \tau(e')$. Let us now consider cases where e and e' are not both in σ_i. Adding an event $e = (X, t)$ to an existing causal net means creating causal dependencies from predecessors of conditions in X to e. Since we add at most one event to σ at each iteration of the loop, and since we have eG_{i+1}^+e' it cannot be the case that $e \in \sigma_{i+1} \setminus \sigma_i$ and $e' \in \sigma_i \cap \sigma_{i+1}$. Therefore, $e' \in \sigma_{i+1} \setminus \sigma_i$ and $e \in \sigma_i \cap \sigma_{i+1}$. But eG_{i+1}^+e' also implies that $e \in e' \bigtriangledown$ which means event e was added at some iteration k where $k \leq i$. When an event $e = (X, t)$ is appended to a timed causal net σ_k, it gets a time stamp $\tau(e) = CLK + lft(t_e) - td(\sigma_k, e)$ or $t(e) = CLK + max\{0, td(\sigma_i, e)\}$. Now, it is easy to see that during the execution of the algorithm, the value of CLK only increases. Hence, we have $\tau(e') \geq \tau(e)$.

□

The last thing that is left to prove is that Algorithm 1 terminates for any input. We show that σ cannot grow unboundedly. Each step of the algorithm adds either an event to ρ_i, or to σ_i. While the number of steps that add events to ρ_i is finite, the algorithm may still not terminate if σ can grow unboundedly, i.e. the while loop is exited at line 22 an unbounded number of times without progress in ρ. Now, the crux of the proof is that at every step, a set of events from U' are listed as possible events. At every step, since we do not have forced 0-delay firing sequences, and since we choose event dates that maximize time progress, these events can not remain ignored forever. A complete proof follows.

Lemma 3. *Algorithm 1 terminates in finitely many steps.*

Proof: We prove that eventually $\rho_i = U'$ for some i. For that it is sufficient to prove that we do not exit by step 22 or by step 17 infinitely often in the while loop. If this loop is executed a finite number of times then it means that one exists the loop with $\rho_i = U'$, which is a finite process.

Let $clk = \theta$, and let e be the event chosen in step 4 at the beginning of while loop at the $i+1^{th}$ iteration and let t be the transition corresponding to this event. Suppose that we exit by step 22 infinitely often. Let us assume that last time we exited from step 6 was at the end of i^{th} iteration. If we exit by this step then this also means that $S_i \neq \emptyset$ which means $lft(t) \neq \infty$. So it implies that we have an infinite run $q_1 \xrightarrow{t_1} q_1 \xrightarrow{t_1} \dots$ starting at q_1, adding events to σ_i. Since net $\mathcal{N}$ is finite, the set of transitions occuring in this run is finite. So there exist at least one transition which occurs infinitely often in this run. Let s be this transition. Now if $lft(s) > 0$ then there exist a minimal constant K such that $\sum_{k=0}^{K} lft(s) > lft(t)$. Therefore, before the K^{th} occurence of s in this infinite sequence, transition t would have become urgent and we would have fired it, contradicting the fact that there is no transition in the sequence which contributes an event of ρ_i. So it must be the case that $lft(s) = 0$, and this is true of any transition occuring infinitely often in this infinite sequence. Now there exist $m \geq 1$ such that each transition occurring after q_m occurs infinitely often. It means that we have a 0-delay timed firing sequence starting q_m, which is a contradiction.

Now to complete the proof by contradiction, we suppose that, we exit by Step 17 infinitely often, which is not possible because events like e are taken from untimed causal net U' which is finite. $\square$

The correctness proof and termination lemma allow to conclude the proof of Theorem 1.$\square$

We can now extend Theorem 1 from causal nets to words. Let $\mathcal{U} = (P, T, F)$ be an untimed Petri net. Given a causal process $U = (ON = (B, E, G), \pi)$ of $\mathcal{U}$, consider the partial order $\preceq = G^*$. We denote by $lin(U)$, the set of words over alphabet of transitions obtained by considering linearizations of the partial order of G^* and projecting onto the labeling alphabet (of transitions T), i.e., $lin(N) = \{\rho \in T^* \mid$ there exists ρ' a linearization of G^* such that $\pi|_E(\rho') = \rho\}$.

Now, recall that given an untimed net $\mathcal{U}$, its language $Lang(\mathcal{U}, M_0)$ is a set of firing sequences, i.e., words over the alphabet of transitions. Now, we obtain our characterization for words rather than causal processes.

Corollary 1 (Words version). *Let $\mathcal{N}$ be a pruned FC-TPN (without 0-delay time firing sequences) and let $\mathcal{U}' = Untime(\mathcal{N})$. Then, for each $w \in Lang(\mathcal{U}', M_0)$ there exists a time causal process N of $\mathcal{N}$ and $w' \in Lang(Untime(N))$ such that $w \preceq w'$.*

Proof: Given a word $w \in Lang(\mathcal{U}', M_0)$, using Proposition 2, we get an untimed causal process U' of net $\mathcal{U}'$, such that $w \in lin(U')$. By Theorem 1, we get a timed causal process N of net $\mathcal{N}$ such that $U' \leq Untime(N)$. Now since $w \in lin(U')$, and $U' \leq Untime(N)$, we can extend w to $w' \in lin(Untime(N))$. $\square$

5. Decidability results for FC-TPNs

In this section, we will show two significant consequences of the connections between processes of $\mathcal{N}$ and $Untime(\mathcal{N})$ shown in Theorem 1. Namely, we show that both firability and termination can be decided for FC-TPNs with no 0-delay time firing sequences.

Theorem 2 (Firability). *Given an FC-TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$ without 0-delay time firing sequences, and a transition $t \in T$, checking firability of transition t in $\mathcal{N}$ is decidable.*

Proof: Given a FC-TPN $\mathcal{N}$, one can compute an equivalent pruned version $\mathcal{N}_{Pruned}$, i.e. a Petri nets with the same timed language and the same set of processes, but which clusters have only firable transitions. One can compute a Petri net $\mathcal{U}' = \mathit{Untime}(\mathcal{N}_{Pruned})$. For every PN, it is well known that coverability of a marking is decidable [21]. A particular transition t is firable in a Petri net $\mathcal{U}$ iff its preset ${}^{\bullet}t$ is coverable. Coverability can be obtained by construction of a coverability tree, or using backward algorithms (see for instance [22] for recent algorithms). In both cases, one can exhibit a sequence of transitions witnessing coverability of a particular marking. If $w = t_0. \dots .t_k$ is such a sequence witnessing coverability of ${}^{\bullet}t$ from M_0 in $\mathit{Untime}(\mathcal{N}_{Pruned})$, then one can immediately infer that $w.t \in \mathit{Lang}(\mathcal{U}')$, and hence t is firable in $\mathcal{U}'$. Using Corollary 1, there exists a timed word $v = (t_0, d_0) \dots (t_k, d_k).(t, d_k + 1) \in \mathit{Lang}(\mathcal{N}_{Pruned})$, and hence $v \in \mathit{Lang}(\mathcal{N})$ and t is firable in $\mathcal{N}$. Conversely, assume that t is not firable in $\mathit{Untime}(\mathcal{N})$ and that t it is firable in $\mathcal{N}$. Then there exists a run ρ of $\mathcal{N}$ firing t. Then, $\mathit{Untime}(\rho)$ is a run of $\mathit{Untime}(\mathcal{N})$ which fires t, which is a contradiction. □

Next we turn to the problem of termination and show that for FC-TPNs without 0-delay time firing sequences, this too is decidable.

Theorem 3 (Termination). *Given an FC-TPN $\mathcal{N}$ (without 0-delay time firing sequences), it is decidable to check if $\mathcal{N}$ terminates.*

Proof: Let $\mathcal{N}$ be a FC-TPN and let $\mathcal{N}_p$ be its pruned version. Since the reachability graphs of $\mathcal{N}$ and $\mathcal{N}_p$ are isomorphic by (Pruning) Lemma 1, it is sufficient to decide if $\mathcal{N}_p$ has only terminating runs. Let $\mathcal{N}' = \mathit{Untime}(\mathcal{N}_p)$. If $\mathcal{N}'$ does not terminate, then it allows sequences of transitions that are unboundedly long. As we know from Corollary 1 that for every word w' of $\mathit{Lang}(\mathcal{N}')$, there exists a timed word w of $\mathit{Lang}(\mathcal{N}_p)$ of length $|w| \geq |w'|$, then $\mathcal{N}_p$ (and hence $\mathcal{N}$) allows sequences of transitions of unbounded lengths, i.e., it does not terminate either. In the other direction, if $\mathcal{N}_p$ has an infinite run, as time constraints in free choice TPNs can only prevent occurrence of a transition, then the untimed net clearly has an infinite run too. Thus, we have reduced the problem to termination of an untimed Petri net, which is decidable by the standard finite reachability tree construction. □

Thus, using the proof technique we developed in the last section, we were able to easily tackle termination and firability for FC-TPNs as shown above. However, this technique of using the relation between untimed processes and processes of untimed nets does not allow us to tackle any property. Further discussion about properties that cannot be tackled directly is defered to Section 7.

6. Robustness for FC-TPNs

As mentioned in the introduction, robustness problems address the question of preservation of properties of systems that are subject to imprecision of time measurement. In this section, for any property or decision problem of TPNs, e.g., firability, termination, we define formally what we mean by robustness for this property under imprecision. Then, we focus on two particular properties, namely, firability and termination and show decidability results in this setting. To the best of our knowledge, this is the first decidability result of robustness for an unbounded class of TPNs allowing urgency.

Let us now formally define the notion of robustness considered in this paper. Inspired by the definitions for robustness in timed automata [5], we define two notions of robustness with respect to guard enlargement and shrinking:

- Given an interval $I(t) = [a, b]$ for a transition t, and $\Delta \in \mathbb{Q}_{>0}$, the *enlargement* of $I(t)$ by Δ is the interval $I_\Delta(t) = [\max(0, a-\Delta), b+\Delta]$. For a TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$, the *enlargement* of $\mathcal{N}$ by $\Delta \in \mathbb{Q}_{>0}$, is the net $\mathcal{N}_\Delta = (\mathcal{U}, M_0, I_\Delta)$, obtained from $(\mathcal{U}, M_0, I)$ by replacing every interval $I(t)$ by its enlarged version $I(t)_\Delta$.

- Given an interval $I(t)$ for transition t, and $\triangledown \in \mathbb{Q}_{>0}$, the *shrinking* of $I(t)$ by $\triangledown$, denoted $I_\triangledown(t)$, is the interval obtained by replacing its lower bound $eft(t)$ by the bound $eft(t) + \triangledown$, and its upper bound $lft(t)$ by the bound $max\{0, lft(t) - \triangledown\}$. Given a TPN $\mathcal{N} = (\mathcal{U}, M_0, I)$, the *shrinking* of $\mathcal{N}$ by $\triangledown \in \mathbb{Q}_{>0}$ is the net $\mathcal{N}_\triangledown$, obtained by replacing interval $I(t)$ of each transition t, by the interval $I_\triangledown(t)$.

With a slight abuse of notation, in this article Δ will always mean enlargement (by Δ) and $\triangledown$ will be used for shrinking (again by the value $\triangledown$). The first robustness question, is for a chosen enlargement $\Delta \in \mathbb{Q}_{>0}$ (resp. shrinking $\triangledown \in \mathbb{Q}_{>0}$), to decide whether, properties of a net (reachability, boundedness, coverability) or its (untimed) language are preserved in $\mathcal{N}_\Delta$ (resp. $\mathcal{N}_\triangledown$). A more general robustness question consists of deciding whether there exists a value of $\Delta \in \mathbb{Q}_{>0}$ (resp. $\triangledown \in \mathbb{Q}_{>0}$) such that $\mathcal{N}_\Delta$ (resp. $\mathcal{N}_\triangledown$) preserves reachability, coverability, or the (untimed) language of $\mathcal{N}$. A positive answer to this question means that slightly changing guards preserves the considered property, i.e. this property of the system is robust to a small time imprecision. In general, robustness problems (including checking firability of a transition and termination of the net) are undecidable for TPNs, as shown in [6], and become decidable when the considered nets are bounded. An interesting question is whether robustness of some of the above mentioned problems is decidable for TPNs with multiple enabling outside bounded classes of nets. Answering this question would provide useful tools to check properties of systems made of bounded timed processes communicating through bag channels [23].

In this article we focus on both variants of robustness (w.r.t. guard enlargement and shrinking), for two particular properties/problems, namely, robustness of firability and robustness of termination and show decidability results for them in the next two subsections.

6.1. Robustness of firability for FC-TPNs

A natural question in Petri nets is whether some transition is firable or not. When a Petri net models the control flow of a program, this amounts to asking whether some instruction can be executed. The firability robustness problem can be defined both for guard enlargement and guard shrinking.

Definition 11 (Firability robustness problems). *Given an FC-TPN $\mathcal{N}$:*

- **(robustness w.r.t. guard enlargement)** *does there exist an enlargement parameter $\Delta \in \mathbb{Q}_{>0}$ such that Firable($\mathcal{N}$) = Firable($\mathcal{N}_\Delta$)?*
- **(robustness w.r.t. guard shrinking)** *does there exist a shrinking parameter $\triangledown \in \mathbb{Q}_{>0}$ such that Firable($\mathcal{N}$) = Firable($\mathcal{N}_\triangledown$)?*

A net is said to be *robustly firable* w.r.t guard enlargement (resp. guard shrinking), if there exists such a Δ (resp. $\triangledown$).

6.1.1. Robustness of firability with respect to guard enlargement

We start by observing that even in FC-TPNs, firability is not a robust property w.r.t. enlargement i.e., it is not the case that there is always a (non-zero) guard enlargement that preserves firability. Consider, for instance, the FC-TPN $\mathcal{N} = (\mathcal{U}, M_0 = \{p_1\}, I)$ given in Figure 8. Under any perturbation $\Delta > 0$, transition b becomes firable, and hence *Firable*($\mathcal{N}$) $\neq$ *Firable*($\mathcal{N}_\Delta$). The reachable markings of $\mathcal{N}$ are $Untime(Reach(\mathcal{N}, M_0)) = \{\{p_1\}, \{p_2\}\}$. Under guard enlargement $\Delta > 0$, FC-TPN $\mathcal{N}_\Delta$ has set of reachable markings $Untime(Reach(\mathcal{N}_\Delta, M_0)) = \{\{p_1\}, \{p_2\}, \{p_3\}\}$.

However, as checking firability of every transition in a FC-TPN is decidable, given a fixed enlargement Δ, one can decide whether the firability set of $\mathcal{N}$ differs from that of $\mathcal{N}_\Delta$. So the next question is to decide for a given FC-TPN $\mathcal{N}$, whether there exists a $\Delta > 0$ such that enlarging the guards of $\mathcal{N}$ by Δ preserves firability.

Theorem 4. *Let $\mathcal{N}$ be a FC-TPN without 0-delay sequences. Then checking whether $\mathcal{N}$ is robustly firable w.r.t. guard enlargement is decidable. If $\mathcal{N}$ is robustly firable, one can also effectively compute a value Δ such that Firable($\mathcal{N}$) = Firable($\mathcal{N}_\Delta$).*

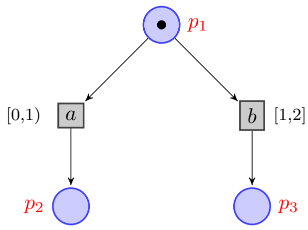

Figure 8: A FC-TPN for which fireability is not robust w.r.t. guard enlargement

Proof: We first observe that in a pruned FC-TPN $\mathcal{N}$, a transition t is firable iff all transitions in t's cluster are firable. Next, all timed words of $\mathcal{N}$ are also timed words of $\mathcal{N}_\Delta$. So, enlarging a net can only result in new behaviors. As a consequence, if a transition t is firable in $\mathcal{N}$, it is necessarily firable in $\mathcal{N}_\Delta$. Further, note that as they have the same underlying untimed net, $\mathcal{N}$ and $\mathcal{N}_\Delta$ are both free-choice and they have the same clusters. Now the pruning operation applied to each of these nets results in removing transitions (it can never add transitions) and pruning lemma holds for both of them. If after the pruning, we still have the same clusters, then the set of fireable transitions remains the same. The only way to have an extra transition that can fire in $Prune(\mathcal{N}_\Delta)$ is if this transition t in cluster C has been removed in $Prune(\mathcal{N})$ but remains in $Prune(\mathcal{N}_\Delta)$ due to enlargement. By our pruning construction this means that there must exist another fireable transition in this cluster in $Prune(\mathcal{N})$ (else we would not remove t). That is,

Claim: For any $\Delta > 0$, $\mathit{Firable}(\mathcal{N}) \neq \mathit{Firable}(\mathcal{N}_\Delta)$ iff there exists a cluster C_Δ of $Prune(\mathcal{N}_\Delta)$ such that $C \subset C_\Delta$ for some cluster C of $Prune(\mathcal{N})$, and at least one transition t of C is firable.

Proof: [of claim] Let us suppose that there exists a cluster C of $Prune(\mathcal{N})$ that contains transitions $t_1, \dots, t_k$, such that t_1 is firable, and let C' be the cluster of $Prune(\mathcal{N}_\Delta)$ that contains $t_1, \dots, t_k$, and additional transitions $t_{k+1} \dots t_{k+q}$. As t_1 is firable, then there exists a timed word $w.(t_1, d_1) \in \mathit{Lang}(\mathcal{N})$. The word $w.(t_1, d_1)$ is also a timed word of $\mathit{Lang}(\mathcal{N}_\Delta)$, so t_1 is firable in $\mathcal{N}_\Delta$. As we know that all transitions from a cluster are firable if one of them is firable, then all transitions $t_{k+1} \dots t_{k+q}$ are firable in $\mathcal{N}_\Delta$. Let us suppose that for every cluster C_Δ of $\mathcal{N}_\Delta$, either i) the cluster C of $\mathcal{N}$ containing a subset of transitions of C_Δ is equal to C_Δ, or ii) no transition of C is firable. If $C_\Delta = C$ for a cluster, then new firable transitions of $\mathcal{N}_\Delta$ do not come from this cluster. If no transition from C is firable in $\mathcal{N}$ but transitions of C and C_Δ are firable in $\mathcal{N}_\Delta$, then, there exists a process in $\mathit{Untime}(\mathcal{N}_\Delta)$ that contains transitions from C. This process contains a (possibly empty) prefix N' that is a process of $\mathit{Untime}(\mathcal{N})$. The configuration reached after N' allows at least one firable transition of $\mathcal{N}_\Delta$ that is not firable in $\mathcal{N}$. Hence, this both contradicts the fact that all clusters that have firable transitions remains unchanged or are never fired. $\square$

Thus, it suffices to look at each cluster of $Prune(\mathcal{N})$ and compute the smallest possible enlargement that does not give new behaviors. More formally, two intervals I_1, I_2 are *neighbors* if the smallest closed intervals containing them have a non-empty intersection. Given two intervals I_1 with end-points $a \leq b$ and I_2 with end-points $c \leq d$ that are not neighbors and such that $b < c$, then one can easily compute a value $\Delta_{I1,I2} < (c-b)/2$. One can for instance choose $\Delta_{I1,I2} = \frac{(c-b)}{4}$. For a cluster C of $\mathcal{N}$ with transitions $t_1 \dots t_k$ such that $t_1 \dots t_i$ are not pruned, and $t_{i+1} \dots t_k$ are pruned, we have $eft(t_j) > min\{lft(t_q), q \in 1..i\}$ for every $j \in i+1..k$. Similarly we can compute $\Delta_C = min\{eft(t_j) - min\{lft(t_q), q \in 1..i\}\}$. Then, enlarging $\mathcal{N}$ by $\frac{\Delta_C}{4}$ does not change the set of transitions preserved by pruning. Now, if any transition of $t_{i+1} \dots t_k$ is a neighbor of $[0, min\{lft(t_q), q \in 1..i\}]$, such a Δ_C does not exist.

Consequently to check robustness of firability, it suffices to check existence of a value Δ_C for each cluster C of $\mathcal{N}$ that has a firable transition. If one such cluster does not allow computing a strictly positive enlargement, then the net is not robustly firable. Otherwise, it suffices to choose as Δ the smallest value allowing enlargement encountered for a cluster of $\mathcal{N}$. Clearly, enlarging $\mathcal{N}$ by Δ does not change the firability set of $\mathcal{N}$. $\square$

We can now characterize two classes of unbounded FC-TPNs for which robustness of fireability set is guaranteed by definition, i.e., nets from these classes are always robustly firable under guard enlargement.

Corollary 2. *Let $\mathcal{N}$ be a FC-TPN such that $Untime(Prune(\mathcal{N})) = Untime(\mathcal{N})$. Then $\mathcal{N}$ is robustly firable w.r.t guard enlargement.*

Corollary 3. *Let $\mathcal{N} = (\mathcal{U}, M_0, I)$ be a FC-TPN which uses only closed guards (in range of I). Then, cN is robustly firable under guard enlargement.*

Proof: First of all, recall that enlargement can only add new transitions and hence new behaviors to a net. Now, consider how enlargement affects pruning. Let $t_1, \dots t_k$ be transitions from the same cluster C, and let $I_1 = [\alpha_1, \beta_1], \dots I_k = [\alpha_k, \beta_k]$ be the closed intervals attached to these transitions. Let $maxup_C = \min\{\beta_i, i \in 1..k\}$. Pruning of this cluster enlarged by some value Δ keeps transition t_j such that $\alpha_j - \Delta \leq maxup_C + \Delta$. Now we can prove that for nets with closed intervals, there exists a value Δ such that $Untime(Prune(\mathcal{N})) = Untime(Prune(\mathcal{N}_\Delta))$. For a pruned cluster C, let us denote by $minlow_C$ the minimal lower bound of intervals associated with a transition that is in C but not in $Prune(C)$. We have $minlow_C > maxup_C$, and we can hence choose a value $d_C < 1/4(maxlow_C - minlow_C)$ such that $\alpha_j - d_C \leq maxup_C + d_C$, i.e such that $Prune(C)$ is the same in $\mathcal{N}$ and $\mathcal{N}_{d_C}$. If one chooses an enlargement by a value Δ strictly smaller than $md = \min\{d_C \mid C \text{ is a pruned cluster of } \mathcal{N}\}$, then the pruned clusters of $\mathcal{N}$ and $\mathcal{N}_\Delta$ are the same, and hence for any such value $0 < \Delta < md$, $Untime(prune(\mathcal{N})) = Untime(prune(\mathcal{N}_\Delta))$. Such a value exists iff $md > 0$, which is guaranteed when intervals are closed, ensuring $maxlow_C - minlow_C > 0$ for every cluster C. $\square$

6.1.2. Robustness of firability w.r.t guard shrinking

Consider the net $\mathcal{N}$ in Figure 9 which has two firable transitions a, b. In the net $\mathcal{N}_\triangledown$ obtained by shrinking guards, for any value of Δ, only transition a remains firable. One can also remark that the net in Figure 9 has closed intervals. This means that unlike for guard enlargement (Corollary 3), firability is not robust w.r.t. guard shrinking in the class of FC-TPNS with closed guards.

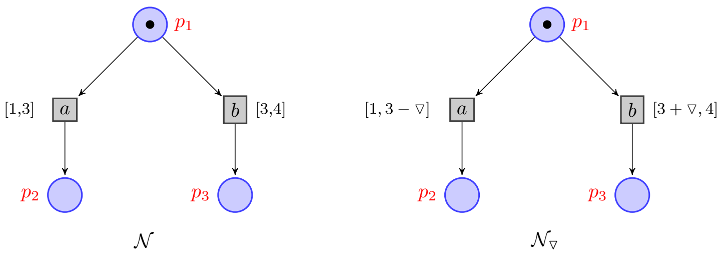

Figure 9: A FC-TPN and guard shrinking

Observe that in a cluster of a free choice net, guard shrinking can disable a transitions. So given a cluster, such that at least one transition is useful (can be fired eventually) in it, the set of clusters in $\mathcal{N}_\triangledown$ is the set of clusters of $\mathcal{N}$ but pruned with intervals $I_\triangledown(t)$. Just as we did for guard enlargment, for a given cluster C, one can compute the maximal value by which transitions of C can be shrinked without changing the contents of $Prune(C)$. Then, following the lines of proof of Theorem 4, by considering the minimal shrinking value among all clusters, one can decide wheter there exists a shrinking value that does not change the firable sets of transitions in all clusters for which some transition is firable in $\mathcal{N}$. Hence we have the following result:

Theorem 5. *Let $\mathcal{N}$ be a FC-TPN without 0-delay sequences. Then checking whether $\mathcal{N}$ is robustly firable under shrinking of guards is decidable. If $\mathcal{N}$ is robustly firable, then one can effectively compute a value $\triangledown$ such that* Firable$(\mathcal{N})$ = Firable$(\mathcal{N}_\triangledown)$.

6.2. Robustness of Termination for FC-TPNs

Next, we move to the problem of termination in FC-TPNs. We are interested in formulating when a system modeled as an FC-TPN is robustly terminating, i.e., it terminates even under (infinitesimally) small perturbations. We start with two definitions to capture guard enlargement and shrinking as before.

Definition 12 (Robustness of termination). *Let $\mathcal{N}$ be a TPN. The termination robustness problem can be formalized as follows.*

- **(robustness wrt guard enlargement)** *Does there exist an infinitesimal $\Delta > 0$ such that all runs of $\mathcal{N}$ are terminating iff all runs of $\mathcal{N}_\Delta$ are terminating ?*
- **(robustness wrt guard shrinking)** *Does there exist an infinitesimal $\nabla > 0$ such that all runs of $\mathcal{N}$ are terminating iff All runs of $\mathcal{N}_\nabla$ are terminating.*

A net $\mathcal{N}$ is said to be robust wrt to termination under guard enlargement (resp. shrinking) if there is a Δ (resp. $\nabla > 0$), such that $\mathcal{N}_\Delta$ (resp. $\mathcal{N}_\nabla$) satisfies the above property (i.e., it has a non-terminating run iff $\mathcal{N}$ has). Termination is robust for class of TPNs *under enlargement/shrinking if it is robust for all nets in the class.*

6.2.1. Robustness of Termination wrt guard enlargement

As for firability, using open guards gives us a simple example of an FC-TPN that is terminating (i.e., has no non-terminating run), but which enlargement is not terminating. For completeness, we show an explicit example of such a net $\mathcal{N}$ in Figure 10. This net is a free choice net, and defines the timed language $L_\mathcal{N} = \{(a, d) \mid 0 \geq d < 1\}$. All runs of $\mathcal{N}$ are terminating. However, for any value of Δ, the language of $\mathcal{N}_\Delta$ becomes $L_{\mathcal{N}_\Delta} = L_\mathcal{N} \cup \{(b, d).(c, d1) \dots (c, d_k) \mid 1 \leq d \leq 2 \wedge \forall i \in 1..kd_{i+1} - d_i \in [0, 1]\}$ Clearly, the enlarged net $\mathcal{N}_\Delta$ has at least one non-terminating run.

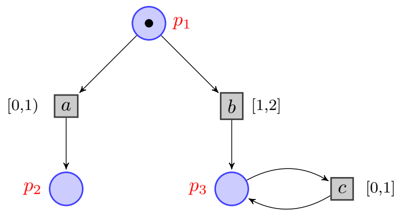

Figure 10: A FC-TPN that is not robust wrt terminating under guard enlargement

However, this situation does not arise for FC-TPNs which use only closed guards.

Proposition 4. *Termination is a robust property wrt guard enlargement for the class of FCTPNs with closed guards.*

Proof: We can use essentially the same arguments as in Corollary 3 to show that for FC-TPNs with closed guards, there exists a value Δ such that we have $Untime(Prune(\mathcal{N})) = Untime(Prune(\mathcal{N}_\Delta))$. Then, if net $\mathcal{N}$ has a non-terminating run ρ, then—as enlargement can only add behaviors—ρ is also a non-terminating run of enlarged net $\mathcal{N}_\Delta$. If all runs of $\mathcal{N}_\Delta$ are terminating, then all runs of $Untime(Prune(\mathcal{N}_\Delta))$ terminate, for any value $0 < \Delta < md$. As a consequence, all runs of $Untime(prune(\mathcal{N}))$ also terminate and hence all runs of $\mathcal{N}$ are terminating runs. $\square$

Now for general FC-TPNs, which may also have some open/semi-open guards, as non-terminating runs could arise due to enlargement, we would like to check whether a given FC-TPN is robust wrt termination or not. The following theorem shows shows that this problem is indeed decidable.

Theorem 6. *Robustness of termination wrt guard enlargement is decidable for FC-TPNs without* 0*-delay sequences.*

Proof: For this proof, we need to extend the notion of pruning seen in Section 4.1. Let $t_1, \dots t_k$ be the set of transitions appearing in a cluster C, and let $I_1, \dots I_k$ be the intervals associated with these transitions, with respective lower bound α_t and upper bound β_t. Let $maxup_C = \min\{\beta_i, i \in \{1, .., k\}\}$ be the minimal upper bound of intervals attached to transitions in C. Recall that any transition which lower bound α_j is greater than $maxup$ will never fire in $\mathcal{N}$. The *extended pruned cluster* obtained from C is the set of transitions t_i such that $\alpha_i \leq maxup$.

The intervals associated with transitions in the extended pruned cluster are the closed intervals $I'(t_i) = [\alpha_i, \min(\beta_i, maxup_C)]$. The major difference between pruning and extended pruning is that extended pruning does not remove transitions that are neighbors of transitions that can fire in $\mathcal{N}$. Intuitively, these additional transitions are the transitions that would become fireable under any enlargement.

Let $\mathcal{N} = (\mathcal{U}, M_0, I)$ be a TPN, and let $\mathcal{N}' = (\mathcal{U}', M_0, I')$ be the net obtained by restricting $\mathcal{U}$ to its extended pruned clusters as shown above. Then we have the following claim: $\mathcal{N}$ is robust wrt to termination under guard enlargement iff either

- $\mathcal{N}$ has non-terminating runs, or
- $\mathcal{N}$ and $\mathcal{N}'$ only have runs that terminate

One can immediately notice that these two conditions can be effectively checked (due to Theorem 3). Now we proceed to prove the first implication ($\Rightarrow$) given above, i.e., assuming that $\mathcal{N}$ is robust wrt termination we prove that one of the two conditions is must hold.

Let us consider the first condition, i.e., if $\mathcal{N}$ has a non-terminating run, then so has $\mathcal{N}'$. If $\mathcal{N}$ has a non-terminating run ρ then, as enlargement can only add new behaviors to the original language of a net, ρ is also a non-terminating run of $\mathcal{N}'$.

Now, assuming the first condition is false, i.e., all runs of $\mathcal{N}$ are terminating, we have to prove that second condition holds, i.e., all runs of $\mathcal{N}'$ are terminating. We show this by contradiction. If $\mathcal{N}$ has only terminating runs and $\mathcal{N}'$ has a non-terminating run ρ then take any $\Delta > 0$ and consider $\mathcal{N}_\Delta$. We can observe easily that ρ is a run in $\mathcal{N}_\Delta$ as well (since all transitions in clusters of $\mathcal{N}'$ are neighors of transitions of $\mathcal{N}$). Hence, for any value of $\Delta > 0$ the termination property differs for $\mathcal{N}$ and $\mathcal{N}_\Delta$, which violates the assumption that $\mathcal{N}$ was robust wrt termination under guard enlargement. This completes the proof of the forward direction.

Let us now prove the reverse direction ($\Leftarrow$): Assuming the first condition, that $\mathcal{N}$ has a nonterminating run, it is easy to see that for any $\Delta > 0, \mathcal{N}_\Delta$ this non-terminating run is preserved. Now, assume that $\mathcal{N}$ and $\mathcal{N}'$ only have runs that terminate. Then we have to show that one can find a value Δ for enlargement such that $\mathcal{N}_\delta$ has only terminating runs for any value $\delta \leq \Delta$. We can exhibit a value Δ such that $Untime(\mathcal{N}') = Untime(\mathcal{N}_\Delta)$, i.e. such that the enlargement by Δ does not create new untimed behaviors, and consequently no new non-terminating runs. Let us consider again the original clusters $C_1, \dots C_k$ of $\mathcal{N}$, their pruning $C'_1, \dots C'_k$, and their extended pruning $C^\epsilon_1, \dots C^\epsilon_k$. One can notice that the clusters of $\mathcal{N}'$ contain transitions that become fireable for the smallest enlargement by some infinitesimal $\epsilon > 0$. For a given pair of transitions t_i, t_j such that $t_i \in C^\epsilon_m$ and $t_j \notin C^\epsilon_m$, we have that $I(t_i)$ has a lower bound α_i smaller that or equal to the minimal upper bound $maxup$ in the cluster C_m, and $I(t_j)$ has a lower bound α_j greater than $maxup$, and it is not either a neighbor of any interval $I^\epsilon(t_q)$ associated with a transition in $t_q \in C^\epsilon_m$. Hence for every such pair of transitions, there exists a value $d_j = \alpha_j - maxup > 0$. In any enlargement by a value δ_j smaller than d_j, t_j will not appear in a pruned cluster of $\mathcal{N}_\delta$. Now, let δ denote the minimal value separating all transitions in/out of clusters obtained by extended pruning. Formally $\delta = \min\{d_j \mid \exists t_j \notin \bigcup_{i \in 1..k} C^\epsilon_i\}$. Then, any enlargement by a positive value $\Delta < \delta$ gives a net whose clusters contain the same transitions as $C^\epsilon_1, \dots C^\epsilon_k$, and hence $Untime(\mathcal{N}') = Untime(\mathcal{N}_\Delta)$. $\square$

6.2.2. Robustness of Termination wrt guard shrinking

The class of FC-TPNs is not robust wrt termination under guard shrinking. As in the case of guard enlargement, we can easily exhibit a net $\mathcal{N}$ which has an infinite run, but which shrunk version $\mathcal{N}_\triangledown$ has only terminating runs, for any value of δ. Consider for instance the net of Figure 11.

One can also notice that the example of Figure 11 has closed guards. Thus, unlike for guard enlargement (Proposition 4), the class of FCTPNs with closed guards is not robust wrt termination under shrinking of guards. However, with a slight modification, we still obtain a decidable procedure to check for robustness.

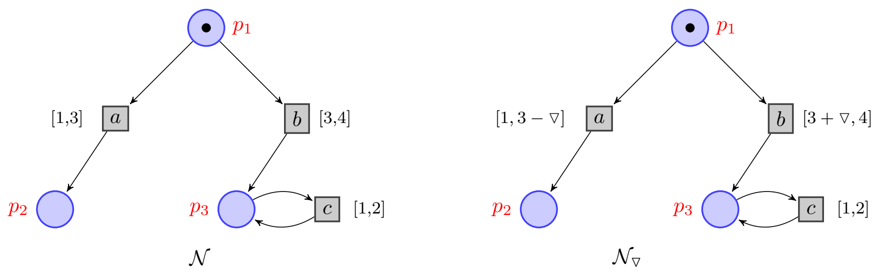

Figure 11: A non-robust FC-TPN wrt termination under guard shrinking

Theorem 7. *Let $\mathcal{N}$ be a FC-TPN witout 0-delay sequences. Then, checking robustness of termination of $\mathcal{N}$ under guard shrinking is decidable.*

Proof: We reuse the same principle as in Theorem 6, but consider clusters that will be obtained under the smallest shrinking. Let C be a cluster with transitions $t_1, \dots t_k$. As in Theorem 6, we can compute $maxup_c$, the minimal upper bound of intervals in C. The reduced cluster of C is a cluster C^{red} over transitions $\{t_i \mid \alpha_i < maxup_C\}$. Clearly, a transition t_i with interval $[\alpha_i, \beta_i]$ such that $\alpha_i > maxup_C$ will never fire, and not part of the pruned cluster C nor of its reduced cluster.

Second, if we have $\alpha_i = maxup_C$, and $\beta_j = maxup_C$, then $I(t_i)$ is a neighbour of $I(t_j)$ for some t_j with upper bound β_j. However, both intervals have at most one common point, that will disappear under any shrinking of guards.

Let $\mathcal{N}' = Prune(\mathcal{N})$ be the pruning of $\mathcal{N}$ and let $\mathcal{U}'$ be the untiming of $\mathcal{N}'$. Let $\mathcal{N}^{red}$ be the restriction of $\mathcal{N}$ to its reduced clusters, and $\mathcal{U}^{red}$ be the untiming of $\mathcal{N}^{red}$. Now, by definition of termination in FC-TPNs, we have the following property: termination is a robust property of $\mathcal{N}$ under guard shrinking iff

$$\mathcal{U}' \text{ has a non-terminating run } \Leftrightarrow \mathcal{U}^{red} \text{ has a non-terminating run}$$

As before, checking this property is feasible on untimed Petri nets, and hence we obtain a decision procedure for robustness of termination. □

7. Discussion

7.1. Necessity of assumptions

Our proof technique used to obtain decidability of firability and termination holds for free-choice Time Petri nets with a multiple server semantics, when we disallow forced 0-delay infinite runs. We now show that all these conditions are necessary for our proofs to hold.

Let us first address the free choice assumption. Without this assumption, all problems listed in Section 4 are undecidable. Indeed, the two counter machine encoding of [2] relies on urgency and uses non-free choice nets, in which transitions have at least one place with a single token. Hence, this encoding works even under a multiple server semantics. In particular, Theorem 1 does not hold without the free choice assumption. Consider the non-free choice net $\mathcal{N}_{nfc}$ in Figure 12 (left). A causal process for $Untime(\mathcal{N}_{nfc})$ is shown in Figure 12 (right, above). One can verify in this untimed net that transition c is firable. However, there is no way to build a timed causal net that contains this causal net. Indeed, transition c is not firable in $\mathcal{N}_{nfc}$ and one cannot add event e_3 in the only timed causal net defined by $\mathcal{N}_{nfc}$ depicted in Figure 12 (right, below). Note also that marking $\{p_1, p_3\}$ is reachable in $Untime(\mathcal{N}_{nfc})$ (and allows firing of c), but not in $\mathcal{N}_{nfc}$.

Next, we discuss the choice of a multi-server semantics. With this semantics, one can consider that a new clock is initialized at any enabling of a cluster (as all transitions from a cluster carry the same set of clocks). Furthermore, when a transition is fired, all remaining instances of transitions in a cluster keep their clocks unchanged. Such a semantics is meaningful and arguably powerful enough, for instance, to address business

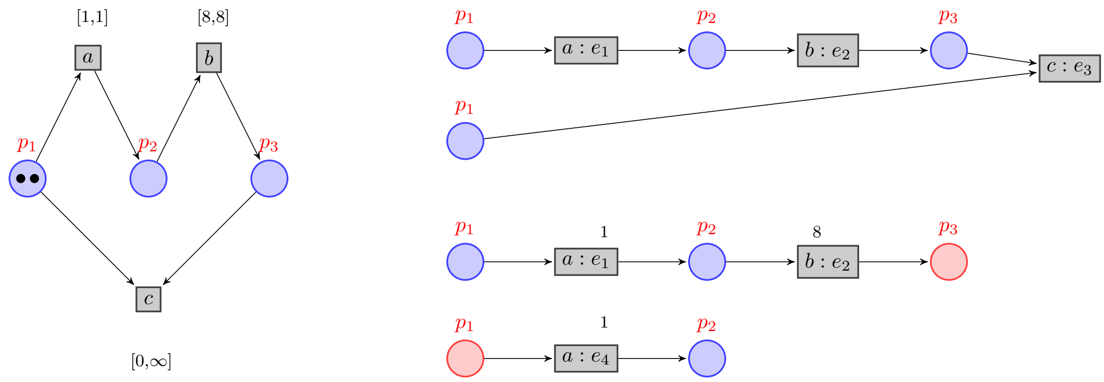

Figure 12: A non-free choice TPN $\mathcal{N}_{nfc}$ (left), and a causal process (right, up) and a timed causal process (right, below) for it.

processes with time that do not contain fork-join constructs, i.e. where case processing are sequences of choices up to completion. Unsurprisingly, with a single server semantics, where we keep only one clock per transition instead of one for every instance of a multi-enabled transition, even our Pruning Lemma 1 fails. This is due to the fact that one cannot remove useless transitions as easily as in multi-server semantics: due to multiple enabling, firing a transition in a cluster under single server semantics does not always reset clocks of competing transitions. These transitions, including pruned ones, may become firable and even urgent at a later date.

Finally, the 0-delay assumption forbids an arbitrary number of transitions to occur at the same time instant. If this condition is not met, then our algorithm used to build a timed process of $\mathcal{N}$ from an untimed process of $\mathit{Untime}(\mathit{Prune}(\mathcal{N}_P))$ does not necessarily terminate (Lemma 3). Furthermore, eagerness of urgent transitions firing in zero time can prevent other transitions from firing, which may result in discrepancies between untiming of timed processes of a net and untimed processes of the untiming of this net. Consider the pruned and free choice net $\mathcal{N}_2$ depicted in the Figure below. The only allowed executions of $\mathcal{N}_2$ are $\mathit{Lang}(\mathcal{N}_2) = \{(a,0)^k \mid k \in \mathbb{N}\}$. Hence, $\mathcal{N}_2$ has a 0-delay firing sequence a^ω, and transition b is not firable. However, $\mathit{Untime}(\mathcal{N}_2)$ allows sequences of the form $a^k.b$, and hence transition b is firable. Note that this does not mean that there is no effective procedure to build a timed causal net from an untimed causal net, and our algorithm could be adapted to handle Zeno behaviors.

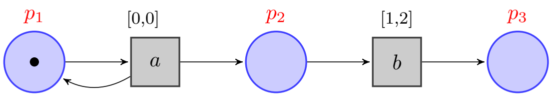

Figure 13: A Free choice TPN $\mathcal{N}_2$ with Zeno behavior

7.2. Difficulty with tackling other decision problems

As mentioned earlier, our proof technique, using relation between untimed processes and processes of untimed nets does not work for any property. In particular for reachability, coverability and for boundedness, we hit some roadblocks that we now explain.

We start with reachability. Consider, for instance, the net shown in Figure 14. It is clear that marking $\{p_1, q_1\}$ is not reachable from the initial marking $\{p_1, q_3\}$ with a multi-server semantics. However, it is reachable in the corresponding untimed net. Note also that marking $\{p_1, q_3\}$ is also reachable for this net under a weak semantics (as proposed in [3]), which proves that strong multi-server and weak semantics of TPNs differ.

Figure 14: Pruning and untiming does not preserve reachability

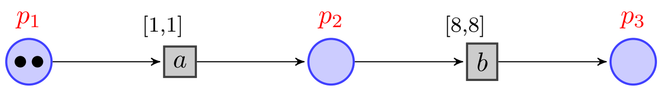

Figure 15: A free-choice TPN $\mathcal{N}_{cov}$

Next, we move to coverability. Consider the FC-TPN $\mathcal{N}_{cov}$ shown in Figure 15. Clearly, when removing time constraints from $\mathcal{N}_{cov}$, the obtained (untimed) Petri net allows sequence of transitions $a.b$, which leaves the net in a marking M such that $M(p_1) = M(p_3) = 1$ and $M(p_2) = 0$. However, the timed language of $\mathcal{N}$ contains only $w = (a, 1)(a, 2)(b, 8)$ and its prefixes. So, marking M is not reachable or even coverable by $\mathcal{N}$. Thus, unlike for fireability and termination, one cannot immediately decide coverability (or reachability) of a marking in an FC-TPN just by looking at its untimed version. This does not contradict Corollary 1: once untimed, the causal process for w does indeed contain the causal process associated with $a.b$.

Finally, the question of boundedness also does not immediately follow on the same lines as the proofs of theorems 2 and 3. Consider, for instance, the net $\mathcal{N}_{bd}$ of figure 16. The untimed version of this net allows sequences of transitions of the form $(t_1.t_2)^k$, for arbitrary value $k \in \mathbb{N}$. At each iteration of this sequence, place p_3 receives a new token. This net is free choice, and following the result of theorem 1, for every untimed process U_k associated with a sequence $(t_1.t_2)^k$, there exists a timed process N'_k of $\mathcal{N}_{bd}$. However, this process necessarily contains as many occurrences of t_3, t_1 and t_2. As t_3 occurs immediately as soon as p_3 is filled, the only sequences of moves allowed by N'_k are of the form $(t_1.t_2.t_3)^k$, and hence $\mathcal{N}_{bd}$ is bounded, even if $Untime(\mathcal{N}_{bd})$ is unbounded.

8. Conclusion

In this paper, we defined a class of unbounded time Petri nets with free-choice, which are well-behaved with respect to several properties. In particular, we showed decidability of firability and termination. We crucially use the multi-server semantics to obtain these results. Next, we looked at robustness questions under both guard enlargement and shrinking for the problem of firability and termination for these FC-TPNs. In both cases, we show subclasses that are robust and decision procedures to check for robustness in general.

However, for closely-related problems such as coverability and boundedness both decidability and robustness are still left open. Indeed, the proofs relying of the relation between timed causal nets and their untimed versions established by the prunning lemma 1 cannot be extended directly to handle these problems as shown through the examples above. Despite these remarks, we believe that we can indeed modify the techniques in this paper to get decidability for coverability and boundedness problems for FC-TPNs. It is unclear whether reachability for FC-TPNs would similarly be decidable. Finally, several questions for robustness are still open for further investigation.

References

[1] P. Merlin, A study of the recoverability of computing systems, Ph.D. thesis, University of California, Irvine, CA, USA (1974).

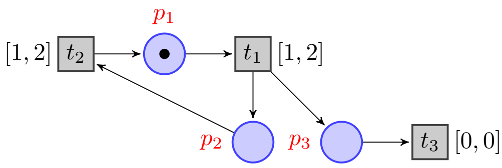

Figure 16: An FC-TPN $\mathcal{N}_{bd}$ that is bounded, but whose untimed version is not

[2] N. Jones, L. Landweber, Y. Lien, Complexity of some problems in Petri nets, TCS 4 (3) (1977) 277–299.
[3] P. Reynier, A. Sangnier, Weak time Petri nets strike back!, in: CONCUR, Vol. 5710 of LNCS, Springer, 2009, pp. 557–571.
[4] S. Akshay, B. Genest, L. Hélouët, Decidable classes of unbounded Petri nets with time and urgency, in: PETRI NETS'16, Vol. 9698 of LNCS, Springer, 2016, pp. 301–322.
[5] A. Puri, Dynamical properties of timed automata, In DEDS 10 (1-2) (2000) 87–113.
[6] S. Akshay, L. Hélouët, C. Jard, P.-A. Reynier, Robustness of time Petri nets under guard enlargement, Fundam. Inform. 143 (3-4) (2016) 207–234.
[7] H. Boucheneb, D. Lime, O. Roux, On multi-enabledness in time Petri nets, in: Proc. of PETRI NETS'13, Vol. 7927 of LNCS, Springer, 2013, pp. 130–149.
[8] J. Esparza, J. Desel, Free Choice Petri nets, Cambridge University Press, 1995.
[9] P. Abdulla, A. Nylén, Timed Petri nets and BQOs, in: Proc. of ICATPN 2001, Vol. 2075 of LNCS, Springer, 2001, pp. 53–70.
[10] V. Ruiz, D. de Frutos-Escrig, F. Cuartero, Timed processes of timed Petri nets, in: Proc. of ICATPN, Vol. 935 of LNCS, Springer, 1995, pp. 490–509.
[11] J. Winkowski, Algebras of processes of timed Petri nets, in: CONCUR '94, Vol. 836 of LNCS, Springer, 1994, pp. 194–209.
[12] T. Aura, J. Lilius, A causal semantics for time Petri nets, TCS 243 (1-2) (2000) 409–447.
[13] D. Bushin, I. Virbitskaite, Time process equivalences for time Petri nets, in: Workshop on Concurrency, Specification and Programming, Vol. 1269 of CEUR Workshop, 2014, pp. 257–268.
[14] T. Chatain, C. Jard, Complete finite prefixes of symbolic unfoldings of safe time Petri nets, in: ICATPN'06, Springer, 2006, pp. 125–145.
[15] T. Chatain, C. Jard, Back in time Petri nets, in: Proc. of FORMATS'13, Vol. 8053 of LNCS, Springer, 2013, pp. 91–105.
[16] S. Akshay, L. Hélouët, R. Phawade, Combining free choice and time in petri nets, in: 23rd International Symposium on Temporal Representation and Reasoning, TIME 2016, Kongens Lyngby, Denmark, October 17-19, 2016, IEEE Computer Society, 2016, pp. 120–129.
[17] B. Berthomieu, M. Diaz, Modeling and verification of time dependent systems using time Petri nets, IEEE Trans. in Software Engineering 17 (3) (1991) 259–273.
[18] A. Cerone, A. Maggiolo-Schettini, Time-based expressivity of time Petri nets for system specification, TCS 216 (1-2) (1999) 1–53.
[19] M. Boyer, M. Diaz, Multiple enabledness of transitions in Petri nets with time, in: Proc. of PNPM'01, IEEE, 2001, pp. 219–228.
[20] D. Lime, O. Roux, Model checking of time Petri nets using the state class timed automaton, Journal of Discrete Event Dynamic Systems 16 (2) (2006) 179–205.
[21] C. Rackoff, The covering and boundedness problem for vector addition systems, TCS 6 (1978) 223–231.
[22] A. Finkel, J. Leroux, Recent and simple algorithms for Petri nets, Software and System Modeling 14 (2) (2015) 719–725.
[23] L. Clemente, F. Herbreteau, A. Stainer, G. Sutre, Reachability of communicating timed processes., in: FoSSaCS, Vol. 7794 of LNCS, 2013, pp. 81–96.