On-The-Fly Model Checking of Timed Properties on Time Petri Nets

Kais Klai

LIPN, CNRS UMR 7030
Université Paris 13, Sorbonne Paris Cité
99 avenue Jean-Baptiste Clément
F-93430 Villetaneuse, France
kais.klai@lipn.univ-paris13.fr

Abstract. This paper deals with model checking of timed systems modeled by Time Petri nets (TPN). We propose a new finite graph, called Timed Aggregate Graph (TAG), abstracting the behavior of bounded TPNs with strong time semantics. The main feature of this abstract representation compared to existing approaches is the encoding of the time information. This is done in a pure way within each node of the TAG allowing to compute the minimum and maximum elapsed time in every path of the graph. The TAG preserves runs and reachable states of the corresponding TPN and allows for on-the-fly verification of reachability properties. We illustrate in this paper how the TAG can be used to check some usual timed reachability properties and we supply an algorithm for extracting an explicit timed trace (involving the elapsed time before each fired transition) from an abstract run of the TAG. The TAG-based approach is implemented and compared to two well known TPNs analysis approaches.

1 Introduction

Time Petri nets are one of the most used formal models for the specification and the verification of systems involving explicit timing constraints, such as communication protocols, circuits, or real-time systems. The main extensions of Petri nets with time are *time Petri nets* [18] and *timed Petri nets* [22]. In the former, a transition can fire within a time interval whereas, in the latter, time durations can be assigned to the transitions; tokens are meant to spend that time as reserved in the input places of the corresponding transitions. Several variants of timed Petri nets exist: time is either associated with places (p-timed Petri nets), with transitions (t-timed Petri nets) or with arcs (a-timed Petri nets) [23]. The same holds for time Petri nets [7]. In [21], the authors prove that p-timed Petri nets and t-timed Petri nets have the same expressive power and are less expressive than time Petri nets. Several semantics have been proposed for each variant of these models. Here we focus on t-time Petri nets, which we simply call TPNs. There are two ways of letting the time elapse in a TPN [21]. The first way, known as the *Strong Time Semantics* (STS), is defined in such a manner that time elapsing cannot disable a transition. Hence, when the upper bound of a firing interval is reached, the transition must be fired. The other semantics, called *Weak Time Semantics* (WTS), does not make any restriction on the elapsing of time.

For real-time systems, dense time model (where time is considered in the domain $\mathbb{R}_{\geq 0}$) is the unique possible option, raising the problem of handling an infinite number of states. In fact, the set of reachable states of the TPN is generally infinite due to the infinite number of time successors a given state could have. Two main approaches are used to treat this state space: region graphs [1] and the state class approach [3]. The other methods [2,24,4,10,5,17,6,11] are either refinements, improvements or derived from these basic approaches. The objective of these representations is to yield a state-space partition that groups concrete states into sets of states presenting similar behavior with respect to the properties to be verified. These sets of states must cover the entire state space and must be finite in order to ensure the termination of the verification process.

In this work, we propose a new finite graph, called Timed Aggregate Graph (TAG), abstracting the behavior of bounded TPNs with strong time semantics. A preliminary version of this work has been published in [13,14], where a coarser abstraction of TPNs' state graph is proposed. The key idea behind the approach presented in this paper is the fact that the time information associated with each node is related to the current path leading to this node. In particular, given a node of the TAG, for each couple of enabled transitions $\langle t, t' \rangle$, the value of the earliest and latest firing times of t (reps. t') the last time, in the current path, it "met" t' (resp. t) is stored in the node. This information , represented by a matrix, allows us (1) to maintain the relative differences between the firing times of enabled transitions (diagonal constraints), (2) to determine the fireable transitions at each node, and (3) to compute dynamically the earliest and the latest firing time of each enabled transition for each node of the TAG. This new version of the TAG allows to preserve the timed traces of the underlying TPN while the abstraction proposed in [13,14] is an upper approximation of the set of traces of the underlying TPN. Moreover, one can compute the minimum and maximum elapsed time through every path of the graph which permits on-the-fly verification of timed reachability properties (e.g., is some state reachable between d and D time units).

This paper is organized as follows: In Section 2, some preliminaries about TPNs and the corresponding semantics are recalled. In Section 3, we define the Timed Aggregate Graph (TAG) associated with a TPN and we discuss the main preservation results of the TAG-based approach. In Section 4, we show how the verification of some usual reachability properties can be accomplished on-the-fly by exploring the TAG. Section 5 relates our work to existing approaches. In Section 6, we discuss the experimental results obtained with our implementation compared to two well-known tools, namely Romeo [9] and TINA [5]. Finally, a conclusion and some perspectives are given in Section 7.

2 Preliminaries and Basic Notations

A TPN is a P/T Petri net [20] where a time interval $[t_{\min}; t_{\max}]$ is associated with each transition t.

Definition 1. *A* TPN *is a tuple* $\mathcal{N} = \langle P, T, Pre, Post, I\rangle$ *where:*

- $\langle P, T, Pre, Post\rangle$ *is a* P/T Petri net
- $I : T \longrightarrow \mathbb{N} \times (\mathbb{N} \cup \{+\infty\})$ *is the* time interval function *such that:* $I(t) = (t_{\min}, t_{\max})$*, with* $t_{\min} \le t_{\max}$*, where* $t_{\min}$ *(resp.* $t_{\max}$*) is the earliest (resp. latest) firing time of transition* t*.*

A *marking* of a TPN is a function $m : P \longrightarrow \mathbb{N}$ where $m(p)$, for a place p, denotes the number of tokens in p. A *marked TPN* is a pair $\mathcal{N} = \langle \mathcal{N}_1, m_0\rangle$ where $\mathcal{N}_1$ is a TPN and m_0 is a corresponding *initial marking.* A transition t is enabled by a marking m iff $m \ge Pre(t)$ and $Enable(m) = \{t \in T : m \ge Pre(t)\}$ denotes the set of enabled transitions in m. If a transition t_i is enabled by a marking m, then $\uparrow(m, t_i)$ denotes the set of newly enabled transitions [2]. Formally, $\uparrow(m, t_i) = \{t \in T \mid t \in Enable(m - Pre(t_i) + Post(t_i)) \wedge (t \notin Enable(m - Pre(t_i)) \vee (t = t_i))\}$. If a transition t is in $\uparrow(m, t_i)$, we say that t is newly enabled by the successor of m by firing t_i. Dually, $\downarrow(m, t_i) = Enable(m - Pre(t_i) + Post(t_i)) \setminus \uparrow(m, t_i)$ is the set of oldly enabled transitions. The possibly infinite set of reachable markings of $\mathcal{N}$ is denoted $Reach(\mathcal{N})$. If the set $Reach(\mathcal{N})$ is finite we say that $\mathcal{N}$ is bounded.

The semantics of TPNs can be given in terms of Timed Transition Systems (TTS) [15] which are usual transition systems with two types of labels: discrete labels for events (transitions) and positive real labels for time elapsing (delay). States (configurations) of the TTS are pairs $s = (m, V)$ where m is a marking and $V : T \longrightarrow \mathbb{R}_{\ge 0} \cup \{\bot\}$ a time valuation. In the following, $s.m$ and $s.V$ denote the marking and the time valuation respectively of a state s. If a transition t is enabled in m then $V(t)$ is the elapsed time since t became enabled, otherwise $V(t) = \bot$. Given a state $s = (m, V)$ and a transition t, t is said to be fireable in s iff $t \in Enable(m) \wedge V(t) \ne \bot \wedge t_{min} \le V(t) \le t_{\max}$.

Definition 2 (Semantics of a TPN). *Let* $\mathcal{N} = \langle P, T, Pre, Post, I, m_0\rangle$ *be a marked TPN. The semantics of* $\mathcal{N}$ *is a TTS* $\mathcal{S}_{\mathcal{N}} = \langle Q, s_0, \rightarrow\rangle$ *where:*

1. Q *is a (possibly infinite) set of states*
2. $s_0 = (m_0, V_0)$ *is the initial state such that:*

$$\forall t \in T,\ V_0(t) = \begin{cases} 0 & \textit{if } t \in Enable(m_0) \\ \bot & \textit{otherwise} \end{cases}$$

3. $\rightarrow \subseteq Q \times (T \cup \mathbb{R}_{\ge 0}) \times Q$ *is the discrete and continuous transition relations:*
   (a) *the discrete transition relation:*
   $\forall t \in T :\ (m, V) \xrightarrow{t} (m', V')$ *iff:*

$$\begin{cases} t \in \mathit{Enable}(m) \wedge m' = m - \mathit{Pre}(t) + \mathit{Post}(t) \\ t_{min} \leq V(t) \leq t_{max} \\ \forall t' \in T : V'(t') = \begin{cases} 0 & \mathit{if}\ t' \in \uparrow(m,t) \\ V(t') & \mathit{if}\ t' \in \downarrow(m,t) \\ \perp & \mathit{otherwise} \end{cases} \end{cases}$$

(b) the continuous transition relation: $\forall d \in \mathbb{R}_{\geq 0}$, $(m, V) \xrightarrow{d} (m', V')$ *iff:*

$$\begin{cases} \forall t \in \mathit{Enable}(m),\ V(t) + d \leq t_{max} \\ m' = m \\ \forall t \in T : \\ V'(t) = \begin{cases} V(t) + d & \mathit{if}\ t \in \mathit{Enable}(m); \\ V(t) & \mathit{otherwise.} \end{cases} \end{cases}$$

The above definition requires some comments. First, a state change occurs either by the firing of transitions or by time elapsing: The firing of a transition may change the current marking while the time elapsing may make some new transitions fireable. Second, the delay transitions respect the STS semantics: an enabled transition must fire within its firing interval unless it is disabled by the firing an other transition.

Given a TPN $\mathcal{N}$ and the corresponding TTS $\mathcal{S}_{\mathcal{N}}$, a path $\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots$, where $\alpha_i \in (T \cup \mathbb{R}_{\geq 0})$, is a run of $\mathcal{S}_{\mathcal{N}}$ iff $(s_i, \alpha_i, s_{i+1}) \in \rightarrow$ for each $i = 0, 1, \ldots$. The length of a run π can be infinite and is denoted by $\mid \pi \mid$. The possibly infinite set of runs of $\mathcal{S}_{\mathcal{N}}$ is denoted $[\mathcal{S}_{\mathcal{N}}]$. Without loss of generality, we assume that for each non empty run $\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots$ of a STS corresponding to a TPN, there do not exist two successive labels α_i and α_{i+1} belonging both to $\mathbb{R}_{\geq 0}$. Then, π can be written, involving the reachable markings of $\mathcal{N}$, as $\pi = m_0 \xrightarrow{(d_1,t_1)} m_1 \xrightarrow{(d_2,t_2)} \ldots$ where d_i is the time elapsed at marking m_{i-1} before firing t_i. In order to associate a run π of $\mathcal{S}_{\mathcal{N}}$ with a run of $\mathcal{N}$, denoted $\mathcal{P}(\pi)$, we define the following projection function, where $\centerdot$ denotes the concatenation operator between paths and π^i, for $i = 0, 1 \ldots$, denotes the suffix of π starting at state s_i.

$$\mathcal{P}(\pi) = \begin{cases} s_0.m & \text{if } \mid \pi \mid = 0 \\ s_0.m \xrightarrow{(0,\alpha_1)} \centerdot \mathcal{P}(\pi^1) & \text{if } \alpha_1 \in T \\ s_0.m \xrightarrow{(\alpha_1,\alpha_2)} \centerdot \mathcal{P}(\pi^2) & \text{if } \alpha_1 \in \mathbb{R}_{\geq 0} \wedge \mid \pi \mid \geq 2 \\ s_0.m \xrightarrow{\alpha_1} \centerdot \mathcal{P}(\pi^1) & \text{if } \alpha_1 \in \mathbb{R}_{\geq 0} \wedge \mid \pi \mid = 1 \end{cases}$$

3 Abstraction of a TPN State Space

3.1 Timed Aggregate Graph

In this subsection, we propose to abstract the reachability state space of a TPN using a new graph called Timed Aggregate Graph (TAG) where nodes are called *aggregates* and are grouping sets of states of a TTS. The key idea behind TAGs is the way the time information is encoded inside aggregates. In addition to the marking characterizing an aggregate, the time information is composed of two parts:

- The first part of the time information characterizing an aggregate is a dynamically updated interval, namely (α_t, β_t), associated with each enabled transition t. This interval gives the earliest and the latest firing times of any enabled transition starting from the corresponding aggregate. Either the corresponding transition is fireable at the current aggregate and the system must remain within the aggregate at least α_t time units and at most β_t time units (as long as the other enabled transitions remain fireable) before firing t, or t is not possible from the current aggregate (e.g. because of some diagonal constraint), and the system must move to an other aggregate by firing other transitions until t becomes fireable. In the latter case, the system must consume at least α_t, and can consume at most β_t to make t fireable in the future.
- The second part of the time information characterizing an aggregate is a matrix, namely $Meet$, allowing to dynamically maintain the relative differences between the firing times of enabled transitions (diagonal constraints). Given two enabled transitions t_1 and t_2, $Meet(t_1, t_2)$ is an interval representing the earliest and the latest firing times of t_1 the last time both t_1 and t_2 were enabled (through the paths leading to the aggregate).

Before we formally define the TAG and illustrate how the attributes of an aggregate are computed dynamically, let us first formally define aggregates.

Definition 3 (Timed Aggregate). *Let $\mathcal{N} = \langle P, T, Pre, Post, I\rangle$ be a* TPN. *A* timed aggregate *associated with $\mathcal{N}$ is a tuple $a = (m, E, Meet)$, where:*

- *m is a marking*
- *$E = \{\langle t, \alpha_t, \beta_t\rangle \mid t \in Enable(m),\ \alpha_t \in \mathbb{N} \wedge \beta_t \in \mathbb{N} \cup \{+\infty\}\}$ is a set of enabled transitions each asssociated with two time values.*
- *$Meet$ is a matrix s.t. $\forall t, t' \in Enable(m)$, $Meet(t, t') = \langle \alpha, \beta\rangle$ where α (resp. β) represents the earliest (resp. latest) firing time of t the last time t and t' are both enabled before reaching the aggregate a.*

As for the states of a TTS, the attributes of an aggregate a are denoted by $a.m$, $a.E$ and $a.Meet$. Moreover, $a.Meet(t, t').\alpha$ (resp. $a.Meet(t, t').\beta$) is denoted by $a.\alpha_t^{m(t,t')}$ (resp. $a.\beta_t^{m(t,t')}$), or simply $\alpha_t^{m(t,t')}$ (resp. $\beta_t^{m(t,t')}$) when the corresponding aggregate is clear from the context. We use also $\alpha^{m(t,t')}$ (resp. $\beta^{m(t,t')}$) to denote $\alpha_t^{m(t,t')}$ (resp. $\beta_t^{m(t,t')}$) when the involved transition t is clear from the context.

The E attribute of an aggregate a allows to compute the minimum and the maximum time the system can elapse when its current state is within a. The following predicates (δ and Δ) compute these information for a given aggregate.

Definition 4 (Minimum and maximum stay times). *Let $a = \langle m, E\rangle$ be an aggregate, the minimum and maximum time the system can stay at a are denoted by $\delta(a)$ and $\Delta(a)$ respectively, and are defined by the two following predicates:*

– $\delta(a) = \min_{\langle t,\alpha_t,\beta_t\rangle \in E}(\alpha_t)$
– $\Delta(a) = \max_{\langle t,\alpha_t,\beta_t\rangle \in E}(\beta_t)$

The minimum (resp. maximum) stay time $\delta(a)$ (res. $\Delta(a)$) of an aggregate a allows to encapsulate the continuous transition relation within a.

Given an aggregate $a = \langle m, E\rangle$ and an enabled transition t (i.e., $\langle t, \alpha_t, \beta_t\rangle \in E$), two primordial issues must be achieved to define the semantics of the TAG: (1) is t fireable from a?, and (2) if it is the case, how do we obtain the successor aggregate by firing t from a. In the following, we answer these issues.

Definition 5. *Let $a = \langle m, E, Meet\rangle$ be an aggregate and let $\langle t, \alpha_t, \beta_t\rangle \in E$. Then, t is fireable at a, denoted by $a \overset{t}{\rightsquigarrow}$, iff $\forall \langle t', \alpha_{t'}, \beta_{t'}\rangle \in E$, $\alpha_t^{m(t,t')} \leq \beta_{t'}^{m(t',t)}$*

A transition t is fireable at an aggregate a iff there is no transition t', that is enabled by a, whose latest firing time was strictly smaller than the earliest firing time of t the last time both transitions were enabled.

Now that the firability condition is formally defined, the following definition computes the successor aggregate obtained by the firing of a given transition. In this definition, the notion of newly (and oldly) enabled transitions is extended to aggregates as follows: $\uparrow (a,t) = \uparrow (a.m, t)$ and $\downarrow (a,t) = \downarrow (a.m, t)$ for each transition t enabled by $a.m$

Definition 6. *Let $a = \langle m, E, Meet\rangle$ be an aggregate and let $\langle t, \alpha_t, \beta_t\rangle \in E$.*

Assume that t is fireable at a (following Definition 5). The aggregate $a' = \langle m', E', Meet'\rangle$ obtained by firing t from a, denoted by $a \overset{t}{\rightsquigarrow} a'$, is obtained as follows:

1. $m' = m - Pre(t) + Post(t)$
2. $E' = E'_1 \cup E'_2$, *where:*
   - $E'_1 = \bigcup_{t' \in \uparrow(a,t)} \{\langle t', t'_{\min}, t'_{\max}\rangle\}$
   - $E'_2 = \bigcup_{t' \in \downarrow(a,t)} \{\langle t', \alpha'_{t'}, \beta'_{t'}\rangle\}$ *where:*
     – $\alpha'_{t'} = \alpha_{t'} - SCR(a, t')$, *where* $SCR(a,t') =$
       $Max(0, (Min_{t'' \in Enable(a)}(Min(\beta^{m(t',t'')}, \beta^{m(t'',t')}) - (\alpha_{t'}^{m(t',t'')} - \alpha_{t'}))$
     – $\beta'_{t'} = \beta_{t'} - Max(0, (\alpha_t^{m(t,t')} - (\beta_{t'}^{m(t',t)} - \beta_{t'}))$
   - $\forall(\langle t_1, \alpha_1, \beta_1\rangle,\ \langle t_2, \alpha_2, \beta_2\rangle) \in E' \times E'$

$$Meet'(t_1,t_2) = \begin{cases} [t_{1_{\min}}, t_{1_{\max}}] & \textit{if } t_1 \in \uparrow (a,t) \\ [\alpha_1, \beta_1] & \textit{if } t_1 \in \downarrow (a,t) \wedge t_2 \in \uparrow (a,t) \\ Meet(t_1,t_2) & \textit{if } t_1 \in \downarrow (a,t) \wedge t_2 \in \downarrow (a,t). \end{cases}$$

The computation of a successor a' of an aggregate a by the firing of a transition t is guided by the following intuition: If $\downarrow (a,t) \neq \emptyset$, then *the more the system can remain at a, the less it can remain at a' and vice versa.* Otherwise, the time elapsed within a' is independent from the time elapsed within a. Thus, given a transition t' enabled by a', two cases are considered: if t' is newly enabled, then its earliest and latest firing times are statically obtained by $t'_{\min}$ and $t'_{\max}$ respectively. Otherwise, the more one can remain at a, the less will be the necessary wait time at a' before firing t'. The function SCR (Still Can Remain) allows to compute the maximum remaining time at a under the hypothesis that, since t' became enabled, it remains the maximum time at each encountered aggregate before reaching a (note that this is different from $\Delta(a)$). Thus $SCR(a, t')$ is obtained by the following reasoning: given a transition t'' that is enabled by a, it is clear that since the last time t' and t'' became both enabled, the maximum elapsed time can not be greater than $Min(\beta^{m(t',t'')}, \beta^{m(t'',t')})$ (because of the STS semantics which is used in this paper). The maximum time the system can remain at a is then obtained by subtracting from this quantity the time that is already spent during the path leading to a (i.e., $(\alpha_{t'}^{m(t',t'')} - \alpha_{t'})$). By analyzing all the transitions enabled by a the function SCR takes the minimum values in order to not violate the STS semantics rule. Similarly, the latest firing time of t' corresponds to the situation where, between the last time t and t' were both enabled and the current aggregate a, each fired transition is fired as soon as possible. Each time a transition is fired, its earliest firing time is subtracted from the latest firing time of the old transitions. However, if the quantity of time that must be subtracted from the latest firing time of t' has already been subtracted in between, then the latest firing time of t' at a' is the same latest firing time of t' at the aggregate a.

Concerning the $Meet$ attribute, given two transitions t_1 and t_2 that are enabled at a', the value of $Meet(t_1, t_2)$ is simply obtained by considering the membership of these transitions to $\uparrow(a, t)$ and to $\downarrow(a, t)$. Finally, by considering that $\infty - \infty = 0$, the previous definition allows to handle transitions having an unbounded latest firing time.

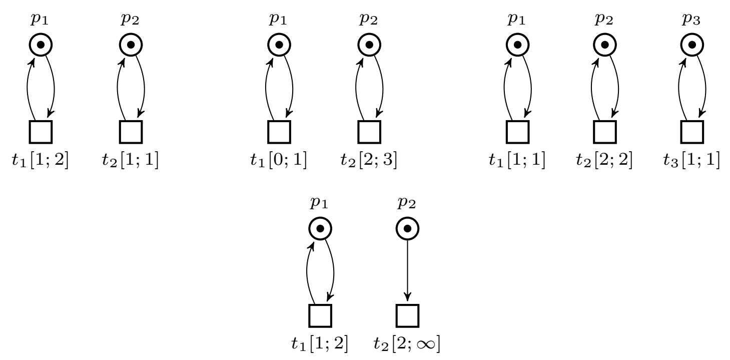


Fig. 1. Four TPN Examples

Now, we are ready to formally define the TAG associated with a marked TPN $\mathcal{N}$. It is a labeled transition system where nodes are timed aggregates. It has an initial aggregate, a set of actions (the set of transitions of $\mathcal{N}$) and a transition relation. The initial aggregate is easily computed by considering **static** information of the TPN while the transition relation is directly obtained by Definition 5 and Definition 6.

Definition 7 (Timed Aggregate Graph). *Let $\mathcal{N} = \langle P, T, Pre, Post, I, m_0\rangle$ be a TPN. The* TAG *associated with $\mathcal{N}$ is a tuple $G = \langle \mathcal{A}, T, a_0, \delta\rangle$ where:*

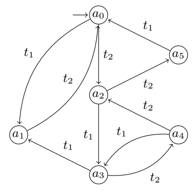
1. *$\mathcal{A}$ is a set of timed aggregates;*
2. *$a_0 = \langle m_0, h_0\rangle$ is the initial timed aggregate s.t.:*
   (a) m_0 is the initial marking of $\mathcal{N}$.
   (b) $E_0 = \{\langle t, t_{\min}, t_{\max}\rangle \mid t \in Enable(m_0)\}$
   (c) $\forall t, t \in Enable(a),\ Meet(t, t') = [t_{\min}, t_{\max}]$
3. *$\delta \subseteq \mathcal{A} \times T \times \mathcal{A}$ is the transition relation such that:*
   $\forall a \in A,\ \forall t \in T,\ (a, t, a') \in \delta$ iff $a \overset{t}{\rightsquigarrow} a'$

Since each transition having an unbounded static latest firing time will always maintain the same latest firing time at each aggregate where it is enabled, one can prove that the number of aggregates of a TAG is bounded when the corresponding TPN is bounded. Indeed, given a reachable marking m, the number of different aggregates having m as marking can be bounded by the number of possible values of its attributes. This number is finite because of the following facts: (1) if the number of the transitions that are enabled by m is e, there are $2^{|e|}$ possible subsets of old transitions; (2) for a given subset of old transitions o, the number of possible arrangements of the old transitions regarding the enabling time is at most equal to $\mid o \mid!$ (the 2^n elements corresponding to the orderings where two or more old transitions became enabled at the same time are not considered); (3) given an arrangement $t_1 \leq t_2 \leq \cdots \leq t_{|o|}$, the number of possible values of $\alpha_{t_1}^{m(t_1,t_2)}$ is at most equel to $\sum_{i=0}^{t_{1_{\min}}} (t_{1_{\max}} - i + 1)$. Similarly, the possible values of $\alpha_{t_2}^{m(t_2,t_3)}$ is equal to $\sum_{i=0}^{t_{2_{\min}}} (t_{2_{\max}} - i + 1)$, etc. Thus, the number of the possible different values of the matrix $Meet$, for this particular arrangement, is obtained by $\Pi_{j=2}^{|o|} \sum_{i=0}^{t_{j-1_{\min}}} (t_{j-1_{\max}} - i + 1)$; (4) for each enabled transition t (with $t_{max} \neq \infty$), there are at most $\sum_{i=0}^{t_{\min}} (t_{\max} - i + 1)$ different intervals that can represent the earliest and latest firing times associated with t in a given aggregate (i.e., α_t and β_t). When $t_{max} = \infty$, the number of possible time intervals associated with t is $t_{min} + 1$.

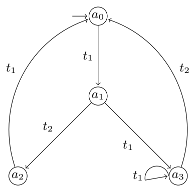
Figure 2 illustrates the TAGs corresponding to the TPNs of Figure 1. In the three first TAGs, the marking associated with each aggregate is omitted (it is the same as the initial one). The second column of the tables gives the dynamic earliest and latest firing times of the enabled transitions (i.e., t_1, t_2 and t_3 respectively). For sake of readability of the figures, the $Meet$ attribute is omitted.

Although the four models of Figure 1 are quite simple, they are representative enough to explain the TAG construction. Indeed, in the first one the transitions intervals overlap, while the case of disjoint intervals is considered through the second and the third models. Finally, the fourth model illustrates the case of an unbounded latest firing time. More significant examples are considered in Section 6.

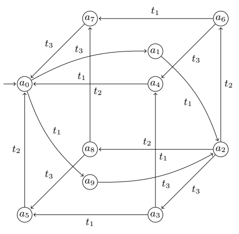
aggregate	E
a_0	$\{\langle 1,2\rangle, \langle 1,1\rangle\}$
a_1	$\{\langle 1,2\rangle, \langle 0,0\rangle\}$
a_2	$\{\langle 0,1\rangle, \langle 1,1\rangle\}$
a_3	$\{\langle 1,2\rangle, \langle 0,1\rangle\}$
a_4	$\{\langle 0,2\rangle, \langle 1,1\rangle\}$
a_5	$\{\langle 0,0\rangle, \langle 1,1\rangle\}$



aggregate	E
a_0	$\{\langle 0,1\rangle, \langle 2,3\rangle\}$
a_1	$\{\langle 0,1\rangle, \langle 1,3\rangle\}$
a_2	$\{\langle 0,0\rangle, \langle 2,3\rangle\}$
a_3	$\{\langle 0,1\rangle, \langle 0,3\rangle\}$



aggregate	E
a_0	$\{\langle 1,1\rangle, \langle 2,2\rangle, \langle 1,1\rangle\}$
a_1	$\{\langle 0,0\rangle, \langle 1,1\rangle, \langle 1,1\rangle\}$
a_2	$\{\langle 1,1\rangle\}, \langle 1,1\rangle, \langle 1,1\rangle$
a_3	$\{\langle 0,0\rangle, \langle 0,0\rangle, \langle 1,1\rangle\}$
a_4	$\{\langle 0,0\rangle, \langle 2,2\rangle, \langle 1,1\rangle\}$
a_5	$\{\langle 1,1\rangle, \langle 0,0\rangle, \langle 1,1\rangle\}$
a_6	$\{\langle 0,0\rangle, \langle 2,2\rangle, \langle 0,0\rangle\}$
a_7	$\{\langle 1,1\rangle, \langle 2,2\rangle, \langle 0,0\rangle\}$
a_8	$\{\langle 1,1\rangle, \langle 0,0\rangle, \langle 0,0\rangle\}$
a_9	$\{\langle 1,1\rangle, \langle 1,1\rangle, \langle 0,0\rangle\}$



aggregate	marking	E
a_0	(1,1)	$\{\langle 1,2\rangle, \langle 2,\infty\rangle\}$
a_1	(1,0)	$\{\langle 0,0\rangle\}$
a_2	(1,0)	$\{\langle 1,2\rangle\}$
a_3	(1,1)	$\{\langle 1,2\rangle, \langle 0,\infty\rangle\}$
a_4	(1,0)	$\{\langle 0,2\rangle\}$

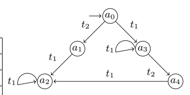


Fig. 2. TAGs associated with TPNs of Figure 1

3.2 Preservation Results

In this section, we establish the main result of our approach: The TAG is an exact representation of the reachability state space of a TPN. In fact, for each path in the TPN (resp. in the corresponding TAG) it is possible to find a path in the TAG (resp. TPN) involving the same sequence of transitions and where the time elapsed within a given state is between the minimum and the maximum stay time of the corresponding aggregate.

Theorem 1. *Let $\mathcal{N}$ be a TPN and let $G = \langle \mathcal{A}, T, a_0, \delta\rangle$ be the TAG associated with $\mathcal{N}$. Then $\forall \pi = m_0 \xrightarrow{(d_1,t_1)} m_1 \xrightarrow{(d_2,t_2)} \ldots \xrightarrow{(d_n,t_n)} m_n \xrightarrow{d_{n+1}}$, with $d_i \in \mathbb{R}_{\geq 0}$, for $i = 1 \ldots n+1$, $\exists \pi = a_0 \xrightarrow{t_1} a_1 \longrightarrow \ldots \xrightarrow{t_n} a_n$ s.t. $\forall i = 0 \ldots n$, $d_{i+1} \leq \Delta(a_i)$, $m_i = a_i.m$ and $\forall i = 1 \ldots n$, $d_i \geq \alpha_{{i-1}_{t_i}}$.*

Proof. Let $\overline{\pi} = m_0 \overset{(d_1,t_1)}{\longrightarrow} m_1 \overset{(d_2,t_2)}{\longrightarrow} \dots \overset{(d_n,t_n)}{\longrightarrow} m_n \overset{d_{n+1}}{\longrightarrow}$ be a path of $\mathcal{N}$, with $d_i \in \mathbb{R}_{\geq 0}$, for $i = 1 \dots n+1$. Given a path $a_0 \longrightarrow a_1 \dots$, we denote by α_{i_t} (res. β_{i_t}), for $i = 0 \dots$, the dynamic earliest firing time (resp. latest firing time) of a transition t enabled by an aggregate a_i.

Let us prove by induction on the length of $\overline{\pi}$ the existence of a path π in the TAG satisfying the conditions of Theorem 1.

- $\mid \overline{\pi} \mid = 0$: Obvious since $m_0 = a_0.m$ (by construction) and since d_1 is less or equal to $\min_{t \in Enable(m_0)} t_{max}$ which is exactly the value of $\Delta(a_0)$.
- $\mid \overline{\pi} \mid = 1$ i.e., $\overline{\pi} = m_0 \overset{(d_1,t_1)}{\longrightarrow} m_1 \overset{d_2}{\longrightarrow}$. It is clear that $\alpha_{0_{t_1}} \leq d_1 \leq \Delta(a_0)$. The fact that t_1 is fireable at m_0 implies that it is at a_0 ($\forall t \in Enable(m_0)$, $t_{1_{min}} \leq t_{max}$) and its firing leads to the aggregate a_1 satisfying $a_1.m = m_1$. Let us assume that $d_2 > \Delta(a_1)$ and let t_m be the transition that is enabled at a_1 and which has the smallest latest firing time i.e., $\beta_{1_{t_m}} = \Delta(a_1)$. If t_m is newly enabled at a_1 then d_2 should clearly be greater or equal to $\Delta(a_1)$. If $t_m \in \downarrow (a_0, t_1)$ then $\beta_{1_{t_m}} = t_{m_{max}} - t_{1_{min}}$. Since $d_1 \geq t_{1_{min}}$, then $t_{m_{max}} - t_{1_{min}} \geq t_{m_{max}} - d_1$. The fact that $d_2 > \beta_{1_{t_m}}$ would imply that $d_1 + d_2 > t_{m_{max}}$ which is contradictory with the STS semantics. Thus $d_2 \leq \Delta(a_1)$.
- Assume that for any path $\overline{\pi}$ s.t. $\mid \overline{\pi} \mid \leq n$, there exists a path in the TAG with the same trace and satisfying the above conditions. Let $\overline{\pi} = m_0 \overset{(d_1,t_1)}{\longrightarrow} m_1 \overset{(d_2,t_2)}{\longrightarrow} \dots \overset{(d_n,t_n)}{\longrightarrow} m_n \overset{(d_{n+1},t_{n+1})}{\longrightarrow} m_{n+1} \overset{d_{n+2}}{\longrightarrow}$ be a path of length $n+1$. Let $\pi = a_0 \overset{t_1}{\longrightarrow} a_1 \overset{t_2}{\longrightarrow} \dots \overset{t_n}{\longrightarrow} a_n$ be the path in the TAG associated with the n-length prefix of $\overline{\pi}$ (by the induction hypothesis). Then $d_{n+1} \leq \Delta(a_n)$. Let us demonstrate that $d_{n+1} \geq \alpha_{n_{n+1}}$: It is clear that this is the case when $t_{n+1} \in \uparrow (a_n, t_{n+1}$. If $t_{n+1} \in \downarrow (a_n, t_{n+1}$, let $LastNew_i(t)$ be the function that returns the greatest integer, smaller than (or equal to) i, such that $t \in \uparrow (a_{l-1}, t_l)$. If such a value does not exist, then t became enabled, for the last time, at the initial aggregate a_0 and the function returns 0. Let $k = LastNew_i(t_{n+1})$, then $\alpha_{n_{n+1}} = t_{n+1_{min}} - \sum_{i=k}^{n-1} SCR(a_i, t_{n+1})$. The STS semantics implies that $\sum_{i=k}^{n-1} SCR(a_i, t_{n+1}) \geq \sum_{i=k}^{n-1} d_{i+1}$ Thus $t_{n+1_{min}} - \sum_{i=k}^{n-1} SCR(a_i, t_{n+1}) \geq t_{n+1_{min}} - \sum_{i=k}^{n-1} d_{i+1}$, and $d_{n+1} > \alpha_{n_{t_{n+1}}}$ would means that $\sum_{i=k}^{n} d_{i+1} < t_{n+1_{min}}$ which would prevent the firing of t_{n+1} at m_n. Thus, $d_{n+1} \leq \alpha_{n_{t_{n+1}}}$. Let us show now that t_{n+1} is fireable at a_n. Assume the opposite, this would imply that there exists a transition t enabled by a_n such that $\alpha^{m(t_{n+1},t)} > \beta^{m(t,t_{n+1})}$. Let $LastNew_n(t_{n+1}) = l$, $LastNew_n(t) = k$, and let us consider the three following cases:
  1. $l = k$, then $\beta^{m(t,t_{n+1})} = t_{max}$ and $\alpha^{m(t_{n+1},t)} = t_{n_{min}}$ and the fact that $t_{n_{min}} > t_{max}$ would prevent t_{n+1} from being fireable at m_n which is not the case. Thus, t_{n+1} is fireable at a_n as well.
  2. $l < k$. In this case, $\alpha^{m(t_{n+1},t)} = t_{n+1_{min}} - \sum_{j=l}^{k-1} SCR(a_j, t_{n+1})$ and $\beta^{m(t,t_{n+1})} = t_{max}$. Again, the STS semantics implies that $\sum_{i=l}^{k-1} SCR(a_i, t_{n+1}) \geq \sum_{i=l}^{k-1} d_{i+1}$. Thus, $t_{n+1_{min}} - \sum_{i=l}^{k-1} SCR(a_i, t_{n+1}) \leq t_{n+1_{min}} - \sum_{i=l}^{k-1} d_{i+1}$, and $\alpha^{m(t_{n+1},t)} > t_{max}$ would means that $\sum_{i=l}^{k} d_{i+1} < t_{n+1_{min}}$

which would prevent the firing of t_{n+1} at m_n. Thus, $\alpha^{m(t_{n+1},t)} \leq \beta^{m(t,t_{n+1})}$ and t_{n+1} is necessarily fireable at a_n.

3. $l > k$. In this case, $\alpha^{m(t_{n+1},t)} = t_{n+1_{min}}$ and $\beta^{m(t,t_{n+1})} = t_{max} - \sum_{i=k}^{l-1} Max(0, (\alpha^{m(t_{i+1},t)} - (\beta^{m(t,t_{i+1})} - \beta_{i_t}))$. Knowing that $\sum_{i=k}^{l-1} Max(0, (\alpha^{m(t_{i+1},t)} - (\beta^{m(t,t_{i+1})} - \beta_{i_t})) \leq \sum_{i=k}^{l-1} d_{i+1}$ (otherwise, the time spent between k and some $i \leq l$ is smaller than $\alpha^{m(t_i,t)}$, which is contradictory with the recurrence hypothesis), $t_{max} - \sum_{i=k}^{l-1} Max(0, (\alpha^{m(t_{i+1},t)} - (\beta^{m(t,t_{i+1})} - \beta_{i_t})) \geq t_{max} - \sum_{i=k}^{l-1} d_{i+1}$. Thus, if $t_{n+1_{min}} > \beta^{m(t,t_{n+1})}$ then $t_{n+1_{min}} > t_{max} - \sum_{i=k}^{l-1} d_{i+1}$ which prevent the firing of t_{n+1} at m_n (before firing t) which is not true. Thus, t_{n+1} is fireable at a_n.
Let us now demonstrate that $d_{n+2} \leq \Delta(a_{n+1})$. Assume the opposite, and let t_m be the transition enabled by a_{n+1} which has the smallest latest firing time i.e. $\beta_{n+1_{t_m}} = \Delta(a_{n+1})$. It is clear that if $t_m \in\uparrow (a_n, t_{n+1})$ then $d_{n+2} \leq \Delta(a_{n+1})$. Otherwise, if $t_m \in\downarrow (a_n, t_{n+1})$ and $k = LastNew_{n+1}(t_m)$ then $\beta_{n+1_{t_m}} = t_{m_{max}} - \sum_{i=k}^{n} Max(0, (\alpha^{m(t_{i+1},t_m)}$ Again, since $\sum_{i=k}^{n} d_{i+1} \geq \sum_{i=k}^{n} Max(0, (\alpha^{m(t_{i+1},t_m)}$, then $\beta_{n+1_{t_m}} \leq t_{m_{max}} - \sum_{i=k}^{n} d_{i+1}$, and the fact that $d_{n+1} > \beta_{n+1_{t_m}}$ would imply that $d_{n+2} + \sum_{i=k}^{n} d_{i+1} > t_{m_{max}}$ which is not allowed by the STS semantics. Thus, $d_{n+2} \leq \Delta(a_{n+1})$.

Theorem 2. *Let $\mathcal{N}$ be a TPN and let $G = \langle \mathcal{A}, T, a_0, \delta \rangle$ be the TAG associated with $\mathcal{N}$. Then, for any path $\pi = a_0 \xrightarrow{t_1} a_1 \longrightarrow \ldots \xrightarrow{t_n} a_n$ in the TAG, there exists a run $\overline{\pi} = m_0 \xrightarrow{(d_1,t_1)} m_1 \longrightarrow \ldots \xrightarrow{(d_n,t_n)} m_n$ in $\mathcal{N}$, s.t. $\forall i = 0 \ldots n$, $m_i = a_i.m$, $\forall i = 1 \ldots n$, $\alpha_{i-1_{t_i}}) \leq d_i \leq \Delta(a_{i-1})$, and $\forall d \in \mathbb{R}_{\geq 0}$, $m_n \xrightarrow{d} \Leftrightarrow d \leq \Delta(a_n)$*

Proof. Let $\pi = a_0 \xrightarrow{t_1} a_1 \longrightarrow \ldots \xrightarrow{t_n} a_n$. We denote by α_{i_t} (res. β_{i_t}) the dynamic earliest firing time (resp. the dynamic latest firing time) of a transition t at aggregate a_i, for $i \in \{0, \ldots, n-1\}$. Let us demonstrate that the path $\overline{\pi} = m_0 \xrightarrow{(d_1,t_1)} m_1 \longrightarrow \ldots \xrightarrow{(d_n,t_n)} m_n$ obtained by the following algorithm satisfies the requirement. The function $LastNew_i(t)$ returns the greatest integer l, smaller than i, such that $t \in\uparrow (a_{l-1}, t_l)$. If such a value does not exist, then t became enabled, for the last time, at the initial aggregate a_0 and the function returns 0. We propose to proceed by construction and built a path $\overline{\pi}$ satisfying the Theorem 2. We use the following algorithm to compute a set of delays d_i, for $i = 1 \ldots n$ and prove that the $a_0.m \xrightarrow{(d_1,t_1)} a_1.m \longrightarrow \ldots \xrightarrow{(d_n,t_n)} a_n.m$ is a run of the TPN associated with the TAG.

```
Input: an abstract path $\pi = a_0 \xrightarrow{t_1} a_1 \longrightarrow \ldots \xrightarrow{t_n} a_n$
Output: a concrete path $\overline{\pi} = m_0 \xrightarrow{(d_1,t_1)} m_1 \longrightarrow \ldots \xrightarrow{(d_n,t_n)} m_n$
begin
1    $\forall i = 1 \ldots n$
2        $d_i \leftarrow \alpha_{i-1_{t_i}}$
3    $\forall i = n-1 \ldots 1$
4     $k = LastNew_i(t_{i+1})$
5    If($\sum_{j=k}^{i-1} d_{j+1} < t_{i+1_{min}}$)
```

```
6         $\forall k \leq j < i$
7             $d_{j+1} = Max(d_{j+1}, \alpha_{j_{t_{i+1}}} - \alpha_{j+1_{t_{i+1}}})$
8     If$(\sum_{j=k}^{i-1} d_{j+1} > t_{i+1_{\max}})$
9         $\forall k \leq j < i$
10            $d_{j+1} = Min(d_{j+1}, \alpha_{j_{t_{i+1}}} - \alpha_{j+1_{t_{i+1}}})$
11    $\forall t \in Enable(a_i.m)$
12     $l = LastNew_i(t)$
13     If$((k > l) \wedge (t_{\max} - \sum_{j=l}^{k-1} d_{j+1} < t_{i+1_{\min}}))$
14         $\forall l \leq j < k$
15                $d_{j+1} = Min(d_{j+1}, \alpha_{j_t} - \alpha_{j+1_t})$
end
```

The intuition of the above algorithm is to build a concrete path $\overline{\pi}$ guided by the abstract path π. $\overline{\pi}$ is built by traversing π by backtracking. Initially (lines $1-2$), the stay time at each marking is set to the minimum i.e., as soon as the desired transition is fireable. Then, starting from the last aggregate, each time a transition t_{i+1} is fired from an aggregate a_i (for $i = 1 \ldots n$), the firability conditions are ensured by (possibly) changing the time that is elapsed before reaching a_i. Roughly speaking, two conditions must be satisfied in order to make the transition t_{i+1} fireable from a_i (i.e., $a_i.m$): (1) The first condition is that the elapsed time, since t_{i+1} became enabled for the last time, belongs to the interval $[t_{i+1_{min}}, t_{i+1_{max}}]$ while the second condition (2), is that there is no transition t enabled by a_i that prevents the firing of t_{i+1}. The only way for the last condition to be satisfied is that t has been enabled (for the lat time) before t_{i+1} and the elapsed time between the moment t became enabled and t_{i+1} became enabled is strictly greater than $t_{max} - t_{i+1_{min}}$. The first condition is treated at lines $5-10$: If the elapsed time since t_{i+1} has been enabled for the last time and the current state $(a_i.m)$ is strictly smaller than $t_{i+1_{min}}$ (lines $5-7$) then it must be increased without exceeding $t_{i+1_{max}}$. This is ensured by the fact that, by construction of the TAG, $\sum_{j=k}^{i-1}(\alpha_{j_{t_{i+1}}} - \alpha_{j+1_{t_{i+1}}}) = T_{i+1_{min}}$ and $\sum_{j=k}^{i-1}(\alpha_{j_t} - \alpha_{j+1_t}) \leq t_{i+1_{max}}$ for any transition $t \in\downarrow (a_{i-1}, t_i)$. Now, the elapsed time since the last time t_{i+1} became enabled can exceed $t_{i+1_{max}}$. This can occur if, in order to ensure the firing of some transition t_j (for $j > i+1$) this time has been increased by the algorithm (lines $5-7$). Thus, one has to decrease this time while maintaining the firability of the transition t_j. This is ensured by lines $8-10$. The last condition that could prevent t_{i+1} from being fireable at a_i. is that condition (2) is violated: the time elapsed between the moment some transition t, enabled before, t_{i+1}, and the moment t_{i+1} became enabled is bigger than $t_{max} - t_{i+1_{min}}$. This can happen when the firing of some transition t_j, with $j > i+1$, involved the increase of this quantity of time. This case is treated at lines $11-15$, by fixing this problem while maintaining the future firability of t_j.

Thus, the algorithm ensures the construction of a run of the TPN associated with the TAG that has the same trace. It is clear that the values of d_i, for $i = 1 \ldots n$, respects the conditions of Theorem 2. Now, Theorem 1 ensures that if $m_n \xrightarrow{d}$, for some $d \in \mathbb{R}_{\geq 0}$, then $d \leq \Delta(a_n)$. Finally, given $d \in \mathbb{R}_{\geq 0}$ s.t.,

$d \leq \Delta(a_n)$, the algorithm used to build $\overline{\pi}$ implies that the involved markings are reached as soon as possible. By construction of the TAG, $\Delta(a_n)$ is the maximum time the system can stay at m_n.

Using the above results one can use the TAG associated with a TPN in order to analyse both event and state based properties. In particular, we can check whether a given marking (resp. transition) is reachable (resp. is fireable) before (or after) some time.

4 Checking Time Reachability Properties

Our ultimate goal is to be able, by browsing the TAG associated with a TPN, to check timed reachability properties. For instance, we might be interested in checking whether some state-based property φ is satisfied within a time interval $[d, D)$, with $d \in \mathbb{N}$ and $D \in (\mathbb{N} \cup \{\infty\})$, starting from the initial marking. The following usual reachability properties belong to this category.

1. $\exists\Diamond_{[d;D]}\varphi$: There exists a path starting from the initial state, consuming between d and D time units and leading to a state that satisfies φ.

2. $\forall\Box_{[d;D]}\varphi$: For all paths starting from the initial state, all the states, that are reached after d and before D time units, satisfy φ.

3. $\forall\Diamond_{[d;D]}\varphi$: For all paths starting from the initial state, there exists a state in the path, reached after d and before D time units that satisfies φ.

4. $\exists\Box_{[d;D]}\varphi$: There exists a path from the initial state where all the states, that are reached after d and before D time units, satisfy φ.

For the verification of time properties, an abstraction-based approach should allow the computation of the minimum and maximum elapsed time over any path. In the following, we establish that the TAG allows such a computation.

Definition 8. *Let $\mathcal{N}$ be a TPN and let $G = \langle \mathcal{A}, T, a_0, \delta \rangle$ be the corresponding TAG. Let $\pi = a_0 \xrightarrow{t_1} a_1 \longrightarrow \ldots \xrightarrow{t_n} a_n$ be a path in G. For each aggregate a_i (for $i = 0 \ldots n$), $MinAT_\pi(a_i)$ (resp. $MaxAT_\pi(a_i)$) denotes the minimum (resp. maximum) elapsed time between a_0 and a_i. In particular, $MinAT(a_0) := 0$ and $MaxAT(a_0) := \Delta(a_0)$.*

Proposition 1. *Let $\mathcal{N}$ be a TPN and let $G = \langle \mathcal{A}, T, a_0, \delta \rangle$ be the corresponding TAG. Let $\pi = a_0 \xrightarrow{t_1} a_1 \longrightarrow \ldots \xrightarrow{t_n} a_n$ be a path in G. We denote by α_{i_t} (resp. β_{i_t}) the dynamic earliest (resp. latest) firing time of a transition t at aggregate a_i, for $i = 1 \ldots n$. Then, $\forall i = 1 \ldots n$, the following holds:*

- $MinAT_\pi(a_i) = MinAT_\pi(a_{i-1}) + \alpha_{i-1_{t_i}}$
- $MaxAT_\pi(a_i) = MaxAT(a_{i-1}) + Min_{t \in Enable(a_i)} SCR(a_i, t)$

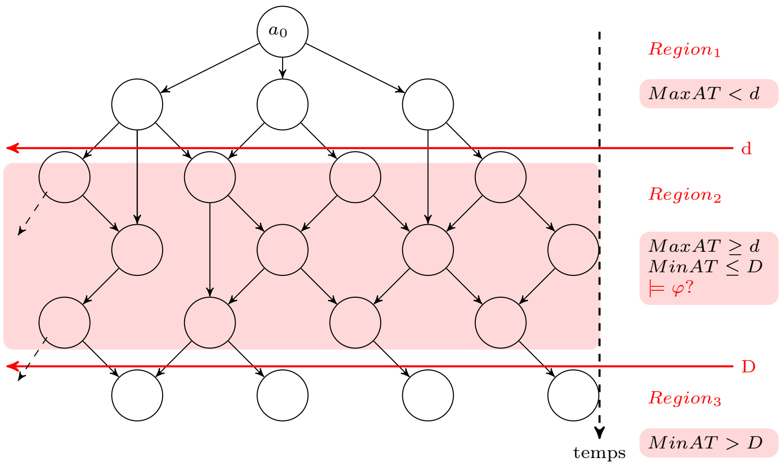


Fig. 3. Reachability analysis on the TAG

Using the previous proposition, one can browse the TAG graph and compute the minimum and maximum bounds of the elapsed time of the current path on-the-fly. If a path of the TAG is considered as a counterexample for some time reachability property, one can use the algorithm given in the proof of Theorem 2 in order to build a concrete counterexample. Here we do not give the detailed algorithms for checking reachability properties on-the-fly, but we give the main intuition. The TAG is represented as a tree which is partitioned into three regions (see. Figure 3). The first region ($Region_1$) contains the aggregates that are reachable strictly before d time units. The second region ($Region_2$) contains the aggregates that are reachable between d and D time units and the last region contains the aggregates that are reachable strictly after D time units. In case $D = \infty$ $Region_3$ is empty. By doing so, the verification algorithms behave as follows: only aggregates belonging to $Region_2$ are analyzed with respect to φ. $Region_1$ must be explored in order to compute the maximal and minimum access time of the traversed aggregates, but $Region_3$ is never explored. In fact, as soon as an aggregate is proved to belong to $Region_3$ the exploration of the current path is stopped. Furthermore, one has to check for a particular kind of Zeno behavior: if a cycle involving only aggregates whose minimal and maximal access times are equal, then the exploration of the current branch is stopped.

For instance checking the formula number 1 is reduced to the search of an aggregate a in $Region_2$ that satisfies φ. As soon as such an aggregate is reached the checking algorithm stops the exploration and returns *true*. When, all the aggregates of $Region_2$ are explored (none satisfies φ) the checking algorithm returns *false*. Dually, the formula number 2 is proved to be unsatisfied as soon as an aggregate in $Region_2$ that do not satisfy φ is reached. When all the aggregates of $Region_2$ are explored (each satisfies φ) the checking algorithm returns *true*.

Checking formulae number 3 and 4 is slightly more complicated. In fact, checking formula number 3 is reduced to check if, along any path in $Region_2$, there exists at least one aggregate satisfying φ. As soon as a path in $Region_2$ is completely explored without encountering an aggregate satisfying φ, the exploration is stopped and the checking algorithm returns *false*. Otherwise, it returns *true*. Finally, checking formula 4 is reduced to check that there exists a path in $Region_2$ such that all the aggregates belonging to this path satisfy φ. This formula is proved to be true as soon as such a path is found. Otherwise, when all the paths of $Region_2$ are explored (none satisfies the desired property), the checking algorithm returns *false*.

5 Related works

This section reviews the most known techniques, proposed in the literature, that abstract and analyse the state space of real-time systems described by means of TPN. Abstraction techniques aim at constructing, by removing some irrelevant details, a contraction of the state space of the model, which preserves properties of interest. The existing abstraction approaches mainly differ in the states agglomeration criteria, the characterization of states and state classes (interval states or clock states), the kind of preserved properties.

The *States Class Graph* (SCG) [3] was the first method of state space representation adapted to TPNs. A class (m, D) is associated with a marking m and a time domain D represented by a set of inequalities over variables. The variables represented in the SCG are the firing time intervals of enabled transitions. The SCG allows for the verification of some TPN properties like reachability, boundness. However, it preserves the linear time properties only. To address this limitation, a refinement of the method was proposed in [24], in the form of a graph called *Atomic States Class Graph* (ASCG). The authors use a cutting of state class by adding linear constraints so that each state of an atomic class has a successor in all the following classes. With this improvement, they are able to verify CTL* properties on TPN, but with the limitation that the time intervals of transitions are bounded. A new approach for the construction of atomic classes was proposed in [4] and allows the verification of CTL* without restriction on time intervals. The state class approach is implemented in a software tool called TINA [5].

The *Zones Based Graph* (ZBG) [10] is an other approach allowing to abstract the TPN state space. This approach is inspired by the *Region Graph* (RG) [1] technique, initially introduced for *timed automata*. In practice, the number of regions is too large for an effective analysis, thus, the regions are grouped into a set of *zones*. A zone is a convex union of regions and can be represented by a DBM (Difference Bound Matrix) [8]. In [10], the clocks of transitions are directly encoded within the zones. This allows to verify temporal and quantitative properties but not CTL* properties. As for timed automata, a disadvantage of the method is the necessary recourse to approximation methods (k-approximation or kx-approximation) in the case where the infinity is used in the bounds of time intervals. Lime and Roux also used TPNs to model system behavior [17]. They used the state class approach to build a *timed automaton* that preserves the behavior of the TPN using as less clock variables as possible. The resulting model is then verified using the UPPAAL tool [16]. However, even though UPPAAL can answer about quantitative temporal properties, it can only verify a subset of TCTL. Adding a new transition to measure time elapse was proposed in [6] to perform TCTL model-checking in TPNs. Using this transition, TCTL formulae are translated into CTL formulae. Then a ZBG for TPN is refined leading to a graph called *Atomic Zone Based Graph* (AZBG) that preserves CTL properties.

Unlike the TAG, in all existing approaches, the time information does not appear explicitly in nodes which leads to additional and costly calculations such as: the manipulation of DBM to encode the zones (for zones based approaches) and the classes (for state-class based approaches), the approximations to counter the problem of unbounded transitions, conversion of graphs to timed automata (using UPPAAL) to model check properties (etc). In our work the time information is encoded within the aggregates allowing to check time properties just by browsing the graph, which has a significant impact on the construction complexity. The encoding of the timing information in the aggregates is such that the minimum and maximum elapsed time in every path of the TAG can be computed.

6 Experimental results

The efficiency of the verification of timed reachability properties is closely linked with the size of the explored structure to achieve this verification. Thus, it was important to first check that the TAG is a suitable/reduced abstraction before performing verification on it. Our approach for building TAG-TPN was implemented in a prototype tool (written in C++), and used for experiments in order to validate the size of the graphs generated by the approach (note that the prototype was not optimized for time efficiency yet, therefore no timing figures are given in this section). All results reported in this section have been obtained on a Mac-os with 2 gigahertz Intel with 8 gigabytes of RAM. The implemented prototype allowed us to have first comparison with existing approaches with respect to the size of obtained graphs. This section is dedicated to report, compare and discuss the experimental results obtained with three approaches: SCG, ZBG and TAG-TPN. Notice that we used the ROMEO tool to build both SCGs and ZBGs. The built versions preserve Linear-time Temporal Logic (LTL) properties. We tested our approach on several TPN models and we report here the obtained results for three well known examples of parametric TPN models.

The considered models are: (1) a TPN representing a composition of producer/consumer models by fusion of a single buffer (of size 5) [12], (2) the second example (adapted from [19]) is the Fischer's protocol for mutual exclusion, and (3) the last is the train crossing example [4].

Table1 reports the results obtained with the SCG, the ZBG and the TAG-TPN approaches, in terms of graph size number of nodes/number of edges).

Parameters	SCG (with Tina) (nodes / arcs)	ZBG (with Romeo) (nodes / arcs)	TAG-TPN (nodes / arcs)
Nb. prod/cons	TPN model of producer/consumer		
1	34 / 56	34 / 56	34 / 56
2	748 / 2460	593 / 1 922	740 / 2438
3	4604 / 21891	3240 / 15200	4553 / 21443
4	14086 / 83375	9504 / 56038	13878 / 80646
5	31657 / 217423	20877 / 145037	30990 / 207024
6	61162 / 471254	39306 / 311304	60425 / 449523
7	107236 / 907 708	67224 / 594795	106101 / 856050
Nb. processes	Fischer protocol		
1	4 / 4	4 / 4	4 / 4
2	18 / 29	19 / 32	20 / 32
3	65 / 146	66 / 153	74 / 165
4	220 / 623	221 / 652	248 / 712
5	727 / 2536	728 / 2 615	802 / 2825
6	2378 / 9154	2379 / 10098	2564 / 10728
7	7737 / 24744	7738 / 37961	8178 / 39697
8	25080 / 102242	25081 / 139768	26096 / 144304
Nb. processes	Train crossing		
1	11 / 1 4	11 / 1 4	11 / 14
2	123 / 218	114 / 200	123 / 218
3	3101 / 7754	2817 / 6944	2879 / 7280
4	134501 / 436896	122290 / 391244	105360 / 354270

Table 1. Experimentation results

The obtained preliminary results show that the size of the TAG is comparable to the size of the graphs obtained with the ZBG and the SCG approaches. The TAG achieves better performances than both SCG and ZBG for the train crossing example, while it is slightly worse for the Fischer's protocol and performs similarly to SCG but worse than ZBG for the producer/consumer example.

This is an encouraging result because of the following reasons: The TAG allows for checking *timed* properties while the SCG approach do not. Also, it can be used for the verification of event-based timed properties while the ZBG approach do not. An other difference consists in the fact that the verification of timed properties can be achieved directly on the TAG, without any synchronisation with an additional automaton (representing the formula to be checked), nor any prior step of translation to timed automata. Moreover, using the algorithm given in the proof of Theorem 2, and in the prospect of using the TAG in order to check timed properties, one can exhibit (e.g., from a counterexample abstract path in the TAG) an explicit run involving the time spent at each reached marking. Finally, we claim that the TAG is a suitable abstraction for further reductions, especially the partial order reduction which is based on the exploitation of the independency between the TPN transitions. The third example of Figure 2 is a typical illustration of the gain one could have by applying such a reduction.

7 Conclusion

We proposed a new symbolic graph for the abstraction of the TPN state space. The proposed graph, called TAG, produces a finite representation of the bounded TPN behavior and allows for analyzing of timed reachability properties. Unlike, the existing approaches, our abstraction can be directly useful to check *timed* logic properties. We think that our approach is more understandable than the SCG and the ZBG approaches (the two main approaches for TPNs analysis since three decades) and easily implementable. Another feature of our approach is that each path of the TAG can be matched with a concrete path of the TPN model where the elapsed time at each encountered state is exhibited.

Our ultimate goal is to use the TAG traversal algorithm for the verification of timed reachability properties expressed in the $TCTL$ logic. Several issues have to be explored in the future: We first have to improve our implementation so that time consumption criterion can be taken into account in our comparison to existing tools. We should also, carry out additional experimentations (using more significant use cases) to better understand the limits of our approach and to better compare the TAG technique to the existing approaches. Second, we believe that partial order reduction techniques can be used to reduce the size of the TAG while preserving time properties but without necessarily preserving all the paths of the underlying TPN. Finally, two challenging perspectives can be considered in the future: (1) the design and the implementation of model checking algorithms for verification of $TCTL$ formulae, and (2), the extension of our approach to timed automata.

References

1. R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
2. B. Berthomieu and M. Diaz. Modeling and Verification of Time Dependent Systems Using Time Petri Nets. *IEEE Trans. Software Eng.*, 17(3):259–273, 1991.
3. B. Berthomieu and M. Menasche. An Enumerative Approach for Analyzing Time Petri Nets. In *IFIP Congress*, pages 41–46, 1983.
4. B. Berthomieu and F. Vernadat. State Class Constructions for Branching Analysis of Time Petri Nets. In *TACAS 2003*, volume 2619 of *LNCS*, pages 442–457. Springer, 2003.
5. B. Berthomieu and F. Vernadat. Time Petri Nets Analysis with TINA. In *QEST*, pages 123–124, 2006.
6. H. Boucheneb, G. Gardey, and O. H. Roux. TCTL Model Checking of Time Petri Nets. *J. Log. Comput.*, 19(6):1509–1540, 2009.
7. M. Boyer and O. H. Roux. Comparison of the Expressiveness of Arc, Place and Transition Time Petri Nets. In *ICATPN 2007*, volume 4546 of *LNCS*, pages 63–82. Springer, 2007.

8. D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems*, pages 197–212. Springer-Verlag, 1990.
9. G. Gardey, D. Lime, M. Magnin, and O. (h. Roux. Roméo: A Tool for Analyzing time Petri nets. In *In Proc. CAVÕ05, vol. 3576 of LNCS*, pages 418–423. Springer, 2005.
10. G. Gardey, O. H. Roux, and O. F. Roux. Using Zone Graph Method for Computing the State Space of a Time Petri Net. In *FORMATS 2003*, volume 2791 of *LNCS*, pages 246–259. Springer, 2003.
11. R. Hadjidj and H. Boucheneb. Improving state class constructions for CTL* model checking of time Petri nets. *STTT*, 10(2):167–184, 2008.
12. R. Hadjidj and H. Boucheneb. On-the-fly TCTL model checking for time Petri nets. *Theor. Comput. Sci.*, 410(42):4241–4261, 2009.
13. K. Klai, N. Aber, and L. Petrucci. To appear in a new approach to abstract reachability state space of time petri nets. In *To appear in 20th International Symposium on Temporal Representation and Reasoning, TIME 2013*, 2013.
14. K. Klai, N. Aber, and L. Petrucci. Verification of reachability properties for time petri nets. In *Reachability Problems - 7th International Workshop, RP 2013*, volume 8169 of *Lecture Notes in Computer Science*, pages 159–170. Springer, 2013.
15. K. G. Larsen, P. Pettersson, and W. Yi. Model-checking for real-time systems. In *FCT '95*, volume 965 of *LNCS*, pages 62–88. Springer, 1995.
16. K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL: Status and Developments. In *CAV*, pages 456–459, 1997.
17. D. Lime and O. H. Roux. Model Checking of Time Petri Nets Using the State Class Timed Automaton. *Discrete Event Dynamic Systems*, 16(2):179–205, 2006.
18. P. M. Merlin and D. J. Farber. Recoverability of modular systems. *Operating Systems Review*, 9(3):51–56, 1975.
19. W. Penczek, A. Pólrola, and A. Zbrzezny. SAT-Based (Parametric) Reachability for a Class of Distributed Time Petri Nets. *T. Petri Nets and Other Models of Concurrency*, 4:72–97, 2010.
20. C. A. Petri. Concepts of net theory. In *MFCS'73*. Mathematical Institute of the Slovak Academy of Sciences, 1973.
21. M. Pezzè and M. Young. Time Petri Nets: A Primer Introduction. In *Tutorial at the Multi-Workshop on Formal Methods in Performance Evaluation and Applications*, 1999.
22. C. Ramchandani. Analysis of asynchronous concurrent systems by timed Petri nets. Technical report, Cambridge, MA, USA, 1974.
23. J. Sifakis. Use of Petri nets for performance evaluation. *Acta Cybern.*, 4:185–202, 1980.
24. T. Yoneda and H. Ryuba. CTL model checking of time Petri nets using geometric regions. 1998.