Verification of Reachability Properties for Time Petri Nets

Kais Klai[1,2], Naim Aber[2], and Laure Petrucci[2]

[1] Institut TELECOM SudParis, CNRS UMR Samovar
9 rue Charles Fourier 91011 Evry, France
kais.klai@telecom-sudparis.eu

[2] LIPN, CNRS UMR 7030, Université Paris 13
99 avenue Jean-Baptiste Clément
F-93430 Villetaneuse, France
{naim.aber,kais.klai,laure.petrucci}@lipn.univ-paris13.fr

Abstract. This paper deals with verification of reachability properties on Time Petri Nets (TPN). TPNs allow the specification of real-time systems involving timing constraints explicitly. The main challenge of the analysis of such systems is to construct a finite abstraction of the corresponding (infinite) state graph preserving timed properties. Thus, we propose a new finite graph, called Timed Aggregate Graph (TAG), abstracting the behaviour of bounded TPNs with strong time semantics. The main feature the TAG compared to existing approaches is the encoding of the time information within the nodes of this graph. This allows 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 which allows for the verification of both event- and state-based properties.

Keywords: Time Petri Nets, Reachability properties, Model Checking.

1 Introduction

Time Petri nets are one of the most used formal models for the specification and the verification of systems where the explicit consideration of time is primordial. The main extensions of Petri nets with time are *time Petri nets* [14] and *timed Petri nets* [18]. In the first, a transition can fire within a time interval whereas, in the second, 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) [19]. The same holds for time Petri nets [7]. In [17], 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 [17]. 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. In contrast to that, the *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 [2,20,4,8,5,13,6,9] are either refinements or 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 with similar behaviour 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 contribution for the abstraction and the verification of timed systems and especially those modelled by bounded TPNs.

This paper is organised 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. In Section 4, we propose algorithms for the verification of some usual time properties based on TAGs. In Section 5, we discuss the experimental results obtained with our implementation compared to two well-known tools with respect to the size of the obtained abstraction size. Finally, a conclusion and some perspectives are given in Section 6.

2 Preliminaries and Basic Notations

A t-time Petri net (TPN for short) is a P/T Petri net [16] 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 *where:*
  - P *is a finite set of places;*
  - T *is a finite set of transitions with* $P \cap T = \emptyset$ *;*
  - $Pre : T \longrightarrow \mathbb{N}^P$ *is the backward incidence mapping;*
  - $Post : T \longrightarrow \mathbb{N}^P$ *is the forward incidence mapping;*
- $I : T \longrightarrow \mathbb{N} \times (\mathbb{N} \cup \{+\infty\})$ *is the* time interval function *such that:* $I(t) = (t_{\min}, t_{\max})$*, with* $t_{\min} \leq t_{\max}$*, 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 $m \geq Pre(t)$ and $Enable(m) = \{t \in T : m \geq Pre(t)\}$ 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, $\uparrow(m, t_i) = \{t \in T \mid (m - Pre(t_i) + Post(t_i)) \geq Pre(t) \wedge (m - Pre(t_i)) < Pre(t)\}$. 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, $\downarrow(m, t_i) = \{t \in T \mid (m - Pre(t_i) + Post(t_i)) \geq Pre(t) \wedge (m - Pre(t_i)) \geq Pre(t)\}$ 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) [12] which are usual transition systems with two types of labels: discrete labels for events (transitions) and positive real labels for time elapsing (delay). States of the TTS are pairs $s = (m, V)$ where m is a marking and $V : T \longrightarrow \mathbb{R}_{\geq 0} \cup \{\bot\}$ a time valuation. 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 firable in s iff $t \in \mathit{Enable}(m) \wedge V(t) \neq \bot \wedge t_{min} \leq V(t) \leq t_{\max}$.

Definition 2 (Semantics of a TPN). *Let $\mathcal{N} = \langle P, T, \mathit{Pre}, \mathit{Post}, I, m_0\rangle$ 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:*

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

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

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

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

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

First, the delay transitions respect the STS semantics: an enabled transition must fire within its firing interval unless disabled by the firing of others. Second, 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 firable.

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 $\mid \pi \mid$. 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, α_i and α_{i+1} are not both in $\mathbb{R}_{\geq 0}$. Then, π can be written, involving the reachable markings of $\mathcal{N}$, as $\pi = m_0 \xrightarrow{(d_1, t_1)} \ldots$ s.t. 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 **.** denotes the concatenation operator between paths and π^i, for $i = 0, 1 \ldots$, denotes the suffix of π starting at state s_i.

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

In the following, we define, for a given finite run π of a TPN $\mathcal{N}$, the time elapsed before reaching (resp. firing) a marking (resp. a transition) belonging to this run.

Definition 3. *Let $\mathcal{N}$ be a TPN and let $\pi = m_0 \overset{(d_1,t_1)}{\longrightarrow} \ldots \overset{(d_n,t_n)}{\longrightarrow} m_n$ (resp. $\pi = m_0 \overset{(d_1,t_1)}{\longrightarrow} \ldots \overset{(d_n,t_n)}{\longrightarrow} m_n \xrightarrow{d_{n+1}}$) be a run of $\mathcal{N}$. The access (resp. firing) time of marking m (resp. transition t) in π, denoted $AT_\pi(m)$ (resp. $FT_\pi(t)$), is defined as follows:*

- $AT_\pi(m_0) = 0$
- $\forall 1 \leq i \leq n,\ AT_\pi(m_i) = FT_\pi(t_i) = \sum_{k=1}^{i} d_k$.

3 Timed Aggregate Graph

In this section, 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 that time information is encoded inside aggregates. It includes the time the system is able to stay in the aggregate as well as a dynamic interval associated with each enabled transition. The first feature allows to encapsulate the delay transitions of the corresponding TTS (the arcs of a TAG are labeled with transitions of the corresponding TPN only), while the second allows to dynamically update the earliest and latest firing times of enabled transitions. It also allows to maintain the relative differences between the firing times of transitions.

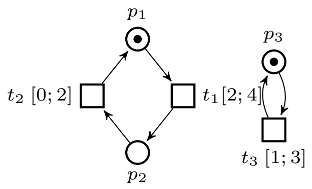
Before we formally define the TAG and illustrate how the attributes of an aggregate are computed, let us first formally define aggregates.

Definition 4 (Timed Aggregate). *A* timed aggregate *associated with a* TPN $\mathcal{N} = \langle P, T, Pre, Post, I \rangle$ *is a 4-tuple* $a = (m, E, h, H)$*, where:*

- *m is a marking*
- $E = \{\langle t, \alpha_t, \beta_t \rangle \mid t \in \textit{Enable}(m),\ \alpha_t \in (\mathbb{Z} \cup \{-\infty\}) \wedge \beta_t \in \mathbb{N} \cup \{+\infty\}\}$ *is a set of enabled transitions each asssociated with two time values.*
- $h = \min_{\langle t,\alpha_t,\beta_t \rangle \in E}(max(0, \alpha_t))$*: the minimum time the system can stay in a*
- $H = \min_{\langle t,\alpha_t,\beta_t \rangle \in E}(\beta_t)$*: the maximum time the system can stay in a*

Each aggregate is characterised by three attributes that are computed dynamically: first, a marking m. Second, a set E of enabled transitions, each associated with two time values. For a given enabled transition t, α_t represents the minimum time the system should wait before firing t and β_t represents the maximum time the system can delay the firing of t. Note that, α_t can be negative which means that t can be fired immediately. Otherwise, α_t represents the earliest firing time of t. Starting from this aggregate, the firing of t can occur between $max(0, \alpha_t)$ and β_t. In the rest of this paper, α_t will be abusively called the *dynamic* earliest firing time of t and β_t its *dynamic* latest firing time. Finally, the h and H attributes represent the minimum and the maximum time, respectively, the system can spend at the current aggregate.

Figure 1 illustrates an example of an aggregate. The associated marking is the marking of the left hand TPN. In this figure, we assume that this aggregate is the initial one. The h (resp. H) attribute corresponds to the minimum earliest firing time (resp. latest firing time) of the enabled transitions. Enabled transitions are associated with their static time intervals.



(a) Time Petri Net

$m = (1, 0, 1)$
$h = 1$
$H = 3$
$E = \{\langle t_1, 2, 4\rangle, \langle t_3, 1, 3\rangle\}$

(b) Aggregate a

Fig. 1. Example of aggregate

The TAG 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.

Definition 5 (Timed Aggregate Graph). *A* TAG *associated with a TPN* $\mathcal{N} = \langle P, T, Pre, Post, I, m_0\rangle$ *is a tuple* $G = \langle \mathcal{A}, T, a_0, \delta\rangle$ *where:*

1. $\mathcal{A}$ *is a set of timed aggregates;*
2. $a_0 = \langle m_0, h_0, H_0, E_0\rangle$ *is the initial timed aggregate s.t.:*
   (a) m_0 *is the initial marking of* $\mathcal{N}$.
   (b) $h_0 = \min_{t\in Enable(m_0)}(t_{\min})$
   (c) $H_0 = \min_{t\in Enable(m_0)}(t_{\max})$
   (d) $E_0 = \{\langle t, t_{\min}, t_{\max}\rangle \mid t \in Enable(m_0)\}$
3. $\delta \subseteq \mathcal{A} \times T \times \mathcal{A}$ *is the transition relation such that:*
   for an aggregate $a = \langle m, h, H, E\rangle$*, a transition* t *s.t.* $\langle t, \alpha_t, \beta_t\rangle \in E$ *and an aggregate* $a' = \langle m', h', H', E'\rangle$, $(a, t, a') \in \delta$ *iff the following holds:*
   (a) $m' = m - Pre(t) + Post(t)$
   (b) $\alpha_t \le H$
   (c) $\forall \langle t', \alpha_{t'}, \beta_{t'}\rangle \in E$,
   $t_{min} > t'_{max} \Rightarrow (t_{min} - \alpha_t) - (t'_{min} - \alpha_{t'}) \ge (t_{min} - t'_{max})$
   (d) $E' = E'_1 \cup E'_2$*, where:*
   • $E'_1 = \bigcup_{t'\in\uparrow(a,t)}\{\langle t', t'_{\min}, t'_{\max}\rangle\}$
   • $E'_2 = \bigcup_{t'\in\downarrow(a,t)}\{\langle t', \alpha_{t'} - H, \beta_{t'} - max(0, \alpha_t)\rangle\}$
   (e) $h' = \min_{\langle t', \alpha_{t'}, \beta_{t'}\rangle\in E'}(max(0, \alpha_{t'}))$
   (f) $H' = \min_{\langle t', \alpha_{t'}, \beta_{t'}\rangle\in E'}(\beta_{t'})$

Given an aggregate $a = \langle m, h, H, E \rangle$ and a transition $t \in T$, t is said to be enabled by a, denoted by $a \xrightarrow{t}$, iff: (1) $\exists(\alpha_t, \beta_t) \in (\mathbb{Z} \cup \{-\infty\}) \times (\mathbb{N} \cup \{+\infty\})$ s.t. $(t, \alpha_t, \beta_t) \in E$ and $\alpha_t \leq H$, and, (2) there is no other transition t', enabled by $a.m$, that should be fired before t. In fact, the first condition is not sufficient when t_{min} is greater than t'_{max} for some other transition t'. The firing of t from a leads to a new aggregate $a' = \langle m', h', H', E' \rangle$ whose attributes are computed as follows:

- The elements of E' are processed by taking the transitions that are enabled by m' and computing their earliest and latest firing times depending on their membership to $\uparrow(a, t)$ and $\downarrow(a, t)$. For each transition $t' \in Enable(a'.m)$, if t' is newly enabled, then its dynamic earliest and latest firing times are statically defined by t'_{min} and t'_{max} respectively. Otherwise, let $\langle t', \alpha_{t'}, \beta_{t'} \rangle \in a.E$ and $\langle t', \alpha'_{t'}, \beta'_{t'} \rangle \in a'.E$, then the maximum time elapsed by the system at $a.m$ (i.e., $a.H$) is subtracted from $\alpha_{t'}$ and the minium time is substracted from $\beta_{t'}$. Indeed, more the system can stay in $a.m$ less it can stay in $a'.m$ (and vice versa). Thus, the earliest firing time of t starting from a' is $max(0, \alpha_{t'} - a.H)$ while its latest firing time is $\beta_{t'} - max(0, \alpha_t)$.
- The computation of $a'.h$ (resp. $a'.H$) is ensured by taking the minimum of the dynamic earliest (resp. latest) firing time of enabled transitions.

According to Definition 5, the dynamic earliest firing time of a transition can decrease infinitely which could lead to an infinite state space TAG. Thus, an equivalence relation allowing to identify equivalent aggregates has been introduced in [11]. This equivalence relation is used in the construction of a TAG so that each newly built aggregate is not explored as long as an already built equivalent aggregate has been. Moreover, in [11], we have established that the TAG, built under this equivalence relation is finite when the corresponding TPN is bounded. We also demonstrated that the TAG is an exact representation of the reachability state space of a TPN. For each path in the TAG (resp. in the TPN) it is possible to find a path in the TPN (resp. TAG) 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.

Figure 2 illustrates the TAG corresponding to the TPN of Figure 1.

For the verification of time properties, an abstraction-based approach should allow the computation of the minimum and maximum elapsed time over any path.

Definition 6. *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 $MaxAT(a_0) = a_0.H$.*
- *For each transition t_i (for $i = 1 \ldots n$), $MinFT_\pi(t_i)$ (resp. $MaxFT_\pi(t_i)$) denotes the minimum (resp. maximum) elapsed time before firing t_i.*

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:*

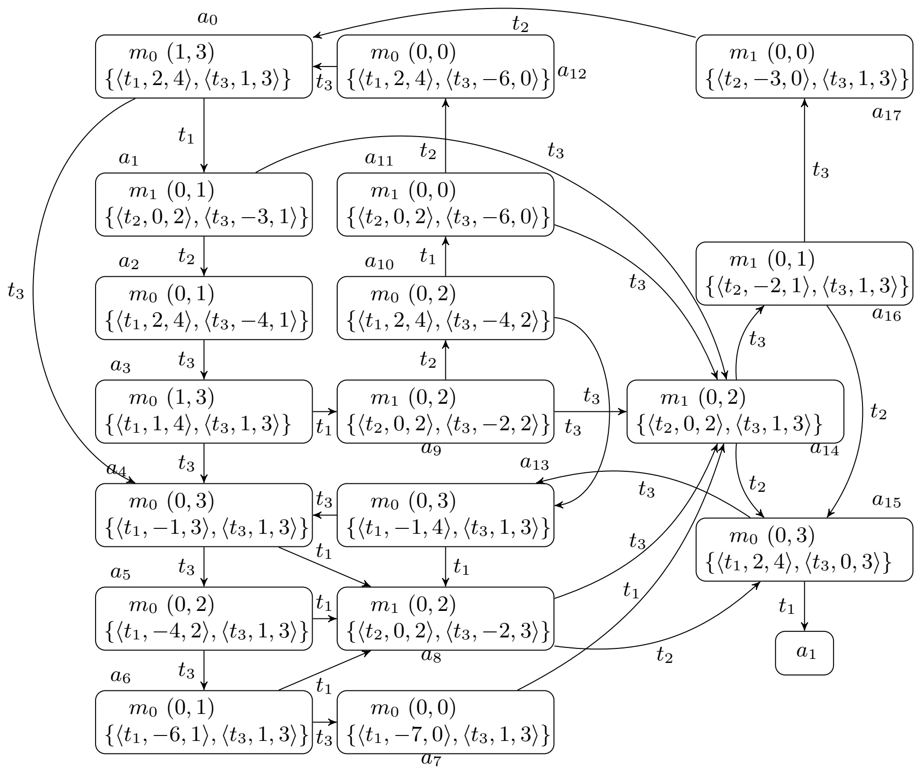

a_0
$m_0\ (1,3)$ $\{\langle t_1,2,4\rangle,\langle t_3,1,3\rangle\}$
a_{12}
$m_0\ (0,0)$ $\{\langle t_1,2,4\rangle,\langle t_3,-6,0\rangle\}$
a_{17}
$m_1\ (0,0)$ $\{\langle t_2,-3,0\rangle,\langle t_3,1,3\rangle\}$
a_1
$m_1\ (0,1)$ $\{\langle t_2,0,2\rangle,\langle t_3,-3,1\rangle\}$
a_{11}
$m_1\ (0,0)$ $\{\langle t_2,0,2\rangle,\langle t_3,-6,0\rangle\}$
a_{16}
$m_1\ (0,1)$ $\{\langle t_2,-2,1\rangle,\langle t_3,1,3\rangle\}$
a_2
$m_0\ (0,1)$ $\{\langle t_1,2,4\rangle,\langle t_3,-4,1\rangle\}$
a_{10}
$m_0\ (0,2)$ $\{\langle t_1,2,4\rangle,\langle t_3,-4,2\rangle\}$
a_3
$m_0\ (1,3)$ $\{\langle t_1,1,4\rangle,\langle t_3,1,3\rangle\}$
a_9
$m_1\ (0,2)$ $\{\langle t_2,0,2\rangle,\langle t_3,-2,2\rangle\}$
a_{14}
$m_1\ (0,2)$ $\{\langle t_2,0,2\rangle,\langle t_3,1,3\rangle\}$
a_4
$m_0\ (0,3)$ $\{\langle t_1,-1,3\rangle,\langle t_3,1,3\rangle\}$
a_{13}
$m_0\ (0,3)$ $\{\langle t_1,-1,4\rangle,\langle t_3,1,3\rangle\}$
a_{15}
$m_0\ (0,3)$ $\{\langle t_1,2,4\rangle,\langle t_3,0,3\rangle\}$
a_5
$m_0\ (0,2)$ $\{\langle t_1,-4,2\rangle,\langle t_3,1,3\rangle\}$
a_8
$m_1\ (0,2)$ $\{\langle t_2,0,2\rangle,\langle t_3,-2,3\rangle\}$
a_1
a_6
$m_0\ (0,1)$ $\{\langle t_1,-6,1\rangle,\langle t_3,1,3\rangle\}$
a_7
$m_0\ (0,0)$ $\{\langle t_1,-7,0\rangle,\langle t_3,1,3\rangle\}$
t_1
t_2
t_3

Fig. 2. The TAG of Fig. 1

- *Minimum and maximum access time*
  - $MinAT_\pi(a_i) = MinAT_\pi(a_{i-1}) + \max(0, \beta_{i-1_{t_i}} - (t_{i_{max}} - t_{i_{min}}))$
  - $MaxAT_\pi(a_i) = MaxAT(a_{i-1}) +$
    $\min(\min_{t\in\uparrow(a_{i-1},t_i)}(t_{\min}), \min_{t\in\downarrow(a_{i-1},t_i)}(\beta_{i-1_t} - a_{i-1}.H))$
- *Minimum and maximum firing time*
  - $MinFT_\pi(t_i) = MinAT_\pi(a_i)$
  - $MaxFT_\pi(t_i) = MaxAT_\pi(a_{i-1})$

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 satisfyied within a time interval $[d, D)$, with $d \in \mathbb{R}_{\geq 0}$ and $D \in (\mathbb{R}_{\geq 0} \cup \infty)$, 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.

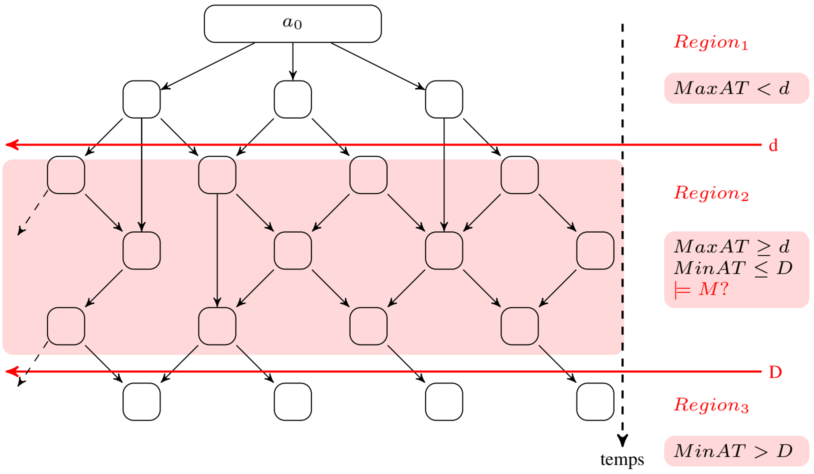


Fig. 3. Reachability analysis on the TAG

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.

Because of lack of space, we do not give the detailed algorithms for checking the above formulae, but give the main intuition.

In order to check any of the above properties, we propose on-th-fly approach where 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 analysed 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.

For instance checking the fromula 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 return *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 return *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*.

A similar approach can be trivially imagined for event-based approaches.

5 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 optimised for time efficiency yet, therefore no timing figures are given in this section). All results reported in this section have been obtained on 2.8 gigahertz Intel with four gigabytes of RAM. The implemented prototype allowed us to have first comparison with existing approaches with respect to the size of obtained graphs. We used the TINA tool to build the SCGs, ROMEO tool for the ZBGs and our tool for the TAGs. We tested our approach on several TPN models and we report here the obtained results. The considered models are representative of the characteristics that may have a TPN, such as: concurrency, synchronisation, disjoint firing intervals and infinite firing bounds. The two first models (Figure 4(a) and Figure 4(b)) are two parametric models where the number of processes can be increased. In Figure 4(a), the number of self loops ($p_n \dashrightarrow t_n \dashrightarrow p_n$ is increased while in Figure 4(b) the number of processes, whose behavior is either local, by transition t_i, or global by synchronization with all the other processes, by transition t_0, is increased.

In addition to these two illustrative examples, we used two well known other parametric TPN models. The first one [10] represents a composition of producer/consumer models. The second (adapted from [15]) is the Fischer's protocol for mutual exclusion.

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). The obtained results for the producers/consumers models show that the TAG yields better abstraction (linear order) than the SCG and the ZBG approaches. Each time a new module of producer/consumer is introduced, the size of graphs increases for all three approaches. However, the SAG achieves a better performance than the two other approaches. For the TPN of Figure 4(a), the obtained results show that the size of the TAG exponentially increases when the the parallelism occur in the structure of TPN. This is also the case also for the ZBG and the SCG methods, and we can see that our method behaves better

Table 1. Experimentation results

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	407 / 1 255
3	4 604 / 21891	3 240 / 15 200	1 618 / 6 892
4	14 086 / 83 375	9 504 / 56 038	3 972 / 20 500
5	31 657 / 217 423	20 877 / 145 037	8 175 / 48 351
6	61 162 / 471 254	39 306 / 311 304	15 157 / 99 539
7	107 236 / 907 708	67 224 / 594 795	26 113 / 186 363
8	175 075 / 1 604 319	107 156 / 1 044 066	42 503 / 324 600
9	270 632 / 2 655 794	161 874 / 1 718 104	66 103 / 534 055
10	400 648 / 4 175 413	234 398 / 2 687 147	99 036 / 839 011
Nb. self-loops	TPN example with concurrency (Figure 4(a))		
1	39 / 72	40 / 74	39 / 72
2	471 / 1 296	472 / 1 299	354 / 963
3	6 735 / 25 056	6 736 / 25 060	2 745 / 9 888
4	119 343 / 563 040	119 344 / 563 045	19 488 / 87 375
5	2 546 679 / 14 564 016	? / ?	130 911 / 701 748
Nb. processes	TPN example with synchronization (Figure 4(b))		
1	1 / 2	2 / 4	1 / 2
2	13 / 35	14 / 38	13 / 35
3	157 / 553	158 / 557	118 / 409
4	2 245 / 10 043	2 246 / 10 048	915 / 3 909
5	3 9781 / 21 7681	39 782 / 217 687	6 496 / 33 071
6	848 893 / 5 495 603	848 894 / 5 495 610	43 637 / 258 051
7	? / ?	? / ?	282 514 / 1.90282e+06
Nb. processes	Fischer protocol		
1	4 / 4	4 / 4	4 / 4
2	18 / 29	19 / 32	20 / 32
3	65 / 146	66 / 153	80 / 171
4	220 / 623	221 / 652	308 / 808
5	727 / 2 536	728 / 2 615	1 162 / 3 645
6	2 378 / 9 154	2 379 / 10 098	4 274 / 15 828
7	7 737 / 24 744	7 738 / 37 961	15 304 / 66 031
8	25 080 / 102 242	25 081 / 139 768	53 480 / 265 040

when we increment the self-loop structures in the model. The ZBG's and the SCG's execution have aborted due to a lack of memory when the number of self-loops was equal to 5. The number of edges of the obtained graphs follows the same proportion. In the synchronisation pattern exmaple, our approache behaves well as well. Indeed, with, 1, 2 and 3 processes, the sizes of the obtained graphs are almost similar with the three approaches. But, from 4 synchronised processes, the size of the SCGs and the ZBGs increase exponentially, leading to a state explosion with 7 processes, whereas the TAGs have been computed successfully with 7 processes (and even more). The Fischer protocol model is the only model where our approach leads to relatively bad results

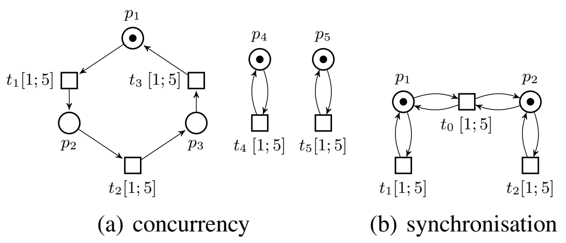


Fig. 4. TPN models used in the experiments

(although the difference with the two other approaches is linear). Our first explanation is that, in case of disjoint firing intervals, the abstraction can be weak in some cases. In fact, when a transition t is enabled by an aggregate t and there exists a trasnition t', not enabled by a, s.t. $t_{min} > t'_{max}$, a is considered non equivalent (while it could be) to all aggregates where the earliest firing time of t is not the same. However, it could be that t and t' are never enabled simultanously. We think that taking into account some structural properties of the model could allow to refine our abstraction.

The experimental results show (in most cases) an important gain in performances in terms of graph size (nodes/arcs) compared to the SCG and the ZBG approaches for the tested examples. This promises performant verification approaches based on the TAG .

6 Conclusion

We proposed adapted algorithms for reachability analysis of time properties based on a new finite abstraction of the TPN state space. Unlike, the existing approaches, our abstraction can be directly useful to check both state and event-based logic properties. 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 implement and experiment our verification algorithms. Second, we believe that the size of the TAG can be further reduced while preserving time properties without necessarily preserving all the paths of the underlying TPN. We also plan to design and implement model checking algorithms for full *TCTL* logic.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. 126(2), 183–235 (1994)
2. Berthomieu, B., Diaz, M.: Modeling and Verification of Time Dependent Systems Using Time Petri Nets. IEEE Trans. Software Eng. 17(3), 259–273 (1991)
3. Berthomieu, B., Menasche, M.: An Enumerative Approach for Analyzing Time Petri Nets. In: IFIP Congress, pp. 41–46 (1983)
4. Berthomieu, B., Vernadat, F.: State Class Constructions for Branching Analysis of Time Petri Nets. In: Garavel, H., Hatcliff, J. (eds.) TACAS 2003. LNCS, vol. 2619, pp. 442–457. Springer, Heidelberg (2003)

5. Berthomieu, B., Vernadat, F.: Time Petri Nets Analysis with TINA. In: QEST, pp. 123–124 (2006)
6. Boucheneb, H., Gardey, G., Roux, O.H.: TCTL Model Checking of Time Petri Nets. J. Log. Comput. 19(6), 1509–1540 (2009)
7. Boyer, M., Roux, O.H.: Comparison of the Expressiveness of Arc, Place and Transition Time Petri Nets. In: Kleijn, J., Yakovlev, A. (eds.) ICATPN 2007. LNCS, vol. 4546, pp. 63–82. Springer, Heidelberg (2007)
8. Gardey, G., Roux, O.H., Roux, O.F.: Using Zone Graph Method for Computing the State Space of a Time Petri Net. In: Larsen, K.G., Niebert, P. (eds.) FORMATS 2003. LNCS, vol. 2791, pp. 246–259. Springer, Heidelberg (2004)
9. Hadjidj, R., Boucheneb, H.: Improving state class constructions for CTL* model checking of time Petri nets. STTT 10(2), 167–184 (2008)
10. Hadjidj, R., Boucheneb, H.: On-the-fly TCTL model checking for time Petri nets. Theor. Comput. Sci. 410(42), 4241–4261 (2009)
11. Klai, K., Aber, N., Petrucci, L.: To appear in a new approach to abstract reachability state space of time petri nets. In: TIME, Lecture Notes in Computer Science. Springer (2013)
12. Larsen, K.G., Pettersson, P., Yi, W.: Model-checking for real-time systems. In: Reichel, H. (ed.) FCT 1995. LNCS, vol. 965, pp. 62–88. Springer, Heidelberg (1995)
13. Lime, D., Roux, O.H.: Model Checking of Time Petri Nets Using the State Class Timed Automaton. Discrete Event Dynamic Systems 16(2), 179–205 (2006)
14. Merlin, P.M., Farber, D.J.: Recoverability of modular systems. Operating Systems Review 9(3), 51–56 (1975)
15. Penczek, W., Pólrola, A., Zbrzezny, A.: SAT-Based (Parametric) Reachability for a Class of Distributed Time Petri Nets. T. Petri Nets and Other Models of Concurrency 4, 72–97 (2010)
16. Petri, C.A.: Concepts of net theory. In: MFCS 1973, pp. 137–146. Mathematical Institute of the Slovak Academy of Sciences (1973)
17. Pezzè, M., Young, M.: Time Petri Nets: A Primer Introduction. Tutorial Presented at the Multi-Workshop on Formal Methods in Performance Evaluation and Applications (1999)
18. Ramchandani, C.: Analysis of asynchronous concurrent systems by timed Petri nets. Technical report, Cambridge, MA, USA (1974)
19. Sifakis, J.: Use of Petri nets for performance evaluation. Acta Cybern. 4, 185–202 (1980)
20. Yoneda, T., Ryuba, H.: CTL model checking of time Petri nets using geometric regions (1998)