Final report of research training period

Arnaud Sangnier
asangnier@yahoo.fr
Master SAR - spécialité CRAR
Université Pierre et Marie Curie (Paris)

Monday 5th September 2005

Location of the training period : Dipartimento di Informatica - Universita degli studi di Torino (Italia)
Italian responsible : Jeremy Sproston (*Jeremy.Sproston @di.unito.it*)
French responsible : Claude Dutheillet (*Claude.Dutheillet@lip6.fr*)

Contents

Description of the training period

For this training period, the title of the subject is :

Toward verification of Time Coloured Petri Net

The description of the subject is :

The aim of this work is to explore frameworks for time coloured Petri nets. The approach is to consider a model which subsumes both coloured Petri nets and time Petri nets, in which intervals on transitions determine the time at which the respective transitions may fire.
The starting point will be a survey of existing works in the field, including proposals for time Petri nets and their numerous variants, coloured Petri nets and timed automata. The associated analysis and verification methods for each of the above frameworks will be considered.

A definition of coloured Petri nets will then be given in which the semantics will take the form of timed transition systems. It will be considered how time coloured Petri nets can be transformed syntactically into time Petri nets and/or (networks of) timed automata. The integration of the proposed syntactic transformations into an existing toolset will be considered and implemented.

It is envisioned that the above translations will work with the notion of coloured state. Based on the experiences of these translations, possible solutions for the use of symbolic states, as in the use of symbolic reachability graph method for coloured Petri nets, can be considered in the context of time Petri nets.

Acknowledgement

I would like to thank the people who have helped me to realize this work and in particular :

- Jeremy Sproston and Claude Dutheillet for all the help they gave to me ;
- Susanna Donatelli for her advice ;
- Massimiliano De Pierro for the help with GreatSPN ;
- Yann Thierry-Mieg for having given to a number of Petri Net models;
- and the laboratory of computer science of the University of Torino for having hosted me.

Abstract

We introduce Transitions Time Well-formed Nets (TTWN), a formalism derived from coloured Petri nets and Time Petri Nets, which can be used for modelling real-time systems. TTWN are Well-formed Nets to which time constraints have been added in the form of rational intervals associated with its transitions. These intervals give the earliest and the latest time during which a transition can be fired once it has been enabled by a marking. It is possible to build a timed automaton whose behavior is equivalent to the behavior of the TTWN, and which can be used subsequently in the automatic verification of the real-time system. We consider two such methods for the construction of such a timed automaton. The implementation of one of the methods for the restricted class of Time Petri Nets will be presented, and compared with the tool Romeo, which computes the state class timed automaton of a Time Petri Net. Finally, we discuss the difficulties raised by the adaptation of the theory of the symbolic reachability graph developed for Well-formed Nets to TTWNs.

Introduction

In order to verify the correct behavior of systems it is interesting to be able to modelize these systems and to test their properties, because if the model is close enough of the real system, the properties of the model are also valid for the system. In this direction, automata and Petri Nets have appeared to be formalisms that could represent properties of a system and on which verification methods are possible. However the properties verified on these models do not take account of real-time issues. Since, for many systems and in particular for real-time systems, time is an important factor, these formalisms have been extended to deal with time parameters. In fact, automata have been extended to timed automata, in which the time constraints are represented with the help of real-valued clocks and constraints over these clocks. As for the Petri Nets, different formalims have been proposed to add real-time to them. One of the methods consists in associating rational intervals to the transitions. Furthermore in order to model larger systems which exhibit symmetries, Petri Nets have been extended to coloured Petri Nets where the black tokens were replaced by elements of predefined sets. In this report, we will concern ourself with adding time constraints to a peculiar type of coloured Petri Net called Well-formed Nets, and then we will study methods to verify properties of systems defined with this new formalism.

Chapter 1

State of the art

1.1 Introduction

This chapter is a copy of the bibliographic report I wrote at the beginning of the training period. It gives a survey of different modelling techniques that permit the representation of real-time systems, in particular timed automata and Transitions Time Petri Net.

1.2 Timed automata

In this section, we will introduce the concept of timed automata. We will first present them in an intuitive way, then we will give the syntax and the semantics of timed automata. Different methods to verify their behavior will also be studied, and finally we will present some extensions of timed automata.

1.2.1 Preliminaries: Transition systems and timed transition systems

In order to understand well the origin of timed automata, it helps to have in mind the definition of transition system and of their extended version with reals representing elapsed time ([1], [7]).
A transition system is a state-transition graph, where the labels of the transitions represent events. A transition system S can be represented by a tuple $\langle Q, Q^O, \Sigma, \rightarrow\rangle$ where :

- Q is the set of the states;
- $Q^0 \subseteq Q$ are the initial states;
- Σ is the set of the events (or labels);
- $\rightarrow \subseteq Q \times \Sigma \times Q$ is the transition relation (i.e. the set of transitions).

When $(q, a, q') \in \rightarrow$, it is denoted $q \xrightarrow{a} q'$ and it means that when the state of the system is q, it can chenge to q' on event a. The Figure 1.1 gives an example of a transition system with $Q = \{q1, q2\}$, $Q^0 = \{q1\}$, $\Sigma = \{a, b\}$ and $\rightarrow = \{(q1, a, q2), (q2, b, q1)\}$. The transition system is drawn as a graph, where nodes represent states, and arcs represent transitions. The initial state, $q1$, is labelled with an arc with no source.

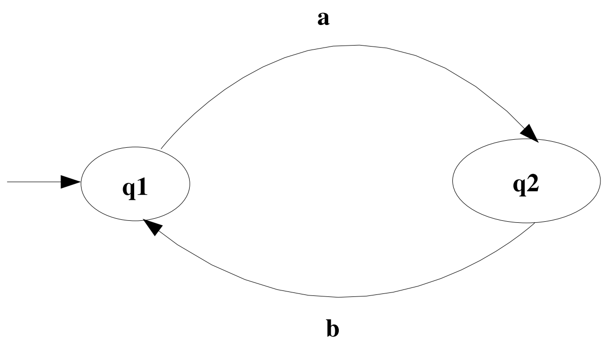


Figure 1.1: Example of a transition system

In order to be able to define parts of a system separately, it is interesting to define the *product* of transistion systems. We consider two transition systems $S_1 = \langle Q_1, Q_1^0, \Sigma_1, \rightarrow_1 \rangle$ and $S_2 = \langle Q_2, Q_2^0, \Sigma_2, \rightarrow_2 \rangle$. To make the product, it is necessary to define the set of events Σ that is accepted by the product. We also define a partial function $f : ((\Sigma_1 \cup \{\bullet\}) \times (\Sigma_2 \cup \{\bullet\})) \hookrightarrow \Sigma$. This function is used to represent the link between the events of S_1 end the events of S_2 and it is called *synchronisation function*.
Then the product of the transition systems, denoted $(S_1 \mid\mid S_2)_{f,\Sigma}$ is a transition system $S = \langle Q, Q^O, \Sigma, \rightarrow \rangle$ with :

- $Q = Q_1 \times Q_2$;
- $Q^0 = Q_1^0 \times Q_2^0$;
- $\rightarrow$ is defined by :

  $\forall (q,q') \in Q_1 \times Q_2$ with $q = (q_1, q_2)$ and $q' = (q_1', q_2')$ and $\forall c \in \Sigma$, $q \xrightarrow{c} q'$ (i.e. $(q,c,q') \in \rightarrow$) if and only if there exists $(a,b) \in (Q_1 \cup \{\bullet\}) \times (Q_2 \cup \{\bullet\})$ such that :

  (i) $f(a,b) = c$;

  and (ii) If $a = \bullet$ then $q_1 = q_1'$;

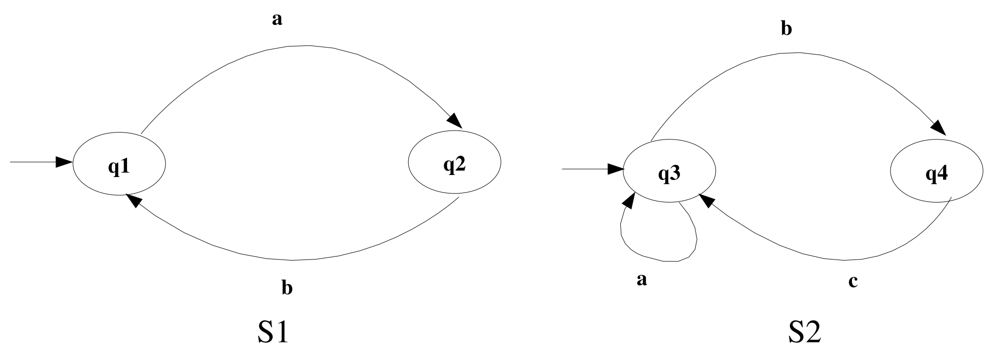
  and (iii) If $b = \bullet$ then $q_2 = q_2'$;

  and (iv) If $a \in \Sigma_1$ then $q_1 \xrightarrow{a} q_1'$;

  and (v) If $b \in \Sigma_2$ then $q_2 \xrightarrow{b} q_2'$.

In a more practical way, we can define a product of two transitions systems $S_1 = \langle Q_1, Q_1^0, \Sigma_1, \rightarrow_1 \rangle$ and $S_2 = \langle Q_2, Q_2^0, \Sigma_2, \rightarrow_2 \rangle$ taking $\Sigma = \Sigma_1 \cup \Sigma_2$, and the synchronisation function $f : ((\Sigma_1 \cup \{\bullet\}) \times (\Sigma_2 \cup \{\bullet\})) \hookrightarrow (\Sigma_1 \cup \Sigma_2)$ defined by :

- $\forall a \in \Sigma_1 \cap \Sigma_2$, $f(a,a) = a$;



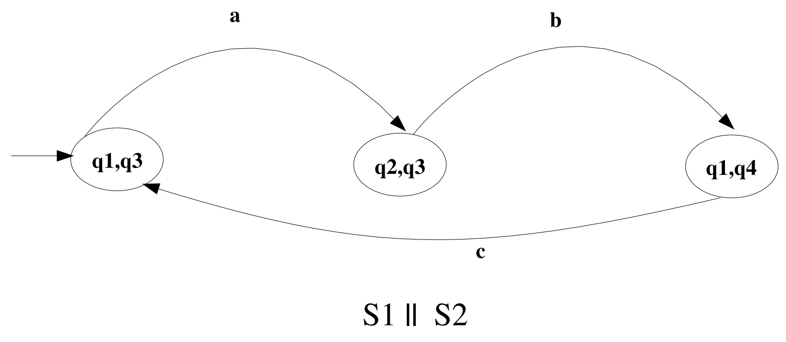


Figure 1.2: Example of a simple product of transition systems

- $\forall a \in \Sigma_1 \setminus \Sigma_2$, $f(a, \bullet) = a$;
- $\forall b \in \Sigma_2 \setminus \Sigma_1$, $f(\bullet, b) = b$.

We will denote this simple product $(S_1 \; || \; S_2)$. Note that this product makes sense in particular when $\Sigma_1 \cap \Sigma_2 \neq \emptyset$. The Figure 1.2 gives an example of such a product.
It has been shown that transition systems are useful to represent the behavior of a system, but they cannot give quantitative timing information for the behavior of real-time systems, since they only give information linked to a finite set of events. In order to represent systems with time parameters, transition systems have been extended to *timed transistion systems*. Timed transition systems are transition systems where the labels of transitions can belong to a finite set of events Σ or can be real numbers. It means that for a timed transition system $S = \langle Q, Q^O, \Sigma, \rightarrow \rangle$, the transition relation $\rightarrow \subseteq Q \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times Q$. The transitions that are labeled with an element of Σ are called *discrete transitions* and the transitions that are labeled with a real positive number are called *continuous transitions*. The product of timed transition systems can be defined in a similar way to the case of untimed transition systems.

1.2.2 Syntax and semantics of timed automata

Informal introduction to timed automata

A timed automaton can be seen like a transition system to which a finite set of clocks are added [1]. The states of the system are then called *locations* and the transitions are called *switches*. The switches are supposed to be instantaneous and time elapses in the locations. For each location it is possible to define an invariant that describes the values of the different clocks that are admitted in this location. For each switch it is possible to define a guard, which is a necessary condition on the values of the clocks for executing the switches, and it is also possible to define a subset of the clocks which describes the clocks that are reset by the execution of the switches.

For instance, we may want to model the behavior of a timer for cooking eggs. When somebody pushes the button of this timer, after 5 time units (5 minutes for instance) the bell of the timer rings for a total duration of 1 time unit. When the button has been pushed, it cannot be pushed again before the bell finished ringing. This system can be modelled by the timed automaton with one clock represented on the Figure 1.3. Initially the timer is in location l1, and it can stay in this location without any time constraint. If someone pushes the button, it goes to the location l2 and the clock x is reset, the system has to stay in the location l2 5 time units and not more; then it goes to location l3 where it has to stay only 1 time unit, and finally it returns to its initial state.

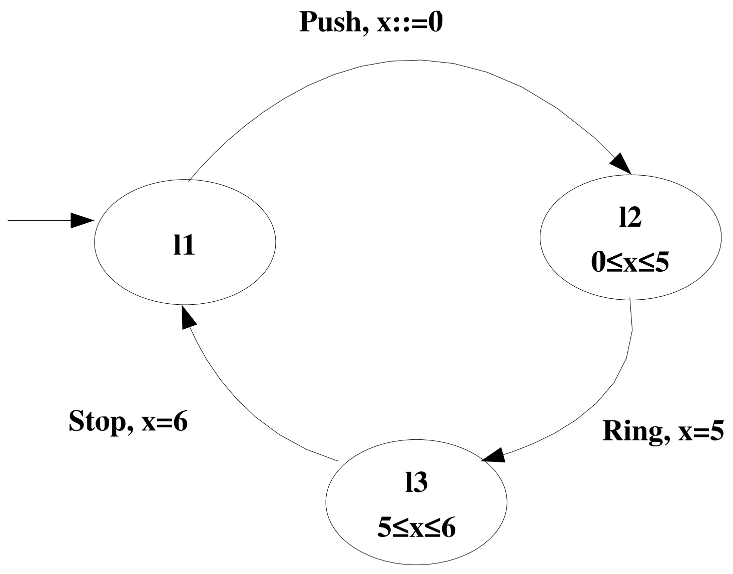


Figure 1.3: Example of a timed automaton

Clock constraints and clock interpretations

As we have seen, it is possible to define guards on switches and invariants on locations that describe possible values for the different clocks of the timed automaton. The guards and the invariants are represented by so-called *clock constraints* [1]. We consider X a set of clocks. The set $\Phi(X)$ of the clock constraints over X is defined by the following grammar :

$$\varphi := x \leq c | x \geq c | x < c | x > c | \varphi_1 \wedge \varphi_2$$

where $x \in X$, $c \in \mathbb{Q}_{\geq 0}$, $\varphi \in \Phi(X)$ and φ_1, $\varphi_2 \in \Phi(X)$.

In order to describe the values of the different clocks, *clock interpretations* are used. A clock interpretation v for a set of clocks X assigns to each clock a real value. Formally, a clock interpretation v is a total function $X \mapsto \mathbb{R}_{\geq 0}$. A clock interpretation v is said to satisfy a clock constraint φ over a set of clocks X if and only if φ is true for the clock values given by v. We also need two notations on the clock interpretations. Let consider v a clock interpretation for the set of clocks X, then :

- $\forall \delta \in \mathbb{R}_{\geq 0}$, $v + \delta$ is the clock interpretation u such that $\forall x \in X, u(x) = v(x) + \delta$ (representing the passage of δ time units);
- $\forall Y \subseteq X$, $v[Y := 0]$ defines the clock interpretation u such that $\forall x \in Y$, $u(x) = 0$ and $\forall x \in X \setminus Y$, $u(x) = v(x)$ (this is used to reset some clocks).

We will denote $V(X)$ the set of the clock interpretations and we define $\bar{0}$ the clock interpretation over X such that $\forall x \in X, \bar{0}(x) = 0$.

Syntax of timed automata

We will now give the syntax of the timed automata [1]. A timed automaton A is a tuple $\langle L, L^0, \Sigma, X, I, E \rangle$ where :

- L is a finite set of locations;
- $L^0 \subseteq L$ is a set of the initial locations;
- Σ is a finite set of labels;
- X is a finite set of clocks;
- I is a total function $L \mapsto \Phi(X)$ that associates to each location an invariant (i.e. a clock constraint);
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$ represents the set of the switches. If we consider a switch $t = (s, a, \varphi, \lambda, s') \in E$, it represents a switch from s to s' on the label (or event) a; φ is the clock constraint (or guard) associated to this switch and $\lambda \subseteq X$ gives the set of clocks that have to be reset when the switch is executed.

Semantics of timed automata

The semantics of such a timed automaton $A = \langle L, L^0, \Sigma, X, I, E \rangle$ can be described [1] with a timed transition system $S_A = \langle Q_A, Q_A^0, \Sigma, \rightarrow \rangle$. First, we define Q_A by $Q_A = \{(s,v) \in L \times V(X) | v \models I(s)\}$. A state of S_A is a pair (s,v) where s is a location of A and v is a clock interpretation such that v satisfies the invariant $I(s)$ of s. The set of initial states of S_A is the set $Q_A^0 = \{(s, \bar{0}) | s \in L^0 and \bar{0} \models I(s)\}$, which means that (s,v) is a initial state of S_A if and only if s is an initial location of A, $\forall x \in X, v(x) = 0$ and the clock interpretation v satisfies $I(s)$. The transition relation $\rightarrow \subseteq Q_A \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times Q_A$ is defined by:

- for the continuous transition :
  $\forall \delta \in \mathbb{R}_{\geq 0}, (s,v) \overset{\delta}{\rightarrow} (s, v+\delta)$ if $\forall \delta' \in [0,\delta], v+\delta'$ satisfies $I(s)$;

- for the discrete transition :
  for all states (s,v) of S_A and for all switches (s,a,φ,λ,s') such that v satisfies φ, there is the transition $(s,v) \overset{a}{\rightarrow} (s', v[\lambda := 0])$.

For example, some sample transitions of the timed transition system associated to the timed automaton of the Figure 1.3 are:

$$(l1,0) \overset{1.5}{\rightarrow} (l1,1.5) \overset{Push}{\rightarrow} (l2,0) \overset{3.2}{\rightarrow} (l2,3.2) \overset{1.8}{\rightarrow} (l2,5) \overset{Ring}{\rightarrow} (l3,5) \overset{1}{\rightarrow} (l3,6) \overset{Stop}{\rightarrow} (l1,6)$$

where the value of the clock x is written as the second component of each state, instead of the associated clock interpretation. For three states q, q', q'' of S_A such that $q \overset{\delta}{\rightarrow} q'$ and $q' \overset{\varepsilon}{\rightarrow} q''$ with $\delta, \varepsilon \in \mathbb{R}$, we also have $q \overset{\varepsilon+\delta}{\rightarrow} q''$ (time-additivity property).
We also remark that since the labels on continuous transitions are real numbers, as are the clock values, the timed transition system S_A associated with a timed automaton A has infinitely many states and infinitely many symbols.

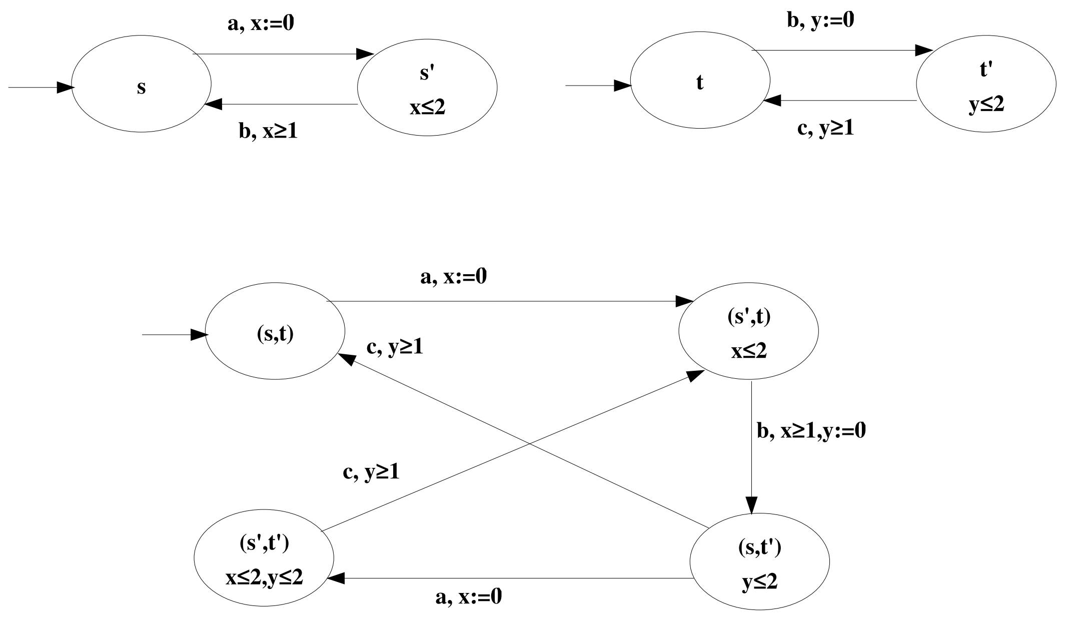


Figure 1.4: Example of a simple product of two timed automata

Product of timed automata

Such as for the transition systems, it appears to be useful to define a product on timed automata [1], [7], that will allow to describe a global system with different timed automata. Our presentation is based on [7]. To build this product, we consider two timed automata $A_1 = \langle L_1, L_1^0, \Sigma_1, X_1, I_1, E_1 \rangle$ and $A_2 = \langle L_2, L_2^0, \Sigma_2, X_2, I_2, E_2 \rangle$, a set of events Σ

and a synchronisation function $f : ((\Sigma_1 \cup \{\bullet\}) \times (\Sigma_2 \cup \{\bullet\})) \hookrightarrow \Sigma$. We suppose that the two sets of clocks X_1 and X_2 are disjoint. Then the product of the two timed automata A_1 and A_2 taking account of Σ and f and denoted $(A_1 \mid\mid A_2)_{f,\Sigma}$ is a timed automaton $A = \langle L_1 \times L_2, L_1^0 \times L_2^0, \Sigma, X_1 \cup X_2, I, E\rangle$ where I is defined by :

$\forall(s_1,s_2) \in L_1 \times L_2, I(s_1,s_2) = I_1(s_1) \wedge I_2(s_2)$

and E is defined by :

(i) $\forall c \in \Sigma$ such that there exists $(a,b) \in \Sigma_1 \times \Sigma_2$ that verifies $f(a,b) = c$ then :
$\forall(s_1,a,\varphi_1,\lambda_1,s'_1) \in E_1$ and $\forall(s_2,b,\varphi_2,\lambda_2,s'_2) \in E_2$,
$((s_1,s_2),c,\varphi_1 \wedge \varphi_2,\lambda_1 \cup \lambda_2,(s'_1,s'_2)) \in E$;

(ii) $\forall c \in \Sigma$ such that there exists $a \in \Sigma_1$ that verifies $f(a,\bullet) = c$ then :
$\forall(s_1,a,\varphi_1,\lambda_1,s'_1) \in E_1$ and $\forall s_2 \in L_2$,
$((s_1,s_2),c,\varphi_1,\lambda_1,(s'_1,s_2)) \in E$;

(iii) $\forall c \in \Sigma$ such that there exists $b \in \Sigma_2$ that verifies $f(\bullet,b) = c$ then :
$\forall(s_2,b,\varphi_2,\lambda_2,s'_2) \in E_2$ and $\forall s_1 \in L_1$,
$((s_1,s_2),c,\varphi_2,\lambda_2,(s_1,s'_2)) \in E$.

As for the transition systems, we can make a simple product taking $\Sigma = \Sigma_1 \cup \Sigma_2$ and the same synchronisation function f as the one given for the simple product of two transition systems. We will denote this simple product $(A_1 \mid\mid A_2)$. Figure 1.4 gives an example of such a product. We remark that for a set of labels Σ and a synchronisation function $S_{(A_1||A_2)_{f,\Sigma}} = (S_{A_1} \mid\mid S_{A_2})_{f,\Sigma}$.

1.2.3 Reachability analysis of timed automata

Presentation of the problem

In this section, we will study methods used to solve the so-called *reachability* problem. In fact, to ensure safety properties on the system, it is interesting to know if the system can be in some states that do not verify a property. A location s' of a timed automaton is said to be reachable from a state q if there exists a run in the associated timed transitions system S_A which begins in q and which leads to $q' = (s', v')$ for some clock interpretations v'. A run r of a timed transition system S_A associated to a timed automaton $A = \langle L, L^0, \Sigma, X, I, E\rangle$ is an infinite sequence of states and of transitions, such that :

$$r = q_0 \xrightarrow{l0} q_1 \xrightarrow{l1} q_2 ... q_i \xrightarrow{li} q_{i+1} ...$$

with $\forall i \in \mathbb{N}, l_i \in \Sigma \cup \mathbb{R}_{\geq 0}$ and $q_i \in Q_A$.

We will denote $R_A(q)$ the set of the runs of S_A (also called runs of A by extension) starting at $q \in Q_A$. The reachability problem consists in finding the states q' of the system reachable from a state q. We denote $Reach_A(q)$ the set of the states of A reachable from q and so we have :

$$Reach_A(q) = \{q' \in Q_A \mid \exists r = q_0 \xrightarrow{l0} q_1 \xrightarrow{l1} q_2 ... \in R_A(q), \exists i \in \mathbb{N}, q_i = q'\}$$

The set of reachable states of A is deoted a $Reach_A = \bigcup_{q \in Q_A^0} Reach_A(q)$. A location l of A is reachable if there exists a state in $Reach_A$ with location l.
As we have seen before, the timed transition system S_A associated to a timed automaton A has infinitely many states and infinitely many symbols and therefore it is not possible to compute the reachable states directly on this timed transition sytem.

Time abstract transition systems and stable quotients

To solve the problem of the inifinite numbers of states and symbols of a timed transition system S_A associated with a timed automaton A, the concept of *time-abstract transition system* has been introduced [1]. The time abstract transition system associated with a timed automaton A and denoted U_A is a transition system $\langle Q, Q^0, \Sigma, \Rightarrow \rangle$ where :

- $Q = Q_A$;
- $Q^0 = Q_A^0$
- Σ is the set of labels of A;
- the transition relation $\Rightarrow$ is defined by :

  $q \overset{a}{\Rightarrow} q''$ with $(q, a, q'') \in Q \times \Sigma \times Q$ if, and only if,$\exists q' \in Q$ and $\delta \in \mathbb{R}_{\geq 0}$ such that $q \xrightarrow{\delta} q' \xrightarrow{a} q''$ exists in S_A.

Then U_A has a finite number of symbols, but the number of states still be infinite because it the same as S_A.
To solve this problem, a solution is to regroup the states into different equivalence classes and to do that an equivalence relation on Q_A can be defined. If this relation is *stable*, then it can be used to solve the reachability problem. An equivalence relation $\sim$ on Q_A is stable if and only if, if $q \sim u$ and $q \overset{a}{\Rightarrow} q'$ with $q, q', u \in Q_A$ and $a \in \Sigma$, then $\exists u' \in Q_A$ such that $u \overset{a}{\Rightarrow} u'$ and $u' \sim q'$. A stable equivalence can also be called a bisimulation of the time abstract transition system. For a stable equivalence relation $\sim$, we can build a transition system, denoted $[U_A]_\sim$ and called stable quotient of U_A according to the stable partition $\sim$, defined by :

- the states are the equivalent classes defined by $\sim$;
- the initial states are the equivalence classes that contain an initial state of U_A;
- the set of labels is Σ (the set of labels of A);
- the transition relation $\rightarrow$ is defined by :

  $\pi \xrightarrow{a} \pi'$ belongs to $[U_A]_\sim$ if $\exists q \in \pi$ and $q' \in \pi'$ such that $q \overset{a}{\Rightarrow} q'$ belongs to U_A.

To solve the reachability problem for a set of locations $L^F \subseteq L$, which means determining if the locations in L^F are reachable from the initial states, a sufficient condition is that the

considered equivalence relation $\sim$ is stable but also that it is L^F-sensitive. An equivalence relation $\sim$ is said to be L^F-sensitive if when $(s,v) \sim (s',v')$, $s,s' \in L^F$ or $s,s' \notin L^F$. The L^F-sensibility ensures that target locations and non-target locations will not be regrouped in a same equivalence class.
This leads to a proposition on the stable quotient graph according to a stable L^F-sensitive partition.

Proposition ([1]) Let A be a timed-automaton, $\sim$ be an equivalence relation over Q_A, and L^F be a set of locations of A such that $\sim$ is stable and L^F-sensitive. Then a location in L^F is reachable if and only if there exists an equivalence class π of $\sim$ such that π is reachable in the quotient $[U_A]_\sim$ and π contains a state whose location is in L^F.

This proposition has the consequence that to solve the reachability problem, if there is an equivalence relation $\sim$ which is stable and L^F-sensitive (and which has a finite number of equivalence classes), the computation of the reachability can be done on the corresponding quotient graph.

Remark In the next sections, we assume that the constants that appear in the clock constraints of the timed automata are defined with respect to non-negative integer values. This can be done, because with a multiplication on all the elements of a finite set in $\mathbb{Q}$, it is possible to obtain a set in $\mathbb{N}$.

Region equivalence and region automaton

In this part, we will define an equivalence relation, which is called region equivalence [1]. This equivalence relation groups the states of Q_A that have the same location taking account of the integer part and of the fractional part of the clocks' values. The integer part is used to know if a clock constraint is encountered for the clock considered (the constants in the clock constraint are in fact integers) and the fractional part is used to know which clocks will change first its integer part.
So, $\forall \delta \in \mathbb{R}_{\geq 0}$, we define $\lfloor \delta \rfloor$ the integer part of δ and $fr(\delta)$ the fractional part of δ such that $\delta = [\delta] + fr(\delta)$. We also define, $\forall x \in X$ (the set of clocks of the timed automaton), c_x the largest integer that appears in the clock constraints featuring the clock x.

The equivalence relation on the clock interpretations, denoted $\approxeq$ and called *region equivalence*, is defined by :

$\forall v, v' \in V(X), v \approxeq v'$ if and only if :

(i) $\forall x \in X, [v(x)] = [v'(x)]$ or $(v(x) > c_x$ and $v'(x) > c_x)$;

(ii) $\forall x, y \in X$ such that $v(x) \leq c_x$ and $v(y) \leq c_y$, $fr(v(x)) \leq fr(v(y)) \Leftrightarrow fr(v'(x)) \leq fr(v'(y))$;

(iii) $\forall x \in X$ such that $v(x) \leq c_x$, $fr(v(x)) = 0 \Leftrightarrow fr(v'(x)) = 0$.

The equivalence classes of this equivalence relation are called clock regions.

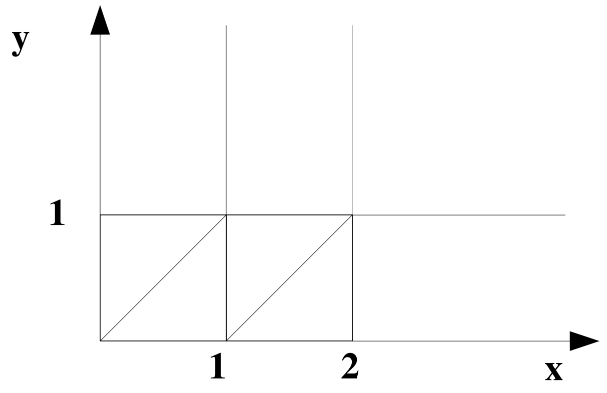


Figure 1.5: Clock regions for two clocks x and y

Figure 1.5 gives an example of clocks region with two clocks x and y with $c_x = 2$ and $c_y = 1$. In this figure, each open-line segment, corner point or open region is a clock region. For instance $[(0,1)]$,$[0 < x = y < 1]$ and $[0 < x < y < 1]$ are clock regions.

Region equivalence $\approxeq$ can be extended to the state space Q_A by the following definitions:

$$\forall (s,v),(s',v') \in Q_A, (s,v) \approxeq (s',v') \Leftrightarrow s = s' \wedge v \approxeq v'.$$

Region equivalence so defined over Q_A is stable and furthermore is L^F-sensitive for any $L^F \subseteq L$. Consequently, it is possible to work in the quotient graph $[U_A]_{\approxeq}$,called the *region automaton* $R(A)$, to solve the reachability problem. Since solving the reachability problem for a timed automaton A and for a set of locations L^F consists in computing the (finite) region graph, it has been proven that this problem is decidable. It has also been shown that this problem is PSPACE-complete.

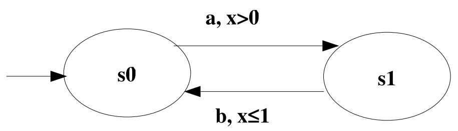


Figure 1.6: Timed automaton A_0 with one clock

The Figures 1.6 and 1.7 give an example of a timed automaton and its associated region automaton.

Region-graph based algorithm

In the reachability problem, we want to find if a target location s (or a set of target locations) will be reached from the initial configuration of the system. There are two types

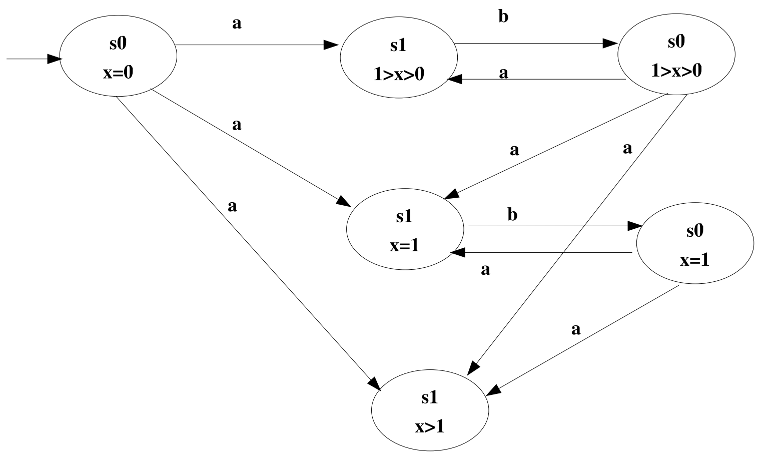


Figure 1.7: Region automaton $R(A_0)$ associated with A_0

of algorithm to solve this problem, the *forward traversal* algorithm and the *backward traversal* algorithm. Our presentation is adapted from [25].
First we will consider, the forward traversal algorithm. We denote $Q_{\approxeq}$ the set of the states of the quotient graph $[U_A]_{\approxeq}$, $Q^0_{\approxeq}$ the set of its initial states and $\rightarrow_{\approxeq}$ its transition relation. We define $\forall i \in \mathbb{N}$:

$$F_0 = Q^0_{\approxeq}$$

$$F_{i+1} = F_i \cup Suc(F_i)$$

where $Suc(F_i) = \{\pi' \in Q_{\approxeq} | \exists \pi \in F_i, \exists a \in \Sigma, \pi \overset{a}{\rightarrow}_{\approxeq} \pi'\}$.
Then a target location s is reachable if, and only if, there exists a state π of $Q_{\approxeq}$ such that the location of π is s and $\pi \in \bigcup_{i \geq 0} F_i$.

For the backward traversal algortihm, we define $\forall i \in \mathbb{N}$:

$$B_0 = \{\pi \in Q_{\approxeq} | loc(\pi) = s\}$$

$$B_{i+1} = B_i \cup Pre(B_i)$$

where the function $loc : Q_{\approxeq} \mapsto L$ gives for each equivalence class its associated location and where $Pre(B_i) = \{\pi \in Q_{\approxeq} | \exists \pi' \in B_i, \exists a \in \Sigma, \pi \overset{a}{\rightarrow}_{\approxeq} \pi'\}$.
The target location s is reachable if and only if $(\bigcup_{i \geq 0} B_i) \cap Q^0_{\approxeq} \neq \emptyset$.

Analysis using representatives

In order to manipulate easily the region automaton $[U_A]_{\approxeq}$, a possibility is to use representatives of the different equivalent classes [25]. The set of representatives is a finite set $D \subset Q_A$ such that $\forall \Pi \in Q_{\approxeq}, \pi \cap D \neq \emptyset$. The set of the representatives of a state π of $[U_A]_{\approxeq}$ is denoted $rep(\pi)$ and is defined by $rep(\pi) = \pi \cap D$. It is then possible to build a transition system $\langle D, D^0, \Sigma, \rightarrow \rangle$ associated to D with :

- $D^0 \subseteq D$ is the set of representatives of the initial states of $[U_A]_{\approxeq}$;
- the transition relation $\rightarrow$ is defined by :

1. $\forall d, d' \in D, \forall a \in \Sigma$, if $d \xrightarrow{a} d'$ then $\pi \xrightarrow{a} \pi'$ where π is the equivalence class of d and π' is the equivalence class of d';
2. $\forall \pi, \pi' \in Q_{\approxeq}, \forall a \in \Sigma$, if $\pi \xrightarrow{a} \pi'$ then $\forall d \in rep(\pi)$, $\exists d' \in rep(\pi')$ such that $d \xrightarrow{a} d'$.

The Figure 1.8 gives an example of a discretization of the clock region space in order to obtain representatives of regions (where the black squares denote representatives).

Just as for the region automaton, to solve the reachability problem it is possible to use forward or backward traversal algorithms on the transition sytem defined above.

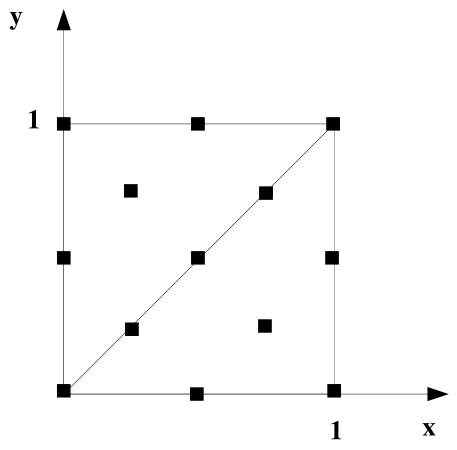


Figure 1.8: Example of representatives of clocks regions

Analysis using time abstracting equivalence

In this section, we will see that it is sometimes possible to use an equivalence relation that gives the coarsest partition of the states space Q_A [25].

A *time-abstracting bisimulation* over Q_A is a binary symmetric relation $B \subseteq Q_A \times Q_A$ such that $\forall (q_1, q_2) \in Q_A \times Q_A, (q_1, q_2) \in B$ if :

1. $\forall a \in \sigma$, if $\exists q_1' \in Q_A$ such that $q_1 \xrightarrow{a} q_1'$, then $\exists q_2' \in Q_A$ such that $q_2 \xrightarrow{a} q_2'$ and $(q_1', q_2') \in B$;

2. $\forall \delta \in \mathbb{R}_{\geq 0}$, if $\exists q_1' \in Q_A$ such that $q_1 \xrightarrow{\delta} q_1'$, then $\exists \delta' \in \mathbb{R}_{\geq 0}$ and $\exists q_2' \in Q_A$ such that $q_2 \xrightarrow{\delta'} q_2'$ and $(q_1', q_2') \in B$.

Note that the region equivalence is a time-abstracting bisimulation. The problem consists in finding the largest time-abstracting bisimulation, which is called the *time abstracting equivalence*; this largest bisimulation is the one that gives the coarsest partition. The time abstracting equivalence is an equivalence relation over Q_A and we remark by definition that it is stable.

To find this coarsest partition, there exist algorithms based for instance on the computation of reachable states and on the testing of stability (see [25]).

Zone automata

Another method consists in reducing the number of states of the region automaton by performing convex union of clocks region. A *clock zone* (see [1]) represents in fact such an union. The set of clock zones can be defined by the following grammar $\Psi(X)$:

$$\psi := x < c | x \leq c | x \geq c | x > c | x - y < c | x - y \leq c | \psi_1 \wedge \psi_2$$

where x and y represent clocks of X, c is an integer constant and ψ_1 and ψ_2 belongs to $\Psi(X)$. We will also denote ψ a clock zone defined by ψ.

The set of the clock zones generated by the grammar $\Psi(X)$ is then a convex set of the k-dimensional euclidian space (where k represents the number of clocks in X) which verifies the properties :

1. Each clock region is a clock zone ;
2. The intersection of two clock zones is a clock zone;
3. The clock constraints of a timed automaton are clock zones;
4. Let ψ_1 and ψ_2 be two clock zones: if the union $\psi_1 \cup \psi_2$ is convex, then it is a clock zone.

For the analysis of reachability, the three following operations on clock zones are usefull :

1. Let ψ_1 and ψ_2 be two clock zones; then the intersection of these two zones is the clock zone defined by $\psi_1 \wedge \psi_2$;
2. For a clock zone ψ, we denote $\psi \Uparrow$ the set of clock interpetations $v + \delta$ with $v \in \psi$ and $\delta \in \mathbb{R}_{\geq 0}$;
   then $\psi \Uparrow$ is a clock zone;
3. Let λ be a subset of clocks (i.e. $\lambda \subseteq X$) and ψ a clock zone, then $\psi[\lambda := 0]$ is the set of clock interpretations such that $\forall v \in \psi$, $v[\lambda := 0]$;
   then $\psi[\lambda := 0]$ is a clock zone.

Then a *zone* is a pair (s, ψ) where s is a location of the timed automaton $A = \langle L, L^0, \Sigma, X, I, E \rangle$ and ψ is a clock zone defined over the set of clocks X of A. For a switch $e = (s, a, \varphi, \lambda, s')$ and a zone (s, ψ), $succ(\psi, e)$ is the set of clock interpretations v' such that there exists $v \in \psi$ and the state (s', v') (with s' a location of A) can be reached from (s, v) by letting time elapse and by executing e. Then $succ(\psi, e)$ can be computed by the following formula :

$$succ(\psi, e) = (((\psi \wedge I(s)) \Uparrow) \wedge I(s) \wedge \varphi)[\lambda := 0]$$

Proposition ([1]) For a zone clock ψ and a switch e of a timed automaton A, $succ(\psi, e)$ is a clock zone.

For a timed automaton $A = \langle L, L^0, \Sigma, X, I, E \rangle$, it is possible to build a so-called *zone automaton* $Z(A)$, which is a transition system $\langle Q, Q^0, \Sigma, \rightarrow \rangle$ defined by :

- $Q \subseteq L \times \Psi(X)$;
- $Q^0 = \{(s, \bar{0}) | s \in L^0\}$;
- The relation transition $\rightarrow$ is defined by :

  for each switch $e = (s, a, \varphi, \lambda, s')$ of A and each clock zone ψ, there is a transition $(s, \psi) \xrightarrow{a} (s', succ(\psi, e))$.

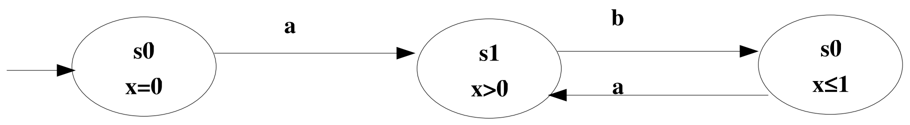


Figure 1.9: The reachable zone automaton $Z(A_0)$ of A_0

The Figure 1.9 gives the reachable zone automaton of the automaton A_0 represented by the Figure 1.6. We can notice that this zone automaton is smaller than the corresponding region automaton $R(A_0)$ of Figure 1.7 since it has less vertices and as well less transitions, but it still represents the behavior of A_0. However, in some cases the zone automaton is no smaller than the region graph.

Difference Bound Matrices : a data structure for timed automata

To represent a clock zone, it is possible to use a data structure called the *Difference Bound Matrix* (see [1]) and which is in fact a matrix over the set $\mathbb{D} = (\mathbb{Z} \times \{<, \leq\}) \cup \{\infty\}$. Each clock zone ψ of $\Psi(X)$ where X is a set of clocks can be represented by a $(k+1) \times (k+1)$ matrix D whose elements are in $\mathbb{D}$ (where k is a number of clocks in X). We denote $D(X)$ the set of difference bound matrices on X. For all $d \in \mathbb{Z} \times \{<, \leq\}$ we will denote by $(d)_1$ the projection of d on $\mathbb{Z}$ and by $(d)_2$ the projection of d on $\{<, \leq\}$. We index the set of clocks X such that $X = \{x_1, x_2, ..., x_k\}$ and we define $x_0 = 0$. A Difference Bound Matrix $D \in D(X)$ is defined by :

$\forall v \in V(X)$, v belongs to the clock zone represented by D if and only if $\forall i, j \in [|0, k|]$

(i) If $D_{ij} \in \mathbb{Z} \times \{<, \leq\}$ and $(D_{ij})_2 =<$ then $x_i - x_j < (D_{i,j})_1$;

(ii) If $D_{ij} \in \mathbb{Z} \times \{<, \leq\}$ and $(D_{ij})_2 =\leq$ then $x_i - x_j \leq (D_{i,j})_1$;

(iii) If $D_{ij} \in \{\infty\}$ then $x_i - x_j < \infty$.

For instance, the clock zone $\psi = (0 \leq x_1 < 2) \wedge (0 < x_2 < 1) \wedge (x_1 - x_2 \geq 0)$ on the set of clocks $X = \{x_1, x_2\}$ can be represented by the difference bound matrix D :

$$D = \begin{pmatrix} \infty & (0,\leq) & (0,<)) \\ (2,<) & \infty & \infty \\ (1,<) & (0,\leq) & \infty \end{pmatrix}$$

We denote that the difference bound matrice D' :

$$D' = \begin{pmatrix} (0,\leq) & (0,\leq) & (0,<)) \\ (2,<) & (0,\leq) & (2,<) \\ (1,<) & (0,\leq) & (0,\leq) \end{pmatrix}$$

represents the same clock zone as D. In order to do the analysis of reachability directly on the difference bound matrices, it is necessary to define a canonical form.The two following operations on $\mathbb{D}$ are needed :

1. Addition over $\mathbb{D}$ defined by :
   - $\forall d \in \mathbb{D}, d + \infty = \infty$;
   - $\forall d, e \in \mathbb{Z} \times \{<, \leq\}$, $d + e = ((d)_1 + (e)_1, (d)_2 \wedge (e)_2)$ (with $(< \wedge <) =<$, $(\leq \wedge \leq) = \leq$ and $(< \wedge \leq) = (\leq \wedge <) =<$).
2. Comparaison over $\mathbb{D}$ defined by :
   - $\forall d \in \mathbb{D}, d \leq \infty$;
   - $\forall d, e \in \mathbb{Z} \times \{<, \leq\}$, $d < e \Leftrightarrow (d)_1 < (e)_1$ or $((d)_1 = (e)_1$ and $(d)_2 < (e)_2)$ (with "$<$" $<$ "$\leq$").

It is then possible to present definitions and properties over the difference bound matrices :

1. A difference bound matrix D is *satisfiable*, if it represents a nonempty clock zone;
2. D is *unsatisfiable* $\Leftrightarrow \exists i_1, ..., i_j \in [|0,k|]$ such that $(D_{i_1 i_2} + D_{i_2 i_3} + ... + D_{i_j i_1}) < (0, \leq)$;
3. D is *canonical* $\Leftrightarrow \forall i, j, l \in [|0,k|], D_{il} \leq D_{ij} + D_{jl}$;
4. If a difference bound matrix is satisfiable, it has an equivalent canonical difference bound matrix;
5. Two canonical difference bound matrices D and D' are equivalent if and only if $\forall i, j \in [|0,k|], D_{ij} = D'_{ij}$.

To realize the analysis of reachability with difference bound matrices it is then possible to extend the operations on the clock zones (taking account that the clock constraints can be represented as clock zones and consequently as difference bound matrices) to difference bound matrices as follow :

- **Intersection** Let D and D' be two canonical difference bound matrices. The intersection D'' of D and D' is the difference bound matrix defined by $\forall i, j \in [|0,k|], D''_{ij} = min(D_{ij}, D'_{ij})$; then it is necessary to verify if D'' is satisfiable, and in the positive case to give its canonical form;

- **Elapsing of the time** $D\Uparrow$ is the matrix defined by $\forall (i,j) \in [|0,k|] \times [|1,k|], D\Uparrow_{ij} = D_{ij}$ and $\forall i \in [|0,k|], D\Uparrow_{i0} = \infty$ and then it is necessary to put $D\Uparrow$ in its canonical form;

- **Clock reset** $D[\lambda := 0]$ is the matrix D' defined by :

  (i) For $x_i \in \lambda$, $D'_{i0} = D'_{0i} = (0, \leq)$;

  (ii) For $x_i, x_j \in \lambda$, $D'_{ij} = (0, \leq)$;

  (iii) For $x_i \in \lambda$, $x_j \notin \lambda$, $D'_{ij} = D_{0j}$ and $D'_{ji} = D_{j0}$;

  (iii) For the other cases, $D'_{ij} = D_{ij}$.

  As for the other operations, the resulting matrix has to be made canonical.

For each of these operations, the difference bound matrix has to be transformed into its canonical form, for which there exist algorithms. With these operations, it is possible to extend the function *Succ* presented for the computation of the zone automata to the case of the difference bound matrices.

Remark In this part we present the difference bound matrices as a data structure to represent the zones, but in fact in the literature, it appears that other data structures can be used, for instance extensions of the BDDs (Binary Decision Diagrams).

Extensions of timed automata

In the literature, for instance in [25], it appears that the definition of timed automata and in particular of the form of the clock constraints is not always the same as the one we gave in this report and which corresponds to timed automata with diagonal-free constraints. A diagonal constraint is a constraint of the form $x - y < c$. Using such constraints, as shown in [5] does not increase the expressiveness of the model and furthermore complicates the building of the clock regions. In our study we limit ourselves to the simplest model of timed automata, which is why we also did not introduce updates of clocks on switches of the form $x := y$ which were also presented in [25] and in [5]. We will see in the part dedicated to the transformation of a time Petri net into a timed automaton, that such updates are not necessary in the context in which we want to work. Finally, in [6] it is proven that the diagonal-free timed automata guarantee decidability of the reachability problem.

1.3 Time Petri nets

In this part, we will introduce the concept of time Petri nets. We will describe them in an intuitive way, then we will give their syntax and semantic; we will also study some methods to build their state space and we will finally see a method that allows to transform a time Petri net into a timed automaton.

1.3.1 Presentation

In recent years, the Petri nets have appeared as a practical formalism to describe distributed systems, and consequently extensions have been proposed to include time parameters into this formalism in order to model real-time systems. There are two main classes of Petri nets that take account of time parameters :

1. Timed Petri nets;
2. Time Petri nets.

For each of this classes, it is necessary to decide where to introduce the time parameters in the formalism: should they be linked to the places, to the transitions or to the tokens? We will base our study on the class of the time Petri nets where the time parametes appear as intervals linked to the transitions and we will denote this kind of time Petri nets **T-TPN**. These intervals give the earliest time and the latest time of the firing of a transition after it has been enabled for a marking. In a T-TPN the firing of a transition is considered as instantaneous and time elapses in the places. For instance, we consider a system with two processes P1 and P2; P2 sends message to P1 and P1 sends an acknowledgement to P2 each time it has received a message. Each type of message needs at least one time unit to go from one process to the other, and P1 needs at most three time units to receive a message and P2 needs at most two time units to receive an acknowledgement. When P1 has received a message, it sends directly an acknowledgement.

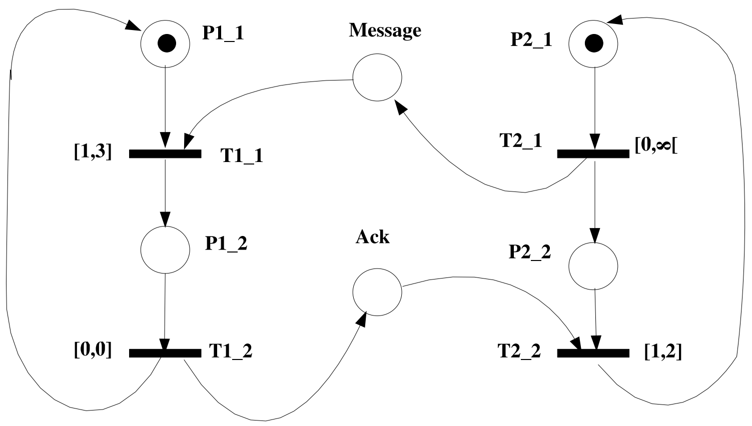


Figure 1.10: An example of T-TPN

This sytem can be modelled by the T-TPN represented on the Figure 1.10. We see that both of the processes are in their initial states P1_1 for P1 and P2_1 for P2; P2 can send a message when it wants because the interval for T2_1 is $[0,\infty[$ and P1 waits for a message; when P2 has sent a message, the transition T1_1 becomes enabled but it can be fired only after at least one time unit (the minimum time that takes the message to go from P2 to P1),but before three time units; hence the interval of T1_1 is $[1,3]$. The sending and receiving of acknowledgements follow similarly.

1.3.2 The T-TPN formalism

Syntax

A T-TPN $\mathcal{T}$ [4],[7] is a tuple $\langle P, T, B, F, M_0, (\alpha, \beta)\rangle$ where :

- $P = \{p_1, ..., p_m\}$ is a finite set of places;
- $T = \{t_1, ..., t_n\}$ is a finite set of transitions;
- $B : (\mathbb{N}^P)^T$ is the backward incidence mapping;
- $F : (\mathbb{N}^P)^T$ is the forward incidence mapping;
- $M^0 \in \mathbb{N}^P$ is the initial marking;
- $\alpha \in (\mathbb{Q}_{\geq 0})^T$ and $\beta \in (\mathbb{Q}_{\geq 0} \cup \{\infty\})^T$ are the earliest and latest firing time mappings.

We remark that, except for the firing time mappings, the syntax is the same as the syntax of a Petri net.

Semantics

The behavior of a T-TPN $\mathcal{T}$ can be represented with a timed transition system $S_{\mathcal{T}}$ ([4], [7]). The following definitions are useful (note : the operations on the vector correspond to the operations on the elements extended to the vectors) :

- A marking is an element of $\mathbb{N}^P$;
- A valuation is a vector $v \in (\mathbb{R}_{\geq 0})^n$ such that each value v_i represents the elapsed time since the last time t_i was enabled or since the launching of the system if t_i was never enabled;
- $\bar{0} \in (\mathbb{R}_{\geq 0})^n$ is the initial valuation with $\forall i \in [|1,n|], \bar{0}_i = 0$;
- A transition t is said to be enabled for a marking M if and only if $\forall p \in P, M \geq B(t)$;
- A transition t_k is said to be newly enabled after the firing of a transition t_i from a marking M if t_k is not enabled for the marking $M - B(t)$ and it is enabled for the marking $M' = M - B(t) + F(t)$;

- The function $\uparrow enabled : T \times \mathbb{N}^P \times T \mapsto \{true, false\}$ is defined by $\uparrow enabled(t_k, M, t_i) = true$ if t_k is newly enabled after the firing of t_i from M;
- $\forall (t_k, M, t_i) \in T \times \mathbb{N}^P \times T, \uparrow enabled(t_k, M, t_i) = (M - B(t_i) + F(t_i) \geq B(t_k)) \wedge ((M - B(t_i) < B(t_k)) \vee (t_k = t_i))$.

The timed transition system $S_{\mathcal{T}} = \langle Q, q_0, T, \rightarrow \rangle$ associated to a T-TPN $\mathcal{T} = \langle P, T, B, F, M_0, (\alpha, \beta) \rangle$ is defined by :

- $Q = \mathbb{N}^P \times (\mathbb{R}_{\geq 0})^n$;
- $q_0 = (M_0, \bar{0})$;
- $\rightarrow \in Q \times (T \cup \mathbb{R}_{\geq 0}) \times Q$ is the transition relation defined by :
  1. The discrete transitions are defined by $\forall t_i \in T$:
  $$(M, v) \xrightarrow{t_i} (M', v') \Leftrightarrow \begin{cases} M \geq B(t_i) \wedge M' = M - B(t_i) + F(t_i) \\ \alpha(t_i) \leq v_i \leq \beta(t_i) \\ v'_k = \begin{cases} 0 & \text{if } \uparrow enabled(t_k, M, t_i) \\ v_k & otherwise \end{cases} \end{cases}$$
  2. The continuous transitions are defined by $\forall \delta \in \mathbb{R}_{\geq 0}$:
  $$(M, v) \xrightarrow{\delta} (M, v') \Leftrightarrow \begin{cases} v' = v + \delta \\ \forall k \in [|1, n|], (M \geq B(t_k) \Rightarrow v'_k \leq \beta(t_k)) \end{cases}$$

The last condition on contiunous transitions ensures that the time that elapses in places cannot increase to a value which would disable transitions that were enabled by the marking. We use $Reach(\mathcal{T})$ to denote the set of markings associated with the set of reachable states of the timed transition systems $S_{\mathcal{T}}$. If $Reach(\mathcal{T})$ is finite, then we say that $\mathcal{T}$ is bounded.

1.3.3 Reachability analysis of T-TPN

Presentation of the problem

The reachability problem associated to a T-TPN $\mathcal{T}$ consists in determining if a marking M of $\mathcal{T}$ is reachable from the initial marking of $\mathcal{T}$. For the T-TPN a similar problem occurs as with timed automata: it is not possible to work directly on the timed transition system, which represents the behavior of the T-TPN, since this timed transition system has infinitely many states and infinitely many labels. In this section we will study two methods that permit the construction of a transition system where the labels expressing the elapsing of the time are eliminated and where the states are regrouped into state classes on which the reachability analysis can be done. However these methods do not always give a result because of the following theorem.

Theorem ([4]) For T-TPN the problems of reachability and of boundedness are undecidable.

State Class Graph

In this part we show how to build a graph which allows reachability analysis and which is called the *state class graph* [4], [21], [12]. To build this graph we need to define the notation I_M which represents the indexes of the transitions that are enabled by the marking M. Suppose that the set T of the transitions of a T-TPN $\mathcal{T}$ is denoted $T = \{t_1, ...t_m\}$. Then, for a T-TPN $\mathcal{T} = \langle P, T, B, F, M_0, (\alpha, \beta)\rangle$ we have :

$$\forall M \in \mathbb{N}^P, I_M = \{i \in [|1, m|] | M \geq B(t_i)\}$$

Then we can define a state class C of a T-TPN $\mathcal{T}$ as a pair (M, D) where M is a marking of $\mathcal{T}$ and D is called the firing domain. The firing domain represents a set of inequalities on variables θ_i, where θ_i represents, for each transition t_i enabled by M, the possible firing time of t_i relative to the time when the class C was entered. Formally D can be denoted $D = \{\theta = [\theta_i]_{i \in I_M} | A \cdot \theta \geq b\}$ where A is a matrix, b is a vector of constants and θ is a vector of elements θ_i indexed by I_M.
Given a class $C = (M, D)$ it is possible to determine the firability of a transition t_i; we say that t_i is firable from the class $C = (M, D)$ (with $D = \{\theta = [\theta_i]_{i \in I_M} | A \cdot \theta \geq b\}$) if and only if :

(i) $i \in I_M$;

(ii) the system of inequalities defined by :

$$\begin{cases} A \cdot \theta \geq b \\ \theta_i \leq \theta_j \quad \forall j \in I_M, j \neq i \end{cases}$$

has a solution.

The inequation of the form $\theta_i \leq \theta_j$ ensures that the elapsing time in the places will not disable transitions that are enabled by the marking. In fact, the system $A \cdot \theta \geq b$ gives inequalities on the relative firing time θ_j, and by adding the inequalities of the form $\theta_i \leq \theta_j$, it is ensured that the firing time of the chosen transition cannot be superior to a possible firing time of another transition.

If we suppose that a firable transition $t_f \in T$ is fired from a class $C = (M, D)$ with $D = \{\theta = [\theta_i]_{i \in I_M} | A \cdot \theta \geq b\}$, then the class $C' = (M', D')$ obtained is computed as follows :

a) $M' = M - B(t_f) + F(t_f)$

b) D' is computed in three steps :

1. In the system of inequalities:

$$\begin{cases} A \cdot \theta \geq b \\ \theta_f \leq \theta_j \quad \forall j \in I_M, j \neq f \end{cases}$$

the change of variables $\theta_j = \theta_f + \theta''_j$ is made $\forall j \in I_M$ such that $j \neq f$; then the variable θ_f is removed (using a method such as Fourier-Motzkin method) so that the system obtained can be denoted :

$$\begin{cases} A'' \cdot \theta'' \geq b'' \\ \theta'' \geq 0 \end{cases}$$

such that the vector θ'' is indexed by $I_M \setminus \{f\}$.

2. Then the variables θ_i such that $i \notin I_M \setminus \{f\} \cap I_{M-B(t_f)}$ (i.e. the variable θ_i that corresponds to transitions t_i that were enabled for M but that are not enabled for the marking $M - B(t_f)$) are removed from the system using the same method as to remove θ_f; in the system of inequalities, it remains only the variables corresponding to transitions enabled for M and M' and not newly enabled;
3. Finally, for all i such that $\uparrow enabled(t_i, M, t_f) = true$, the inequalities $\alpha(t_i) \leq \theta_i \leq \beta(t_i)$ are added to the system of inequalities to obtain :

$$A' \cdot \theta' \geq b'$$

and $D' = \{\theta' = [\theta'_i]_{i \in I'_M} | A' \cdot \theta' \geq b'\}$.

These operations allow us to compute a tree where each node is a state class, that has for sons the classes obtained by firing the firable transitions. Each link between two nodes is then labeled with the name of the fired transition. The root of this tree is the initial state class $C_0 = (M_0, D_0)$ where $D_0 = \{\theta = [\theta_i]_{i \in I_{M_0}} | \forall i \in I_{M_0}, \alpha(t_i) \leq \theta_i \leq \beta(t_i)\}$.By definition of the state classes, any sequences of transitions firable in the T-TPN will be represented by a path in the above tree,which is why the reachability analysis can be done on this tree. Furthermore if the number of nodes of the tree is bounded, a finite graph will be built by regrouping the nodes of the tree that are equal, whrere two classes are equal if they have the same marking and the same firing domain. To determine if two domains are equal, it is possible to transform them to a canonical form [4]. The graph so built is called the *state class graph*.

As we have said, to build the state class graph of a T-TPN $\mathcal{T}$, $\mathcal{T}$ has to have a bounded number of state classes. We define a class of T-TPN, which are the *T-bounded* T-TPN. A T-TPN $\mathcal{T} = \langle P, T, B, F, M_0, (\alpha, \beta) \rangle$ is T-bounded if :

$$(\exists k \in \mathbb{N})(\forall M \in Reach(\mathcal{T}))(\forall t_i \in T)(\exists p \in P) M(p) < (k+1)(B(t_i)(p))$$

which means that no transition can be fired simultaneously more than k times. Then we have the three following theorems.

Theorem ([4]) A T-TPN has a bounded number of state classes if and only if it is bounded.

Theorem (Sufficient Condition 1) ([4]) A T-bounded T-TPN is bounded if there is no pair of state classes $C = (M, D)$ and $C' = (M', D')$ reachable from the initial state class and such that :

(i) C' is reachable from C;

(ii) $M' > M$.

Theorem (Sufficient Condition 2) ([4]) A T-bounded T-TPN is bounded if there is no pair of state classes $C = (M, D)$ and $C' = (M', D')$ reachable from the initial state class and such that :

(i) C' is reachable from C;

(ii) $M' > M$;

(iii) $D' = D$;

(vi) $\forall p \in \{p \in P | M'(p) > M(p)\}, M(p) > max_{i \in [|1,n|]}(B(t_i)(p))$

Then the sufficient condition 2 is used to stop the enumeration of the state classes when a firing sequence is found which does not decrease the marking of any places, and for the places p for which the sequence increases the marking, the marking at the start of the sequence is greater than $max_{i \in [|1,n|]}(B(t_i)(p))$.

Example of the computing of state classes ([4])

In order to understand better how to compute the state classes of a T-TPN, we will give an example for the T-TPN represented on the Figure 1.11.

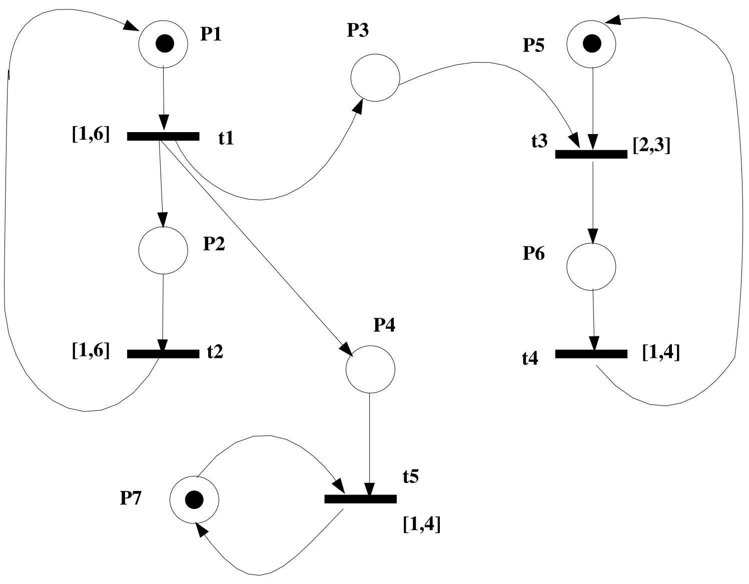


Figure 1.11: A second example of T-TPN

The initial state of this T-TPN is $C_0 = (M_0, D_0)$ with :

$$M_0 = (1,0,0,0,1,0,1) \text{ and } D_0 = \{(\theta_1) | 1 \leq \theta_1 \leq 6\}$$

The transition t_1 is then firable and its firing leads to the state class $C_1 = (M_1, D_1)$ with :

$$M_1 = (0,1,1,1,1,0,1) \text{ and } D_1 = \{(\theta_2, \theta_3, \theta_5) | \begin{cases} 1 \leq \theta_2 \leq 6 \\ 2 \leq \theta_3 \leq 3 \\ 1 \leq \theta_5 \leq 4 \end{cases} \}$$

The transition t_2 can then be fire if the system of inequations :

$$(1) \begin{cases} 1 \leq \theta_2 \leq 6 \\ 2 \leq \theta_3 \leq 3 \\ 1 \leq \theta_5 \leq 4 \\ \theta_2 \leq \theta_3 \\ \theta_2 \leq \theta_5 \end{cases}$$

has a solution, which is the case. The firing of t_2 is then possible if $1 \leq \theta_2 \leq 3$. We denote θ_{2F} the relative time for which t_2 is fired. The new marking obtained is $M_2 = (1,0,1,1,1,0,1)$ for which t_3 and t_5 remain enabled. We perform the change of variables $\theta_3 = \theta'_3 + \theta_{2F}$ and $\theta_5 = \theta'_5 + \theta_{2F}$ in the system (1) which becomes the system :

$$(2) \begin{cases} 1 \leq \theta_{2F} \leq 3 \\ 2 \leq \theta'_3 + \theta_{2F} \leq 3 \\ 1 \leq \theta'_5 + \theta_{2F} \leq 4 \end{cases}$$

We then eliminate θ_{2F} from (2) and we obtain :

$$(3) \begin{cases} 0 \leq \theta'_3 \leq 2 \\ 0 \leq \theta'_5 \leq 3 \\ \theta'_5 - \theta'_3 \leq 2 \\ \theta'_3 - \theta'_5 \leq 2 \end{cases}$$

After the firing of t_2, t_1 becomes newly enabled and we can conclude that the obtained state class $C_2 = (M_2, D_2)$ is such that:

$$M_2 = (1,0,1,1,1,0,1) \text{ and } D_2 = D_1 = \{(\theta_1, \theta_3, \theta_5) | \begin{cases} 1 \leq \theta_1 \leq 6 \\ 0 \leq \theta_3 \leq 2 \\ 0 \leq \theta_5 \leq 3 \\ \theta_5 - \theta_3 \leq 2 \\ \theta_3 - \theta_5 \leq 2 \end{cases} \}$$

Zone graph method

As we have seen, it is possible to build for a T-TPN a so-called state class graph, but even if this graph solves the problem of the infinity of states and labels of the timed transition system associated to a T-TPN, its structure appears to have limitations in particular with regard to the firing domain; in this domain the values of time are relative and the functions used to compute these domains are not bijective, which limits the set of properties that can be verified using this graph. In this section, we present another method to compute the state-space of a bounded T-TPN [12]. This method is an adaptation of the zone based method used for the timed automata (and presented before in this report).

We consider a T-TPN $\mathcal{T} = \langle P,T,B,F,M_0,(\alpha,\beta)\rangle$.To each transition t_i, a clock denoted x_i is associated. We denote $X = \{x_1,...,x_n\}$. Clock zones can then be defined over X the same way as they have be defined for timed automata (see 1.2.3 in this report). This method associates zone to markings. Let M be a marking of $\mathcal{T}$ and Z be a zone over X. The different steps of the computation of the reachable markings from M according to Z are the following :

1. Computation of the elapsing of the time in Z, which is denoted $Z\Uparrow$ in the section 1.2.3;
2. Selection of the possible valuations of clocks for the marking M (i.e. the clock valuations cannot disable a transition that is enabled for the marking). We obtain :
$$Z' = Z \Uparrow \wedge (\bigwedge_{i \in I_M} (x_i \leq \beta(t_i)))$$
Z' is the maximal zone starting from Z for the marking M.
3. Determination of the firable transition. t_i is firable if the clock zone defined by $Z' \cap \{x_i \geq \alpha(t_i)\}$ is not empty;
4. For each firable transition t_i leading to M_i, computation of the zone Z_i associated to the marking M_i:
$$Z_i = (Z \cap \{x_i \geq \alpha(t_i)\})[X_e := 0]$$
where $X_e = \{x_j \in X \mid \uparrow enabled(t_j,M,t_i) = true\}$.

An initial configuration for this algorithm is the marking M_0 associated with the zone for which all the clock values are 0. To test the convergence of this algorithm, a list of zones is associated to each marking, and for each step a comparaison is made which guarantees that zones and markings that already have been computed are not recomputed. However there are some T-TPNs for which this algorithm does not converge.

For example, the algorithm does not converge for the T-TPN represented on the Figure 1.12. In fact, performing the firing sequence $\{t2t3\}^*$, the algorithm computes an infinite number of zones. In fact, $Z'_0 = \{0 \leq x_1 \leq 1 \wedge x_1 - x_2 = 0\}$. Then when firing $t2$ and $t3$, we arrive again in M_0 and the maximal zone obtained is $Z'_1 = \{2 \leq x_1 \leq 3 \wedge x_1 - x_2 = 0\}$;

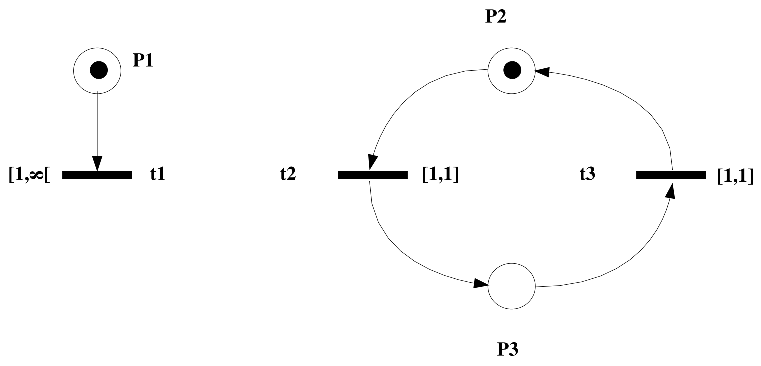


Figure 1.12: An example of T-TPN for which the method does not converge

doing so we can build an infinite set of Z'_i. We remark that if ∞ does not appear in the given T-TPN, the algorithm would converge. However, a general solution to resolve this problem is to use an operator on zones, called *k-approx* that regroups in an equivalence class the zones where the values of clocks linked to an unbounded transition ($[\alpha, \infty[$) are greater than α. Then we have the following theorem and its corollary.

Theorem ([12]) A forward analysis algorithm using k-approx on zone is exact with respect to T-TPN marking reachability for bounded T-TPN.

Corollary ([12]) For a bounded T-TPN without infinity as latest firing time, a foward analysis algorithm using zones computes the exact state-space of the T-TPN.

1.3.4 From a T-TPN to a timed automaton

In this section we will present a method to transform a T-TPN into a timed automaton [7]. This can be useful in practice because there exists an important number of methods and associated tools for analysing the behavior of timed automata. In this section, we will used the notations and the definitions introduced in the part dedicated to the timed automata (part 1.2 of this report).

We consider a T-TPN $\mathcal{T} = \langle P, T, B, F, M_0, (\alpha, \beta) \rangle$ with $P = \{p_1, ..., p_m\}$ and $T = \{t_1, ..., t_n\}$. For each transition t_i, we define a timed automaton $\mathcal{A}_i$ with a clock x_i as the one represented by the Figure 1.13. The states of this automaton are :

- t if the transition is enabled;
- $\bar{t}$ if the transition is disabled;
- $Firing$ if the transition is being fired.

The set of labels of this automaton is $\{?pre, ?post, ?update\}$. The initial state depends of the initial marking, if $M_0 \geq B(t_i)$ the initial state is t else it is $\bar{t}$. This automaton updates an array of integer p which represents the marking of the T-TPN and which is shared by all the timed automata $\mathcal{A}_i$.

In order to build a system with all the automata $\mathcal{A}_i$, another timed automaton *SU* with a clock x called the supervisor is needed. This timed automaton *SU* is represented by the Figure 1.14.

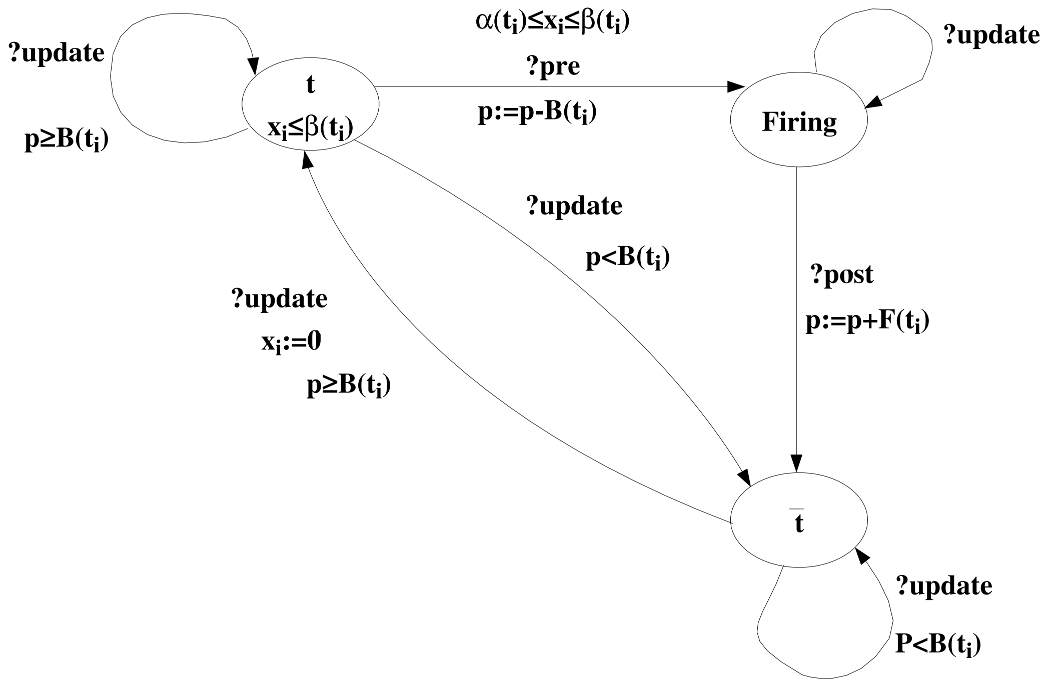


Figure 1.13: The automaton $\mathcal{A}_i$ associated to a transition t_i

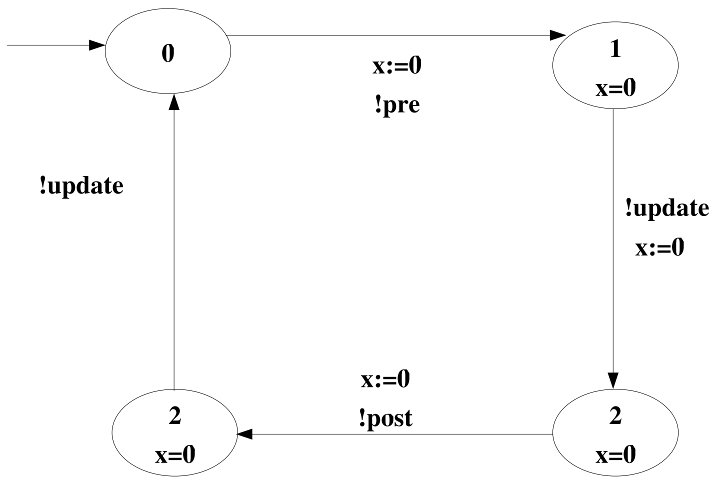


Figure 1.14: The automaton of the supervisor *SU*

In order to regroup these timed automata, we define a set of labels $\Sigma = \{pre_1, ... , pre_n, post_1, ... , post_n, update\}$ and the synchronisation function f such that :

- $f(!pre, \bullet, ..., ?pre, \bullet, ...) = pre_i$ if $?pre$ is the $(i+1)$th argument and all the other arguments are $\bullet$;
- $f(!post, \bullet, ..., ?post, \bullet, ...) = post_i$ if $?post$ is the $(i+1)$th argument and all the other arguments are $\bullet$;
- $f(!update, ?update, ..., ?update) = update$.

The product $\Delta(\mathcal{T}) = (\mathcal{A}_1||...||\mathcal{A}_n||SU)_{f,\Sigma}$ represents the timed automaton associated to the T-TPN $\mathcal{T}$. Note that for the operations concerning the array p, there can only be one update for each transition of $\Delta(\mathcal{T})$; in fact the updates of p are linked to the labels $?pre$ and $?post$, and the synchronisation function ensures that no transition of $\Delta(\mathcal{T})$ corresponds to a double event $?pre$ or a double event $?post$. The conditions on p corresponding to labels $?update$, they are combined on the transitions labeled with $update$ by making their conjunction.

Now that we have described how to build a timed automaton $\Delta(\mathcal{T})$ associated with a T-TPN $\mathcal{T}$, we will present the link between the semantics of $\mathcal{T}$ and that of $\Delta(\mathcal{T})$. Let (M, ν) be a state of $S_{\mathcal{T}}$ and $((s,p), \bar{q}, v)$ a state of $S_{\Delta(\mathcal{T})}$, where s is a location of SU, p is the vector representing the marking, (s,p) is considered as a state of SU, $\bar{q} \in \{t, \bar{t}, Firing\}^n$ represents the states of the $\mathcal{A}_i$ and $v \in (\mathbb{R}_{\geq 0})^n$ represents the values of the clocks x_i. Then we define the relation $\approx$ by :

$$(M,\nu) \approx ((s,p),\bar{q},v) \Leftrightarrow \begin{cases} s = 0 \\ \forall i \in [|1,m|], p[i] = M(p_i) \\ \forall k \in [|1,n|], q_k = \begin{cases} t & if M \geq B(t_k) \\ \bar{t} & else \end{cases} \\ \forall k \in [|1,n|], \nu_k = v_k \end{cases}$$

The following theorem shows that there is effectively a timed bisimulation between the timed transition system associated to $\mathcal{T}$ and the one associated to $\Delta(\mathcal{T})$

Theorem ([7]) For all states (M,ν) of $S_{\mathcal{T}}$ and for all states $((0,p),\bar{q},v)$ of $S_{\Delta(\mathcal{T})}$ such that $(M,\nu) \approx ((0,p),\bar{q},v)$:

$$\forall t_i \in T, (M,\nu) \xrightarrow{t_i} (M',\nu') \Leftrightarrow \begin{cases} ((0,p),\bar{q},v) \xrightarrow{w_i} ((0,p'),\bar{q'},v') \text{ with} \\ w_i = pre_i.update.post_i.update \text{ and} \\ (M',\nu') \approx ((0,p'),\bar{q'},v') \end{cases}$$

$$\forall \delta \in \mathbb{R}_{\geq 0}, (M,\nu) \xrightarrow{\delta} (M',\nu') \Leftrightarrow \begin{cases} ((0,p),\bar{q},v) \xrightarrow{\delta} ((0,p'),\bar{q'},v') \text{ and} \\ (M',\nu') \approx ((0,p'),\bar{q'},v') \end{cases}$$

This theorem allows us to make the analysis of reachability for the T-TPN on the timed automaton, because each sequence of the timed transition system $S_{\mathcal{T}}$ has a corresponding sequence of transitions in the timed transition system associated to $\Delta(\mathcal{T})$ and each sequence of transitions in the timed transition system $S_{\Delta}(\mathcal{T})$ that finishes in a state for which the supervisor is in state 0 has a corresponding sequence of $S_{\mathcal{T}}$ (this second point is due to the fact that all the sequences of transitions of $S_{\Delta(\mathcal{T})}$ that end in a state for which the state of the supervisor is 0 have the form (pre_i.update.post_i.update)*).

Chapter 2

Syntax and Semantic of the Transitions Time Well-formed Nets

2.1 Introduction

The Transitions Time Well-formed Nets that we present in this chapter are models that extend the formalism of Transitions Time Petri Nets introduced in [4] to a formalism with coloured tokens, just as Petri nets were extended to Well-formed Nets, which are presented in [8].

2.2 Presentation of Transitions Time Well-formed Nets *(TTWN)*

To intoduce the formalism, we begin with presenting it in an informal way.

Figure 2.1 gives an example of a TTWN. This example was given without time constraints in [23].This model represents two clients characterized by the values $< 1 >$ and $< 2 >$ connected to two servers $< 1 >$ and $< 2 >$. In the initial state of the system, all the clients are in the state *Cready* and all the servers are in the state *Sready*. A client $< X >$ that is in the state *Cready* can always send (with the transition *cenv*) a message to one of the server $< Y >$ without any time restriction (the interval linked to *cenv* is $[0, \infty]$). When a message characterized by the token $< X, Y >$ arrives in the place $Mess$, if the server $< Y >$ is in the state *Sready*, after 1 time unit and before the 4 following time units, the server $< Y >$ will receive the message (or it will not be in the state *Sready* anymore, if it receives an other message, for instance). After it has received the message, the server $< Y >$ takes at least 2 time units and at most 4 time units to treat it, sends a response with the transition *ssend*, then returns to the state *Sready*. The corresponding client $< X >$ that is in the state *Cwait* takes at least 1 time unit and at most 3 time units to receive the response, then it returns in the state *Cready*.

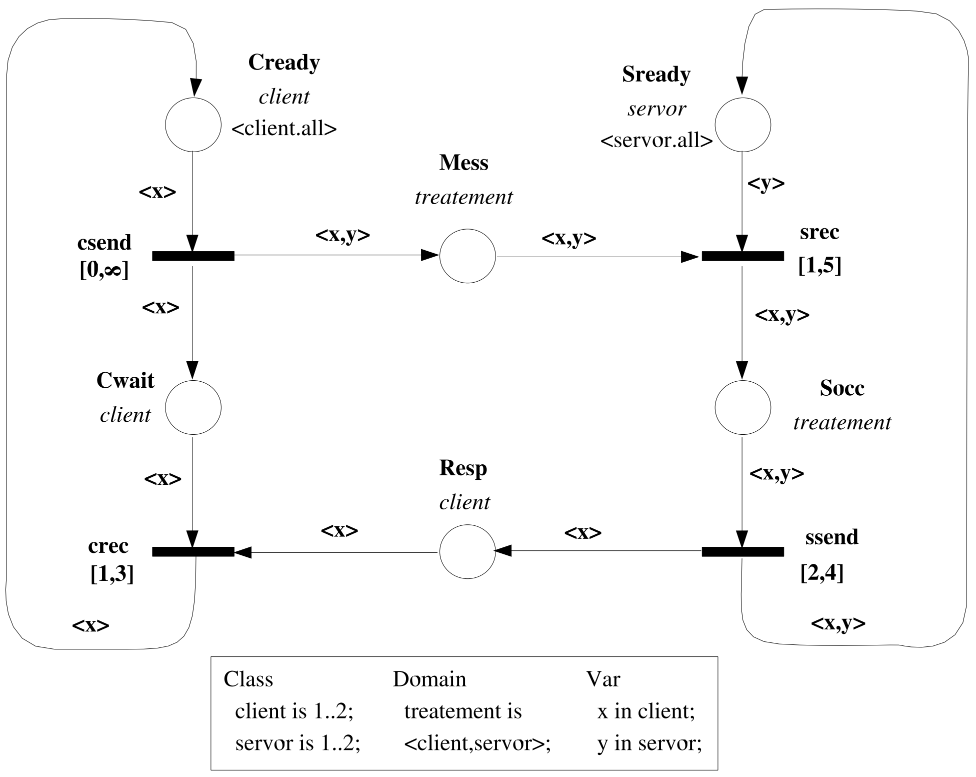


Figure 2.1: An example of Transitions Time Well-formed Nets

To understand better the link between the Transitions Time Petri Nets and the TTWN, it is possible to unfold a TTWN to obtain a Transitions Time Petri Net. For instance the Transitions Time Petri Net represented in figure 2.2 is the result of the unfolding of the TTWN presented on the figure 2.1. For each client and each server, the behavior is represented, as is the communication: since there are 4 possibilities of communication there are four places *Mess_i_j* and four places *Resp_i_j*. The transitions are also duplicated. However, it appears that if the domains and the colour classes were greater, the unfolded net would not be readable. For instance, if we want to add a client in the model, it is necessary to add 9 places and 8 transitions to the Transitions Time Petri Nets; instead it is necessary to extend the colour class *client* to 1..3 in the corresponding TTWN. In the next sections of this chapter, we will define formally this model and then we will give its semantics.

2.3 Useful definitions and notation

Definition 1 *A multi-set over a finite non-empty set Y is a mapping from Y to $\mathbb{N}_{\geq 0}$.*

Each multi-set a over a finite non-empty set Y can be represented by the formal sum $a = \sum_{y \in Y} a(y).y$ (where $a(y)$ represents the number of instances of y in a). For all sets Y, which are finite and non-empty, $Bag(Y)$ represents the set of the multi-sets over Y. $\emptyset_Y$ is the empty multi-set over Y. We will now define some operations on the multi-sets.

Definition 2 *Let Y be a finite and non-empty set. We consider $a, b \in Bag(Y)$ such that*

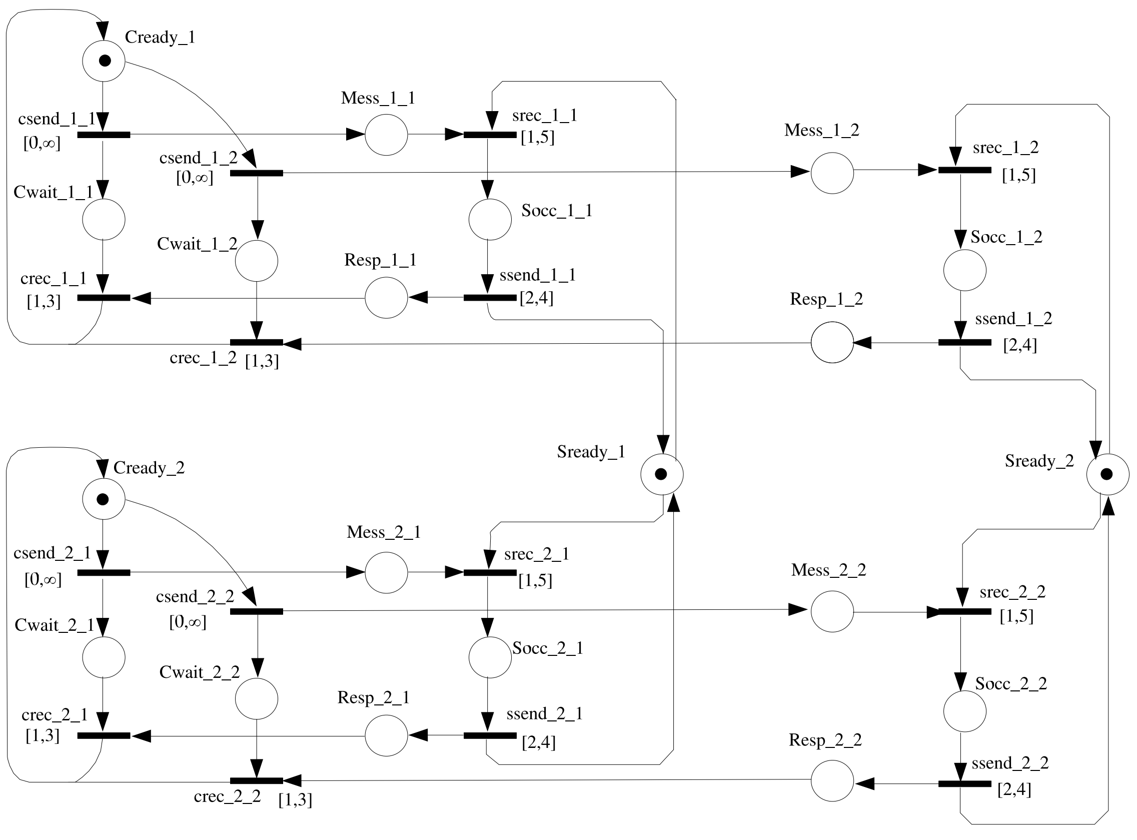


Figure 2.2: An unfolded Transitions Time Well-formed Nets

$a = \sum_{y \in Y} a(y).y$ and $b = \sum_{y \in Y} b(y).y$.

- *The sum $a + b$ of a and b is an element of $Bag(Y)$ defined by $a + b = \sum_{y \in Y}(a(y) + b(y)).y$;*
- *If λ is a positive integer, then the product $\lambda.a$ is an element of $Bag(Y)$ defined by $\lambda.a = \sum_{y \in Y}(\lambda.a(y)).y$;*
- *a is greater or equal to b (denoted $a \geq b$) if and only if : $\forall y \in Y$, $a(y) \geq b(y)$.*
- *a is equal to b (denoted $a = b$) if and only if: $a \geq b$ and $b \geq a$;*
- *a is strictly greater than b (denoted $a > b$) if and only if : $a \geq b$ and $a \neq b$.*

If we consider $C_1, ..., C_k$, k finite sets, it is possible to identify $Bag(C_1 \times ... \times C_k)$ with $Bag(C_1) \times ... \times Bag(C_k)$ with the following rules :

- $\langle c_1, ..., \lambda.c_i, ..., c_k \rangle = \lambda.\langle c_1, ..., c_i, ..., c_k \rangle$;

- $\langle c_1, ..., c_i + c'_i, ..., c_k \rangle = \langle c_1, ..., c_i, ..., c_k \rangle + \langle c_1, ..., c'_i, ..., c_k \rangle$.

For instance, we consider $C_1 = \{a, b\}$ and $C_2 = \{x, y\}$. The element $m = \langle 2.a + b, x + y \rangle$ belongs to $Bag(C_1) \times Bag(C_2)$, and by following the rules defined above,we obtain $m = 2.\langle a, x \rangle + 2.\langle a, y \rangle + \langle b, x \rangle + \langle b, y \rangle$ which is an element of $Bag(C_1 \times C_2)$.

2.4 Syntax

We will now define the syntax of the Transitions Time Well-formed Nets. First we will see the kind of sets to which the tokens can belong, then we will present the form of the functions that are linked to the arcs and the form of the guards that can also be defined in these nets.

2.4.1 Colour classes and colour domains

In a TTWN a colour is associated to each token. These colours belong to colour domains that are cartesian products of colour classes. A colour class is a finite and non-empty set of terminal colours (which means that the definition of these terminal colours does not depend on the other colours). A colour class can be ordered and also parameterized by an integer. It is useful to have an ordered class to deal with the notion of successor. For instance if we consider the ordered colour class $C = \{c_1, c_2, ..., c_n\}$, the successor of the objet c_i is denoted $\oplus c_i$, and for all i in $[|1,n|]$ $\oplus c_i = c_{(i+1) mod n}$ (which means that for instance c_2 is the successor of c_1 and c_1 is the successor of c_n). A colour class can be partitioned into static subclasses, in order to group in the same class elements that can have different behavior. This partitioning is static because it is defined with the colour class. All the static subclasses of a colour class are disjoint.

Definition 3 *A colour domain is a finite cartesian product of colour classes. Each colour class is a finite set of elements. We denote $Cl = \{C_1, ..., C_k\}$ the set of colour classes of a TTWN. Cl is called the family of colour classes and is such that $\forall i \neq j \in [|1,k|]$, $C_i \cap C_j = \emptyset$.*

A colour class can be ordered. The successor of an object c in an ordered class C_i is denoted $\oplus c$. The operator $\oplus$ is such that for all c_j in an ordered class C_i, $\bigcup_{l=1}^{|C_i|} \{\oplus^l c_j\} = C_i$.

When it is defined, a colour class can be partitioned into static subclasses. The q^{th} static subclass of a partitioned class C_i is denoted $C_{i,q}$ and if s_i is the number of static subclasses of C_i, then $C_i = \biguplus_{q=1,...,s_i} C_{i,q}$ (in fact the static subclasses are disjoint).

A colour domain is generally denoted $C = C_1^{e_1} \times ... \times C_k^{e_k}$ or $C = \bigotimes_{i=1}^{k} (C_i)^{e_i}$ where for i in $[|1,k|]$, e_i is an integer, positive or equal to zero, which represents the number of times that the colour class C_i appears in the colour domain C. When $e_i = 0$ for all i in $[|1,k|]$, C is the neutral domain, denoted $\{\varepsilon\}$ (ε is the neutral colour). An object tuple $c \in C$ is denoted $c = \bigotimes_{i=1}^{k} \bigotimes_{j=1}^{e_i} c_i^j = \langle c_1^1, ..., c_1^{e_1}, ..., c_k^1, ..., c_k^{e_k} \rangle$.

2.4.2 Colour functions

In ths section, we will describe the different functions that appear on the arcs of a TTWN. We will first define basic functions, then guarded functions and finally we will present the standard functions which label arcs.

Basic colour functions

There are three elementary colour functions. Each of them takes an element of a colour domain C' and gives a result in a multi-set over a colour class C_i. The identity function (X_{C_i}) allows the selection of an element of C_i. The successor function ($\oplus X_{C_i}$) allows the selection of the successor of an element of C_i (this function can be applied only if the colour class C_i is ordered). Finally the diffusion function (All_{C_i}) allows to consider all the elements of C_i.

Definition 4 *Let C_i be a colour class and $C' = C_1^{e_1} \times ... \times C_k^{e_k}$ be a colour domain. The elementary colour functions are defined from C' to $Bag(C_i)$ by $\forall c = \langle c_1^1, ..., c_1^{e_1}, ..., c_k^1, ..., c_k^{e_k} \rangle$:*

- *$X_{C_i}^j(c) = c_i^j$ (for all j such that $1 \leq j \leq e_i$);*
- *$\oplus X_{C_i}^j(c) = \oplus c_i^j$ if C_i is ordered (for all j such that $1 \leq j \leq e_i$);*
- *$All_{C_i}(c) = \sum_{x \in C_i} x$ and if C_i is partitioned such that $C_i = \biguplus_{q=1,...,s_i} C_{i,q}$, $All_{C_{i,q}}(c) = \sum_{x \in C_{i,q}} x$ (for all q such that $1 \leq q \leq s_i$).*

If C_i is the neutral domain, the only basic colour function possible is All_ε, which is equivalent to an arc with the valuation 1 in an ordinary Petri net.

With these elementary functions, it is possible to define basic colour functions over a colour class C_i as a linear combination of elementary colour functions.

Definition 5 *Let C_i be a colour class and $C' = C_1^{e_1} \times ... \times C_k^{e_k}$ be a colour domain. The set of basic colour functions over the colour class C_i from C' is denoted $F_{C_i,C'}$. It represents functions from C' to $Bag(C_i)$ and is defined by :*

$$F_{C_i,C'} = \left\{ \sum_{q=1}^{s_i} \alpha_{i,q} All_{C_{i,q}} + \sum_{k=1}^{e_i} \beta_{i,k} X_i^k \,\middle|\, Inf_{q \in [|1,s_i|], K \subseteq [|1,e_i|]} (\alpha_{i,q} + \sum_{k \in K} \beta_{i,k}) \geq 0 \right\}$$

if C_i is not ordered.

$$F_{C_i,C'} = \left\{ \sum_{q=1}^{s_i} \alpha_{i,q} All_{C_{i,q}} + \sum_{k=1}^{e_i} (\beta_{i,k}.X_i^k + \gamma_{i,k}. \oplus X_i^k) \,\middle|\, Inf_{q \in [|1,s_i|], K \subseteq [|1,e_i|]} (\alpha_{i,q} + \sum_{k \in K} Inf(\beta_{i,k}, \gamma_{i,k})) \geq 0 \right\}$$

if C_i is ordered.
In these formula, the α, β, γ are integers.

The condition that appears in the definition of the basic colour functions over the linear coefficients ensures that always a positive number of colours is chosen.

From the above definition, it is possible to define the basic colour functions over a colour domain C as a tuple of basic colour functions over the colour classes that compose C.

Definition 6 *Let $C = C_1^{e_1} \times ... \times C_k^{e_k}$ and $C' = C_1'^{e_1} \times ... \times C_k'^{e_k}$ be two colour domains. The set of basic colour functions over the colour domain C from C' is denoted $F_{C,C'}$. It represents functions from C' to $Bag(C)$ and it is defined by :*

$$F_{C,C'} = \left\{ \otimes_{i=1}^{k} \otimes_{j=1}^{e_i} f_i^j = \langle f_1^1, ..., f_1^{e_1}, ..., f_k^1, ... f_k^{e_k} \rangle such\ that f_i^j \in F_{C_i,C'} \right\}$$

Guards and guarded functions

Just as for Well-formed Nets, it is possible to define for the TTWN guards over functions and over transitions.

Definition 7 *A guard is a boolean function over a colour domain $C = C_1^{e_1} \times ... \times C_k^{e_k}$ which is defined by the following standard predicates $\forall c = \langle c_1^1, ..., c_1^{e_1}, ..., c_k^1, ..., c_k^{e_k} \rangle \in C$:*

1. *$[X_{C_i}^{i_1} = X_{C_i}^{i_2}](c)$ is True iff $c_i^{i_1} = c_i^{i_2}$;*
2. *if C_i is ordered, $[X_{C_i}^{i_1} = \oplus X_{C_i}^{i_2}](c)$ is True iff $c_i^{i_1} = \oplus c_i^{i_2}$;*
3. *if C_i is partitioned such that $C_i = \biguplus_{q=1,...,s_i} C_{i,q}$, $[X_{C_i}^{i_1} \in C_{i,q}](c)$ is True iff $c_i^{i_1} \in C_{i,q}$;*
4. *if g_1 and g_2 are guards, $g_1 \vee g_2$, $g_1 \wedge g_2$ and $\neg g_1$ are standard predicates where $\vee, \wedge, \neg$ refer to the classical boolean connectives.*

Now we will present the concept of guarded colour functions which associates a basic colour function with a guard.

Definition 8 *Let C and C' be two colour domains. Let $\phi_{C'}$ be a guard over C' and f be a basic colour function from C' over C. A guarded function g from C' over C is defined by $\forall c \in C'$:*

$$g(c) = [\phi_{C'}]f(c) \overset{def}{=} \begin{cases} f(c) & \textit{if } [\phi_{C'}](c) \\ \emptyset_C & \textit{else} \end{cases}$$

Standard functions

Definition 9 *A standard colour function from a colour domain C' over a colour domain C is a sum of guarded functions from C' over C. Therefore, it takes the form of a function from C' to $Bag(C)$.*

Notations

In order to simplify the model when it is represented graphically, the following shorthands are possible for the introduced notations :

- If there is only a single colour class C_1, All is a shorthand for All_{C_1}
- X_i is a shorthand for X_i^1 when C_i is instantiated only once ;
- X^k is a shorthand for X_1^k when there is only one colour class involved;
- X is a shorthand for X_1^1 when there is a single colour class instantiated once.

Sometimes for the description of the guarded functions it is possible to describe directly the coloured tokens that will be taken or given by the firing of the transition. For example, if we consider the class $C = \{1\} \uplus \{2,3\}$ and the guarded function from C over C defined by $< X >$ associated to a transition whose guard is $[X = 1]$ (the basic functions are represented between the symbols $<$ and $>$ in the nets), the corresponding arc will be labeled with $< 1 >$ (and the guards on the transition will not be written), which means that the only token that is involved is the one whose colour is 1. The variable X can as well be written differently, for instance Z or Y or even names which are more complicated, provided that the name is not All or S. Note that, in the literature on Well-formed Nets, $All_{C_{i,q}}$ is sometimes written $S_{C_{i,q}}$ or as well $C_{i,q}.All$. For the successor, the symbol $\oplus$ is sometimes replaced by the symbol !. In a net, if there are two input arcs on a transition which deal with the same variables, for instance $< X >$, implicitly there is a guard on the transition that ensures that the values of the two tokens transported on the input arcs are the same (in fact, it is the same as to replace $< X >$ by $< Y >$ on one of the arcs, and to put the guard $[X = Y]$ on the corresponding transition). Most of the time the colour domains of the transitions are not written on a graph, because they can be directly deduced from the functions linked to the input and output arcs. Most of these notations depend of the modeler and of the tools used to define the nets.

2.4.3 Syntax of the formalism of the Transitions Time Well-formed Nets

With the above definitions, it is possible to define the syntax of a TTWN.

Definition 10 *A Transition Time Well-formed Net is a 9-tuple* $TTWN = \langle P, T, W^-, W^+, Cl, \mathcal{C}, \Phi, (\alpha, \beta), m_0 \rangle$ *where :*

- $P = \{p_1, ..., p_m\}$ *is a finite set of places;*
- $T = \{t_1, ..., t_n\}$ *is a finite set of transitions such that* $P \cup T \neq \emptyset$ *and* $P \cap T = \emptyset$*;*
- $Cl = \{C_1, ..., C_k\}$ *is the set of colour classes; each* C_i *is a finite non-empty set; for all* i, j *in* $[|1,k|]$ *such that* $i \neq j$, $C_i \cap C_j = \emptyset$*; each* C_i *can be partitioned in* s_i *static subclasses* ($C_i = \biguplus_{q=1,...,s_i} C_{i,q}$)*;*
- $\mathcal{C}$ *is the function that denotes the colour domain of each transition and each place; the domain of* $\mathcal{C}$ *is* $P \cup T$ *and its codomain is* ω*, where* ω *is a set that contains the finite cartesian product of elements of* Cl*;*
- W^+ *(resp.* W^-*) is the forward incidence function (resp. the backward incidence function) that associates to all pairs* (p,t) *of* $P \times T$ *a standard colour function from* $\mathcal{C}(t)$ *over* $\mathcal{C}(p)$ *(a function from* $\mathcal{C}(t)$ *to* $Bag(\mathcal{C}(p))$*);*
- Φ *is a function that associates to each transition a guard (* $\forall t \in T, \Phi(t) : \mathcal{C}(t) \rightarrow \{True, False\}$ *); by default,* $\forall t \in T, \Phi(t) = True$*;*
- $\alpha \in (\mathbb{Q}_{\geq 0})^T$ *and* $\beta \in (\mathbb{Q}_{\geq 0} \cup \{\infty\})^T$ *represent the earliest and latest firing time mappings;*
- m_0 *is the initial marking* $\forall p \in P$, $m_0(p) \in Bag(\mathcal{C}(p))$.

The marking represents the position of the tokens; a marking M associates to each place p a multi-set of $Bag(\mathcal{C}(p))$. We denote $\mathcal{MA}$ the set of the markings of a TTWN. For simplicity, we do not include a notion of priority of transitions, which is usually defined for Well-formed Nets.

2.5 Semantics

Just as forTransitions Time Petri Nets, the semantics of a TTWN $\mathcal{T} = \langle P, T, W^-, W^+, Cl, \mathcal{C}, \Phi, (\alpha, \beta), m_0 \rangle$ can be represented as a timed transition system $S_\mathcal{T}$. First the following definitions are needed.

2.5.1 Useful definitions

We will denote by $\mathcal{CT}$ the set of pairs which combine the transitions with a colour of their colour domain, $\mathcal{CT} = \{(t, c_t) | t \in T \wedge c_t \in \mathcal{C}(t)\}$.

Definition 11 *A transition t is said to be enabled for a colour $c \in \mathcal{C}(t)$ in a marking M if and only if*

$$\forall p \in P, W^-(p,t)(c) \leq M(p) \textit{ and } \Phi(t)(c) = True.$$

When an enabled transition t is fired for a colour $c \in \mathcal{C}(t)$ from a marking M, the new marking obtained is $M' = M[t, c>$, where $\forall p \in P, M'(p) = M(p) - W^-(p,t)(c) + W^+(p,t)(c)$.

Definition 12 *A transition t_k is said to be newly enabled for a colour $c_k \in \mathcal{C}(t_k)$ after the firing of a transition t_i for a colour c_i from a marking M if t_k is not enabled for the marking M'' where M'' is such that $\forall p \in P, M''(p) = M(p) - W^-(p,t_i)(c_i)$, and it is enabled for the marking $M' = M[t, c>$ (if $(t_i, c_i) = (t_k, c_k)$, the transition is newly enabled only if t_k is enabled for c_k from the marking M').*

We introduce then the boolean function $\uparrow enabled : \mathcal{CT} \times \mathcal{MA} \times \mathcal{CT}$ defined by $\forall (t_k, c_k)$, $(t_i, c_i) \in \mathcal{CT}$, for all markings $M \in \mathcal{MA}$ such that t_i is fired for c_i from M, $\uparrow enabled((t_k, c_k)$, $M, (t_i, c_i)) = True$ if and only if t_k is newly enabled for c_k after the firing of t_i for c_i from the marking M. We deduced from this definition that $\forall (t_k, c_k), (t_i, c_i) \in \mathcal{CT}, \forall M \in \mathcal{MA}$ such that t_i is fired for c_i from M:

$$\uparrow enabled((t_k, c_k), M, (t_i, c_i)) = (\bigwedge_{p \in P} (M(p) - W^-(p,t_i)(c_i) < W^-(p,t_k)(c_k)) \vee ((t_k, c_k) = (t_i, c_i))) \wedge$$

$$(\bigwedge_{p \in P} (M(p) - W^-(p,t_i)(c_i) + W^+(p,t_i)(c_i) \geq W^-(p,t_k)(c_k))) \wedge \Phi(t_k)(c_k)$$

Definition 13 *A valuation is a function v that associates to each pair $(t, c) \in \mathcal{CT}$ a value $v(t, c) \in \mathbb{R}_{\geq 0}$ which represents the elapsed time since the last time the transition t was enabled for the colour c or since the launching of the system if t was never enabled for the colour c. $\bar{0}$ is the initial valuation such that $\forall t \in T, \forall c \in \mathcal{C}(t)$, $\bar{0}(t, c) = 0$. We denote $\mathcal{V}$ the set of the valuations.*

2.5.2 Semantics of Transitions Time Well-formed Nets

Definition 14 *The timed transition system $S_\mathcal{T} = \langle Q, q_0, \mathcal{CT}, \rightarrow \rangle$ associated to a TTWN $\mathcal{T} = \langle P, T, W^-, W^+, Cl, \mathcal{C}, \Phi, (\alpha, \beta), m_0 \rangle$ is defined by :*

- $Q = \mathcal{MA} \times \mathcal{V}$ *;*

- $q_0 = (m_0, \bar{0})$;
- $\rightarrow \in Q \times (\mathcal{CT} \cup \mathbb{R}_{\geq 0}) \times Q$ *is the transition relation defined by :*

  1. *The discrete transitions are defined* $\forall (t,c) \in \mathcal{CT}, (M,v) \overset{(t,c)}{\rightarrow} (M',v')$ *if and only if:*

$$\begin{cases} \forall p \in P, M(p) \geq W^-(p,t)(c) \text{ and} \\ \Phi(t)(c) = True \text{ and} \\ \forall p \in P, M'(p) = M(p) - W^-(p,t)(c) + W^+(p,t)(c) \text{ and} \\ \alpha(t) \leq v(t,c) \leq \beta(t) \text{ and} \\ \forall (t',c') \in \mathcal{CT}, v'(t',c') = \begin{cases} 0 & if \uparrow enabled((t',c'),M,(t,c)) = True \\ v(t',c') & otherwise \end{cases} \end{cases}$$

  2. *The continuous transitions are defined by* $\forall \delta \in \mathbb{R}_{\geq 0}, (M,v) \overset{\delta}{\rightarrow} (M,v')$ *if and only if :*

$$\begin{cases} \forall (t,c) \in \mathcal{CT}, v'(t,c) = v(t,c) + \delta \text{ and} \\ \forall (t,c) \in \mathcal{CT}, (\forall p \in P, M(p) \geq W^-(p,t)(c)) \Rightarrow (v'(t,c) \leq \beta(t)) \end{cases}$$

Definition 15 *We denote* $\bar{S}_{\mathcal{T}}$ *the timed transition system obtained from* $S_{\mathcal{T}}$ *by considering only the states reachable from the initial state. We denote* $\bar{Q}_{\mathcal{T}}$ *the set of the reachable states.*

The semantics adopted for the firing of the transitions is said to be single-server because if a transition is for instance two times simultaneously enabled for the same colour, we take account only the fact that it is enabled and after the firing it becomes newly enabled.

Remark Because of the possible valuations and the continuous transitions, the state-graph corresponding to this timed transition system has an infinite number of labels and an infinite number of states. However, the set of events is finite, in fact the set of transitions is finite and for each transition its colour domain is a finite cartesian product of colour classes (which are finite sets).

Chapter 3

State class Timed Automaton of Transitions Time Well-formed Nets

3.1 Introduction

As we have seen in chapter 2, the timed transition system which represents the behavior of a Transitions Time Well-formed Net (TTWN) is appropriate for traditional, finite-state automatic verification methods because it has an infinite number of states and an infinite number of labels. The same problem arose for the Transitions Time Petri Nets, and one of the solution proposed in [21] was to compute a timed automaton, which would have a behavior similar to the one of the Transitions Time Petri Nets and then to verify properties on this timed automaton using tools that already exist. We present in this chapter an adaptation of this method to the case of TTWNs.

3.2 Timed automata with clock renaming

To build the state class timed automaton of a TTWN, we need to extend the syntax and the semantics that we gave from the timed automata in chapter 1 to include renaming of clocks.

3.2.1 Syntax

The renaming of clocks is linked to switches, and means that a clock can take the value of another clock. Formally a timed automaton (with renalming) is a tuple $\langle L, L^0, \Sigma, X, I, E\rangle$ where :

- L is a finite set of locations;
- $L^0 \subseteq L$ is the set of the initial locations;
- Σ is a finite set of labels;
- X is a finite set of clocks;

- I is a total function $L \mapsto \Phi(X)$ that associates to each location an invariant (i.e. a clock constraint);
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times 2^{X^2} \times L$ represents the set of the switches. A switch $t = (s, a, \varphi, \lambda, \rho, s') \in E$ represents a switch from s to s' on the label (or event) a, φ is the clock constraint (or guard) associated to this switch, $\lambda \subseteq X$ gives the set of clocks that have to be reset when the switch is executed, and ρ is the renaming function that associates to a clock $x \in X$ another clock $y \in X$ such that $y = \rho(x)$.

If for a switch $t = (s, a, \varphi, \lambda, \rho, s')$, the renaming function is such that there exists $x, y \in X$ such that $x \neq y$ and $\rho(x) = y$, we will label the corresponding edge $x := y$ to characterize the renaming of the clocks. Considering the introduction of renaming functions in the automaton, we need a new notation for the clock interpretations :

- $\forall \rho \in 2^{X^2}$, and for a clock interpretation v, the clock interpretation $v[\rho]$ is the clock interpretation u such that $\forall x \in X$, $u(x) = v(\rho(x))$.

3.2.2 Semantics

For the semantics, this extension of timed automata with clock renaming only brings a change for the discrete transitions of the timed transition system $S_A = \langle Q_A, Q_A^0, \Sigma, \rightarrow \rangle$ which describes the behavior of the timed automaton A. We recall that $Q_A = \{(s, v) \in L \times V(X) | v \models I(s)\}$ and $Q_A^0 = \{(s, \bar{0}) \in L \times V(X) | s \in L^0 and \bar{0} \models I(s)\}$. The transition relation $\rightarrow \subseteq Q_A \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times Q_A$ is defined by :

- for continuous transitions :
  $\forall \delta \in \mathbb{R}_{\geq 0}, (s, v) \xrightarrow{\delta} (s, v + \delta)$ if $\forall \delta' \in [0, \delta], v + \delta$ satisfies $I(s)$;
- for discrete transitions :
  for all states (s, v) of S_A and for all switches $(s, a, \varphi, \lambda, \rho, s')$ such that v satisfies φ, there is the transition $(s, v) \xrightarrow{a} (s', v[\rho][\lambda := 0])$.

We remark that the renaming of clocks is done before the reset of clocks.

3.3 Extended state class graph

In this section, we will present an adaptation of the extended state class graph presented in [21] to the Transitions Time Well-formed Nets. For a TTWN $\mathcal{T} = \langle P, T, W^-, W^+, Cl, \mathcal{C}, \Phi, (\alpha, \beta), m_0 \rangle$, the associated extended state class graph $\Delta'(\mathcal{T})$ is a transition system where the nodes are so-called extended state classes, and they are linked with edges labeled with elements of $\mathcal{CT} = \{(t, c) | t \in T \wedge c \in \mathcal{C}(t)\}$. We recall that $\mathcal{MA}$ represents the set of the markings of a TTWN.

3.3.1 Extended state class

Definition 16 *An extended state class of a Transitions Time Well-formed Net $\mathcal{T} = \langle P, T, W^-, W^+, Cl, \mathcal{C}, \Phi, (\alpha, \beta), m_0 \rangle$ is a tuple $\langle M, D, \chi, trans \rangle$ where M is a marking belonging*

to $\mathcal{MA}$, D is a firing domain, χ is a set of real valued clocks and $trans \in (2^{\mathcal{CT}})^{\chi}$ maps clocks to sets of transitions and colours of their colour domain.
The firing domain D is a set of inequalities. For each transition $t_i \in T = \{t_1, ..., t_n\}$ the colour domain $\mathcal{C}(t_i)$ can be indexed since it is finite. We denote n_i the number of elements of $\mathcal{C}(t_i)$, and write $\mathcal{C}(t_i) = \{c_{i,j} | 1 \leq j \leq n_i\}$.
The inequalities in D are then of the form :

$$\begin{cases} a_{i,j} \leq \theta_{i,j} \leq b_{i,j} \text{ with } a_{i,j} \in \mathbb{Q}_{\geq 0}, b_{i,j} \in (\mathbb{Q}_{\geq 0} \cup \{\infty\}) \\ (\forall i, j \text{ such that } t_i \text{ is enabled for } c_{i,j} \text{ in } M) \\ \\ -\gamma_{k,l,i,j} \leq \theta_{i,j} - \theta_{k,l} \leq \gamma_{i,j,k,l} \text{ with } \gamma_{k,l,i,j}, \gamma_{i,j,k,l} \in (\mathbb{Q}_{\geq 0} \cup \{\infty\}) \\ (\forall i, j, k, l \text{ such that } (i, j) \neq (k, l) \\ \text{ and } t_i \text{ is enabled for } c_{i,j} \text{ and } t_k \text{ is enabled for } c_{k,l} \text{ in } M) \end{cases}$$

We will denote $\mathcal{EC}$ the set of the extended state classes of a TTWN.

Given a set of inequalities D, the set of its solutions will be denoted $[|D|]$. The relation $trans$ associates to each clock $x \in \chi$ a set of elements of $\mathcal{CT}$ such that $\forall (t, c) \in trans(x)$, x represents the value linked to (t, c). Each pair $(t, c) \in \mathcal{CT}$ must be associated to only one clock (i.e. $\forall (t, c) \in \mathcal{CT}, |trans^{-1}((t, c))| \leq 1$).

To build the graph, we also need the notion of clock similarity, in order to group certain extended state classes together.

Definition 17 *Two extended state classes $C = \langle M, D, \chi, trans\rangle$ and $C' = \langle M', D', \chi', trans'\rangle$ are clock-similar, denoted $C \approx C'$, if and only if they have the same markings, the same number of clocks and their clocks are mapped to the same elements of $\mathcal{CT}$, i.e. :*

$$C \approx C' \Leftrightarrow \begin{cases} M = M' \text{ and} \\ |\chi| = |\chi'| \text{ and} \\ \forall x \in \chi, \exists x' \in \chi', trans(x) = trans(x') \end{cases}$$

It is then possible to define the inclusion of an extended state class into an other one.

Definition 18 *An extended state class $C' = \langle M', D', \chi', trans'\rangle$ is included in an extended state class $C = \langle M, D, \chi, trans\rangle$ (denoted$C' \subseteq C$) if and only if $C \approx C'$ and $[|D'|] \subseteq [|D|]$*

Remark There exists algorithms to decide whether $[|D'|] \subseteq [|D|]$ as noted in [4], [21].

3.3.2 Construction of the extended state class graph

Now that we have defined the extended state classes of a Transitions Time Well-formed Net, we will present the construction of its extended state class graph. In this section we consider a TTWN $\mathcal{T} = \langle P, T, W^-, W^+, Cl, \mathcal{C}, \Phi, (\alpha, \beta), m_0\rangle$. We suppose that the set of transitions can be indexed, such that $T = \{t_1, ..., t_n\}$, and for each transition $t_i \in T$, we index the colour domain such that $\mathcal{C}(t_i) = \{c_{i,1}, ..., c_{i,n_i}\}$ $(n_i = |\mathcal{C}(t_i)|)$.

The extended state class graph $\Delta'(\mathcal{T})$ associated to the TTWN $\mathcal{T}$ is a transition system $\Delta'(\mathcal{T}) = \langle \mathcal{EC}, cl_0, \mathcal{CT}, \rightarrow_{ext} \rangle$. The initial state class $cl_0 = \langle m_0, D_0, \{x_0\}, trans_0 \rangle$ is defined by :

- m_0 is the initial marking of $\mathcal{T}$;
- D_0 is such that :

$$\left\{ \alpha(t_i) \leq \theta_{i,j} \leq \beta(t_i), \forall (t_i, c_{i,j}) \in \mathcal{CT} \text{ such that } t_i \text{ is enabled for } c_{i,j} \text{ in } m_0 \right.$$

- The set of clocks χ_0 of cl_0 is composed from a single clock x_0;
- $trans_0$ is defined by $trans_0(x_0) = \{(t,c) \in \mathcal{CT} | t \text{ is enabled for } c \text{ in } m_0\}$.

To build the extended state class graph, we use a breadth first graph generation algorithm. Given an extended state class C, we want to compute its sons. To do this we need to know, which transitions are firable among the transitions that are enabled from the marking of C.

Definition 19 *A transition t_i is firable for a colour $c_{i,j} \in \mathcal{C}(t_i)$ from an extended state class $C = \langle M, D, \chi, trans \rangle \in \mathcal{EC}$, if and only if t_i is enabled for $c_{i,j}$ in M and the following system of inequalities :*

$$\begin{cases} D \\ \theta_{i,j} \leq \theta_{k,l}, \forall (k,l) \neq (i,j) \textit{ such that } t_k \textit{ is enabled for } c_{k,l} \textit{ in } M \end{cases}$$

has a solution.

The inequalities which are added to D ensure that the relative time of firing of the transition t_i for the colour $c_{i,j}$ cannot be bigger than the upper limits of firing of the other enabled transitions. This is the consequence of the fact that in a marking of a TTWN, the elapsing of time cannot disable transitions that are firable. In the extended state class graph, the classes are linked by edges labeled with transitions and their associated colours which are firable from the extended state class which is the source of the edge. The algorithm terminates when it encounters an extended state class for which the computation has already been done or when there is no more firable transitions. The algorithm to compute the extended state class graph can be written as follows :

- The variables are the graph $ESCG$ and the FIFO queue of extended state class New
- The variables are initialized to : $ESCG := \{cl_0\}$ and $New := cl_0$;
- WHILE New is NOT empty DO
  - $C := remove(New)$ (The first extended state class of New is taken, $C := \langle M_C, D_C, \chi_C, trans_C \rangle$);
  - For all transitions t_i firable from C for some colour $c_{i,j}$ DO
    - (Computation of the extended state class $C' = \langle M_{C'}, D_{C'}, \chi_{C'}, trans_{C'} \rangle$, which is the son of C by the firing of t_i for $c_{i,j}$);

* for all $p \in P$, $M_{C'}(p) = M_C(p) - W^-(p,t_i)(c_{i,j}) + W^+(p,t_i)(c_{i,j})$;
* Computation of $D_{C'}$, the firing domain of C' :
  1. In the system of inequalities :

     $$\begin{cases} D \\ \theta_{i,j} \leq \theta_{k,l}, \forall (k,l) \neq (i,j) \text{ such that } t_k \text{ is enabled for } c_{k,l} \text{ in } M \end{cases}$$

     Perform the change of variables :

     $$\begin{cases} \theta_{i,j} := \theta_F \\ \theta_{k,l} := \theta_F + \theta'_{k,l} \end{cases}$$

     A system of inequalities D' is obtained ;
  2. Eliminate from D' the variables θ_F (using for instance the method of Fourier-Motzkin) , to obtain a system of inequalitie D'';
  3. For all transitions t_k that were enabled for a colour $c_{k,l}$ from M_C and that are not enabled for $c_{k,l}$ from M' (where $\forall p \in P, M'(p) = M_C(p) - W^-(p,t_i)(c_{i,j})$), eliminate the variable $\theta'_{k,l}$ from D'' (using the same method as before), o obtain a system of inequalities $D^{(3)}$;
  4. For all transitions t_k that are newly enabled for a colour $c_{k,l}$ after the firing of t_i for $c_{i,j}$ from M_C (i.e. $\uparrow enabled((t_k,c_{k,l}),M_C,(t_i,c_{i,j})) = True$), add to $D^{(3)}$ the inequalities $\alpha(t_k) \leq \theta_{k,l} \leq \beta(t_k)$, to obtain a system of inequalities $D_{C'}$;
* Computation of $\chi_{C'}$ and $trans_{C'}$:
  1. For each clock $x \in \chi_C$, remove from $trans_C(x)$ all the pairs $(t_k,c_{k,l})$ such that t_k is enabled for $c_{k,l}$ from M_C and is not enabled for $c_{k,l}$ from M' (where $\forall p \in P, M'(p) = M_C(p) - W^-(p,t_i)(c_{i,j})$), to obtain a relation $trans'$;
  2. The clocks whose image by $trans'$ is empty are removed from χ_C, to obtain a set of clocks χ';
  3. For all transitions t_k that are newly enabled for $c_{k,l}$ after the firing of t_i for $c_{i,j}$ from M_C DO :
     + IF a clock x has already been created for computing C'
     + THEN $(t_k,c_{k,l})$ is added to $trans'(x)$;
     + ELSE a new clock x_n is created; n is the smallest available index among the clocks of χ' and $trans'(x_n) := (t_k,c_{k,l})$;
  4. ENNDO; $\chi_{C'}$ and $trans_{C'}$ are then obtain from the resulting set χ' and function $trans'$;
* IF there is an extended state class C'' in $ESCG$ such that C' and C'' are clock similar
* THEN
  + $ESCG = ESCG \cup \{C \overset{(t_i,c_{i,j})}{\rightarrow}_{ext} C''\}$;
  + IF $C' \not\subseteq C''$

```
        + THEN
            ++ $D_{C''} := D_{C''} \cup D_{C'}$;
            ++ IF $C'' \notin New$ (the sons of $C''$ were already computed) THEN compute the sons of $C'$ and add them to the sons of $C''$ ENDIF;
        + ENDIF;
      * ELSE $ESCG = ESCG \cup \{C \overset{(t_i,c_{i,j})}{\rightarrow}_{ext} C'\}$ and $add(New, C')$;
      * ENDIF;
    – ENDDO;
• ENDDO.
```

3.4 State class timed automaton

3.4.1 Construction of the state class timed automaton

In this section, we will give the definition of a state class timed automaton, the definition explains of which also the method used to build this timed automaton.

Definition 20 *Let $\mathcal{T} = \langle P, T, W^-, W^+, Cl, \mathcal{C}, \Phi, (\alpha, \beta), m_0 \rangle$ be a Transitions Time Well-formed Nets and $\Delta'(\mathcal{T}) = \langle \mathcal{EC}, cl_0, \mathcal{CT}, \rightarrow_{ext} \rangle$ its associated extended state class graph. The state class timed automaton $\Delta(\mathcal{T})$ associated to $\mathcal{T}$ is a timed automaton $\langle L, L^0, \Sigma, X, I, E \rangle$ defined by :*

- *$L = \mathcal{EC}$ is the set of the extended classes ;*
- *$L^0 = \{cl_0\}$ is the initial extended state class ($cl_0 = \langle m_0, D_0, \{x_0\}, trans_0 \rangle$);*
- *$X = \bigcup_{\langle M,D,\chi,trans \rangle \in \mathcal{EC}} \chi$;*
- *$\Sigma = \mathcal{CT} = \{(t,c) | t \in T \wedge c \in \mathcal{C}(t)\}$;*
- *E is the set of switches defined by :*

$$\forall C_i = \langle M_i, D_i, \chi_i, trans_i \rangle$$

$$\forall C_j = \langle M_j, D_j, \chi_j, trans_j \rangle$$

$$\exists C_i \overset{(t_i,c_{i,j})}{\rightarrow}_{ext} C_j \Leftrightarrow \exists (s_i, a, \phi, \lambda, \rho, s_j) \in E \textit{ such that}$$

$$\begin{cases} s_i = C_i \textit{ and} \\ s_j = C_j \textit{ and} \\ a = (t_i, c_{i,j}), \\ \phi = (trans_i^{-1}((t_i, c_{i,j})) \geq \alpha(t_i)) \textit{ and} \\ \lambda = \{trans_j^{-1}((t_k, c_{k,l})) | \uparrow enabled((t_k, c_{k,l}), M_i, (t_i, c_{i,j})) = True\} \textit{ and} \\ \forall x \in \chi_i, \forall x' \in \chi_j, \textit{ such that } trans_j(x') \subseteq trans_i(x) \textit{ and } x' \notin \lambda, \rho(x') = x \end{cases}$$

- *$\forall C_i = \langle M_i, D_i, \chi_i, trans_i \rangle \in \mathcal{EC}$, $I(C_i) = \bigwedge_{x \in \chi_i, (t,c) \in trans_i(x)} (x \leq \beta(t))$*

3.4.2 Properties of the state class timed automaton

Bisimulation

In this part, we will define a binary relation between the states of a TTWN $\mathcal{T}$ and the states of its associated state class timed automaton and we will prove that this relation is a bisimulation.

Definition 21 *Let $\mathcal{T} = \langle P,T,W^-,W^+,Cl,\mathcal{C},\Phi,(\alpha,\beta),m_0\rangle$ be a Transitions Time Well-formed Nets and $\Delta(\mathcal{T})$ the associated state class timed automaton. We consider $\bar{Q}_{\mathcal{T}} = \mathcal{MA}\times\mathcal{V}$ the set of reachable states of $\mathcal{T}$ and Q_A the set of states of $\Delta(\mathcal{T})$. We define the relation $\simeq_{sc}\subseteq \bar{Q}_{\mathcal{T}}\times Q_A$ by, $\forall s=(M,\nu_{\mathcal{T}})\in\bar{Q}_{\mathcal{T}}$, $\forall a=(cl,\nu_A)\in Q_A$ (with $cl = \langle M_a,D_a,\chi_a,trans_a\rangle$):*

$$s \simeq_{sc} a \Leftrightarrow \begin{cases} M = M_a \\ \\ \forall (t,c)\in\mathcal{CT} \text{ such that } t \text{ is enabled for } c \text{ from } M, \\ \nu_{\mathcal{T}}((t,c)) = \nu_A(x) \text{ with } x\in\chi_a \text{ such that } (t,c)\in trans_a(x)\end{cases}$$

Once this relation defined, we have the following theorems.

Theorem 1 *For all $(s,a)\in\bar{Q}_{\mathcal{T}}\times Q_A$, if $s\simeq_{sc} a$ then :*

1. *$\forall\delta\in\mathbb{R}_{\geq 0}$, if $\exists s'\in\bar{Q}_{\mathcal{T}}$ such that $s\xrightarrow{\delta}s'$, then $\exists a'\in Q_A$ such that $a\xrightarrow{\delta}a'$ and $s'\simeq_{sc}a'$;*
2. *$\forall (t,c)\in\mathcal{CT}$, if $\exists s'\in\bar{Q}_{\mathcal{T}}$ such that $s\xrightarrow{(t,c)}s'$, then $\exists a'\in Q_A$ such that $a\xrightarrow{(t,c)}a'$ and $s'\simeq_{sc}a'$.*

Proof Let $s=(M,\nu_{\mathcal{T}})$ be a state in $\bar{Q}_{\mathcal{T}}$ and $a=(cl,\nu_A)$ with $cl=\langle M_a,D_a,\chi_a,trans_a\rangle$ be a state in Q_A such that $s\simeq_{sc}a$.
We consider $\delta\in\mathbb{R}_{\geq 0}$ and we suppose that there exists $s'\in\bar{Q}_{\mathcal{T}}$ such that $s\xrightarrow{\delta}s'$. Then $s'=(M,\nu'_{\mathcal{T}})$ with $\nu'_{\mathcal{T}}=\nu_{\mathcal{T}}+\delta$. For all $(t,c)\in\mathcal{CT}$, such that t is enabled for c from M,we have then (by definition of continuous transition), $\nu_{\mathcal{T}}((t,c))+\delta\leq\beta(t)$. Since $s\simeq_{sc}a$,we can deduce that for all (t,c) such that t is enabled for c from M, for $x=trans_a^{-1}((t,c))$ we have $\nu_A(x)+\delta\leq\beta(t)$. For all $x\in\chi_a$, if $(t,c)\in trans_a(x)$, by definition of $trans_a(x)$, t is enabled for c from $M_a=M$, then we have $\bigwedge_{x\in\chi_a,(t,c)\in trans_a(x)}(\nu_A(x)+\delta)\leq\beta(t))$, which means that $\nu_A+\delta$ satisfies $I(cl)=\bigwedge_{x\in\chi_a,(t,c)\in trans_a(x)}(x\leq\beta(t))$. We deduce that $a\xrightarrow{\delta}a'$ with $a'=(cl,\nu'_A)$ (where$\nu'_A=\nu_A+\delta$).
The markings that appear in a' and s' are equal because, the extended state class of a' is the same as the one of a, the marking of s' is the same as the marking of s and the marking of a and s are equal. We consider $(t,c)\in\mathcal{CT}$ such that t is enabled for c from M,we denote x the clock in χ_a such that $x\in trans_a^{-1}((t,c))$, then $\nu_{\mathcal{T}}((t,c))=\nu_A(x)$, we can deduce that $\nu_{\mathcal{T}}((t,c))+\delta=\nu_A(x)+\delta$. Since $\nu'_{\mathcal{T}}=\nu_{\mathcal{T}}+\delta$ and $\nu'_A=\nu_A+\delta$, since the extended state class is the same in a and in a', we deduce that $\forall(t,c)\in\mathcal{CT}$ such that t is enabled for c from M, $\nu'_{\mathcal{T}}((t,c))=\nu'_A(x)$. We conclude that $s'\simeq_{sc}a'$.

We consider $(t,c)\in\mathcal{CT}$ and we suppose that there exists $s'=(M',\nu'_{\mathcal{T}})\in\bar{Q}_{\mathcal{T}}$ such that

$s \overset{(t,c)}{\rightarrow} s'$. To prove that there exists $a' \in Q_A$ such that $a \overset{(t,c)}{\rightarrow} a'$, we will follow the following steps :

1. We will prove that t is firable for c from the extended state class cl. This will allow us to say that there is an edge in the extended state class graph of the form $cl \overset{(t,c)}{\rightarrow}_{ext} cl'$; hence that there exists a switch in the timed automaton of the form $(cl,(t,c),\phi,\lambda,\rho,cl')$

2. We will show that ν_A satisifies the guard condition ϕ of this switch;

3. We will conclude that $a \overset{(t,c)}{\rightarrow} a'$ with $a' = (cl', \nu'_A)$, $cl' = \langle M'_a, D'_a, \chi'_a, trans'_a\rangle$ and $M'_a = M'$.

We will use the following lemma :

Lemma 1 $\forall (t,c) \in \mathcal{CT}$, $\forall s,s' \in \bar{Q}_{\mathcal{T}}$, if $s \overset{(t,c)}{\rightarrow} s'$, then $\forall a = (cl,\nu_A) \in Q_A$ such that $s \simeq_{sc} a$, t is firable for c from cl.

We consider the three points in turn. (1.) This point is proven by Lemma 1.
(2.) Since we have $s \overset{(t,c)}{\rightarrow} s'$, we deduce that $\nu_{\mathcal{T}}((t,c)) \geq \alpha(t)$. Let $\{x\} = trans_a^{-1}((t,c))$. Since $s \simeq_{sc} a$, we have $\nu_A(x) \geq \alpha(t)$. Then because the guard ϕ on the switch that was found in part (1.) is by construction $x \geq \alpha(t)$ we conclude that ν_A satisfies ϕ.
(3.) Parts (1.) and (2.) allow us to conclude that there exists $a' = (cl', \nu'_A) \in Q_A$ such that $a \overset{(t,c)}{\rightarrow} a'$ and $cl' = \langle M'_a, D'_a, \chi'_a, trans'_a\rangle$ with, by construction of cl', for all $p \in P$: $M'_a(p) = M_a(p) - W^-(p,t)(c) + W^+(p,t)(c) = M(p) - W^-(p,t)(c) + W^+(p,t)(c) = M'(p)$.
Now we want to prove that $\forall (t',c') \in \mathcal{CT}$ such that t' is enabled for c' from M', for $x' \in \chi'_a$ such that $\{x'\} = trans_a'^{-1}((t',c'))$ we have $\nu'_{\mathcal{T}}((t',c')) = \nu'_A(x')$. We consider a pair (t',c') in $\mathcal{CT}$ such that t' is enabled for c' from M'. Two cases are possible :

1. $\uparrow enabled((t',c'),M,(t,c)) = False$. Then it means that t' is enabled for c' from M, and we deduce that for x such that $(t',c') \in trans_a(x)$, we have $\nu_{\mathcal{T}}((t',c')) = \nu_A(x)$. By the definition of $s \overset{(t,c)}{\rightarrow} s'$, we have $\nu'_{\mathcal{T}}((t',c')) = \nu_{\mathcal{T}}((t',c'))$. We denote $\{x'\}$ $=trans_a'^{-1}((t',c'))$. By construction of the state class automaton, since t' is not newly enabled for c' we deduce that $trans'_a(x') \subseteq trans_a(x)$ (in fact, during the construction of the extended state class graph, when a set $trans(x)$ that contains enabled transtions which are not newly enabled, is built, no pair (t,c) is added to this set; only removal of pairs is performed, and here (t',c') is not removed because the associated transition is not disabled by the firing of t for c). We deduce that the renaming function ρ of the switch is such that $\rho(x') = x$, and hence we have, by construction of $a \overset{(t,c)}{\rightarrow} a'$, $\nu'_A(x') = \nu_A(\rho(x')) = \nu_A(x)$. We conclude that $\nu'_A(x') = \nu'_{\mathcal{T}}((t',c'))$.

2. $\uparrow enabled((t',c'),M,(t,c)) = True$. We have by defnition of $s \overset{(t,c)}{\rightarrow} s'$, $\nu'_{\mathcal{T}}((t',c')) = 0$. By construction of the extended state class, a new clock x' has been created for cl' such that $(t',c') \in trans'_a(x')$ and xe have $x \in \lambda$. Hence $\nu'_A(x') = 0 = \nu'_{\mathcal{T}}((t',c'))$.

We conclude that $s' \simeq_{sc} a'$.
□

Proof of Lemma 1
We consider $(t,c) \in \mathcal{CT}$ and $s,s' \in \bar{Q}_{\mathcal{T}}$ such that $s \overset{(t,c)}{\to} s'$. We suppose that there exists $a = (cl, \nu_A) \in Q_A$ such that $s \simeq_{sc} a$ and such that t is not firable for c from cl. Let M be the marking of s. Because we have $s \simeq_{sc} a$, M is also the marking associated to cl. Since we have $s \overset{(t,c)}{\to} s'$ for the TTWN $\mathcal{T}$, we deduce that there is a path in the associated transtion system of the form :

$$(m_0, \bar{0}) \overset{\delta_0}{\to} (m_0, v'_0) \overset{(t_0,c_0)}{\to} (m_1, v_1) \overset{\delta_1}{\to} (m_1, v'_1) \overset{(t_1,c_1)}{\to} (m_2, v_2) ... \overset{\delta_n}{\to} (M, v)$$

Since the extended state class graph represents by construction the behavior of the system, we deduce that there is in this graph a path :

$$cl_0 \overset{(t_0,c_0)}{\to} cl_1 ... \overset{(t_{n-1},c_{n-1})}{\to} cl'$$

where the marking of cl' is M and where t is firable for c from cl'. Since t is not firable from c from cl, we deduce that cl and cl' are different, and hence cl and cl' are not clock similar. We denote $cl = \langle M, D, \chi, trans \rangle$ and $cl' = \langle M', D', \chi', trans' \rangle$. For an extended class, the set of clocks and the relation $trans$ realize a partition of the set $\{(t,c) | t \text{ is enabled for } c \text{ from } M\}$ and group together the transitions that have been newly enabled for the same firing and from the same marking. For cl and cl', this partition is not the same because these two extended state classes are not clock similar. This partition also groups the pairs (t,c) that are linked to the same clock of x. Since the transitions which have been newly enabled for a colour from the same markings and which are enabled from M have the same clock values for ν, and since $s \simeq_{sc} a$, we can deduce that the partition of cl of the set $\{(t,c) | t \text{ is enabled for } c \text{ from } M\}$ groups transitions that have been newly enabled for a colour from the same markings in the path :

$$(m_0, \bar{0}) \overset{\delta_0}{\to} (m_0, v'_0) \overset{(t_0,c_0)}{\to} (m_1, v_1) \overset{\delta_1}{\to} (m_1, v'_1) \overset{(t_1,c_1)}{\to} (m_2, v_2) ... \overset{\delta_n}{\to} (M, v)$$

which means that the partition is the same as the one cl'. Hence cl and cl' are clock-similar, which is a contradiction.
□

Theorem 2 *For all $(s,a) \in \bar{Q}_{\mathcal{T}} \times Q_A$, if $s \simeq_{sc} a$ then :*

1. *$\forall \delta \in \mathbb{R}_{\geq 0}$, if $\exists a' \in Q_A$ such that $a \overset{\delta}{\to} a'$, then $\exists s' \in \bar{Q}_{\mathcal{T}}$ such that $s \overset{\delta}{\to} s'$ and $s' \simeq_{sc} a'$;*
2. *$\forall (t,c) \in \mathcal{CT}$, if $\exists a' \in Q_A$ such that $a \overset{(t,c)}{\to} a'$, then $\exists s' \in \bar{Q}_{\mathcal{T}}$ such that $s \overset{(t,c)}{\to} s'$ and $s' \simeq_{sc} a'$.*

This second theorem can be proved the same way as the first theorem was proved. From these two theorems, we can deduce a third theorem.

Theorem 3 *The binary relation $\simeq_{sc} \subseteq \bar{Q}_{\mathcal{T}} \times Q_A$ is a bisimulation.*

If we consider a TTWN $\mathcal{T} = \langle P, T, W^-, W^+, Cl, \mathcal{C}, \Phi, (\alpha, \beta), m_0 \rangle$ and its associated state class timed automaton $\Delta(\mathcal{T})$, since we have by construction $(m_0, \bar{0}) \simeq_{sc} (cl_0, \bar{0})$, we can conclude that a marking m is reachable from m_0 in $\mathcal{T}$ if and only if there exists a state of $\Delta(\mathcal{T})$ whose associated marking is m. Furthermore by finding all the states of $\Delta(\mathcal{T})$ whose markings are m and the sequences that lead to them, we can have all the sequences in $\mathcal{T}$ that lead to m, which will allow us to verify real-time properties of $\mathcal{T}$ by verifying them on $\Delta(\mathcal{T})$. This is useful because there already exist tools to check real-time properties on timed automata, such as Kronos [24] and Uppaal [3].

Boundedness

To be able to construct the state class timed automaton of a TTWN, we need the number of extended state classes to be bounded, because if this number is not bounded, the construction of the extended state class graph will never end, and consequently it will not be possible to transform it into a timed automaton. In this part we present sufficient conditions for a TTWN to have a bounded number of extended state classes. We will denote $\mathcal{R}(m_0)$ the set of markings which a TTWN can reach from its initial marking m_0. First we introduce a new definition.

Definition 22 *A TTWN $\mathcal{T} = \langle P, T, W^-, W^+, Cl, \mathcal{C}, \Phi, (\alpha, \beta), m_0 \rangle$ is bounded if and only if :*

$$(\exists k \in \mathbb{N})(\forall m \in \mathcal{R}(m_0))(\forall p \in P)(\forall c \in \mathcal{C}(p))(m(p)(c) \leq k)$$

Then we have the following theorem.

Theorem 4 *A TTWN has a finite number of extended state classes if and only if it is bounded.*

Proof The proof for this theorem is similar to the one described in [4] . In fact, in this article the lemmas that are used to prove the equivalent theorem for the Transitions Time Petri Nets are deduced from the form of the firing domains and from the operations over these firing domains. Since the firing domains have the same form and the operations over these firing domains are the same for the TTWN, the theorem can be extended to the TTWN.
□

Remark The only point that changes from the article [4] is the used of the definition of T-boundedness.The notion of T-boundedness was used in [4] because the semantics was slightly different, since if a transition was "simultaneously" enabled more than once, two times for instance, then there were two clock valuations.However inour model if a transition is enabled more than once for the same colour "simultaneously", we take account only of one firing, and once it has been done the transition becomes newly enabled for the colour (single-server semantics). Hence we do not need to make use of the concept of T-boundedness.

The previous theorem does not help to solve the problem of finiteness of the extended state class graph because of the following theorem.

Theorem 5 *The reachability and boundedness problems for TTWN's are undecidable.*

Proof It is possible to transform a Transitions Time Petri Net into a TTWN (for instance by taking only one colour class with one element and labeling all the arcs with $< X >$). Since reachability and boundedness are undecidable for the Transitions Time Petri Nets (Theorem 1 in [4]), we conclude that these two problems are also undecidable for the TTWN.
□

Even if we cannot have results on the full class of TTWN to determine if they are bounded, the two following theorems give some sufficient conditions for the boundedness of a TTWN.

Theorem 6 ***(Sufficient Condition 1)*** *A TTWN is bounded if no pair of extended state classes* $Cl = \langle m, D, \chi, trans \rangle$ *and* $Cl' = \langle m', D', \chi', trans' \rangle$ *are reachable from its initial extended state class and are such that :*

(i) Cl' *is reachable from* Cl*;*

(ii) $\forall p \in P$, $m'(p) \geq m(p)$ *and* $\exists p \in P$, $\exists c \in \mathcal{C}(p)$ *such that* $m'(p)(c) > m(p)(c)$.

Theorem 7 ***(Sufficient Condition 2)*** *A TTWN is bounded if no pair of extended state classes* $Cl = \langle m, D, \chi, trans \rangle$ *and* $Cl' = \langle m', D', \chi', trans' \rangle$ *are reachable from its initial extended state class and are such that :*

(i) Cl' *is reachable from* Cl*;*

(ii) $\forall p \in P$, $m'(p) \geq m(p)$ *and* $\exists p \in P$, $\exists c \in \mathcal{C}(p)$ *such that* $m'(p)(c) > m(p)(c)$ *;*

(iii) $[|D|] = [|D'|]$*;*

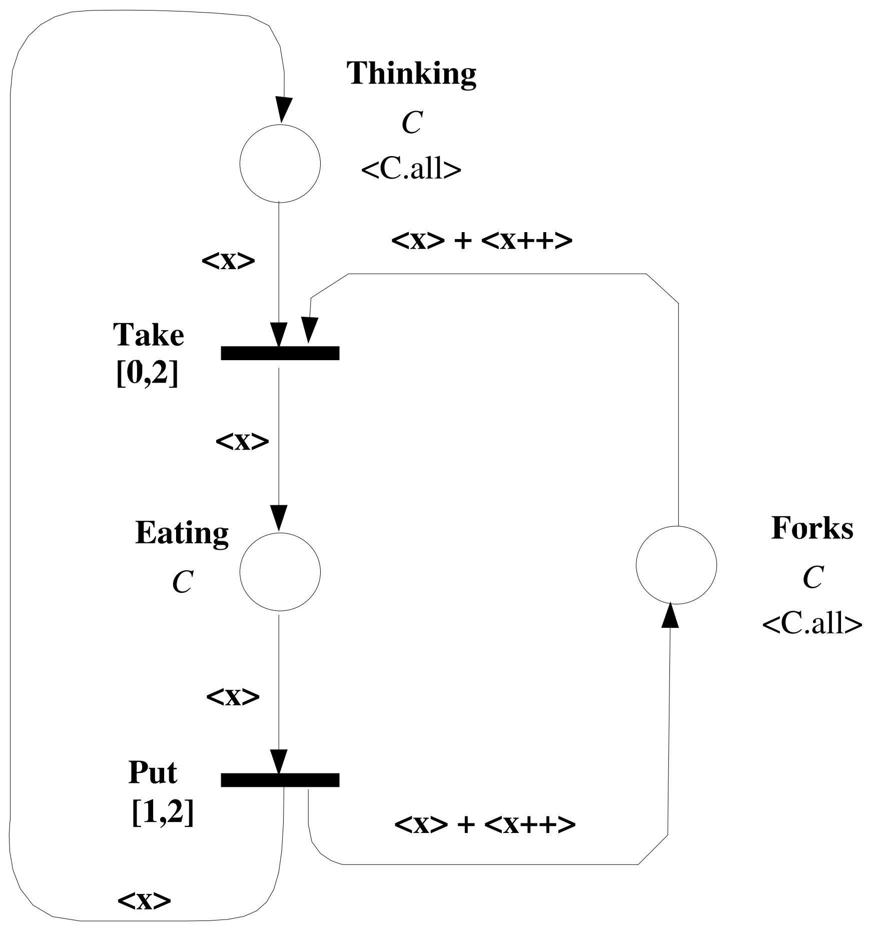
(iv) $\forall (p,c) \in \{(p,c) | p \in P \wedge c \in \mathcal{C}(p) \wedge m'(p)(c) > m(p)(c)\}$,
$m(p)(c) > Max_{(t,c') \in \mathcal{C}T} W^-(p,t)(c')(c)$.

These two theorems are also directly deduced from the equivalent theorems which appear in [4] for the Transitions Time Petri Nets. The proofs are identical to those that could be found in that article, which are based on construction of unbounded sequences using the results of the article [19] .

These two last theorems can be used in the algorithm to halt execution it when a pair of extended state classes that verifies the four conditions of the second sufficient condition will be encountered. It is true that doing so the algorithm might stop for TTWN that are bounded but since the conditions are quite strict, it will work for a large set of bounded TTWN. But since the condition $[(iii)]$ and $[(iv)]$ cost a lot to verify, sometimes the first sufficient condition might be chosen for the algorithm even if it is effective for fewer TTWN.

3.5 Example

In this section, we extend the well-known problem of the dining philosophers with time constraints. In fact, we have supposed that when its forks are enabled, a philosopher will take them in at most the next two time units, if the forks stay enabled during this time. We have also supposed that each philosopher takes at least one time unit and at most two time units to eat. Figure 3.1 gives the TTWN associated to this model. Figure 3.2 gives the associated state class timed automaton (in this timed automaton, T represents the transition Take and P represents the transition Put).



Class	Var
C is 1..4;	x in C

Figure 3.1: The philosophers problem with time constraints

The extended state classes that feature in the state class timed automaton represented by the figure 3.2 are defined by :

- $Cl_0 = \langle m_0, D_0, \chi_0, trans_0 \rangle$ with :

  1. $m_0(Thinking) = <1> + <2> + <3> + <4>$, $m_0(Forks) = <1> + <2> + <3> + <4>$ and $m_0(Eating) = \emptyset$;
  2. D_0 is defined by :
  $$\begin{cases} 0 \leq \theta_{T,<1>} \leq 2 \\ 0 \leq \theta_{T,<2>} \leq 2 \\ 0 \leq \theta_{T,<3>} \leq 2 \\ 0 \leq \theta_{T,<4>} \leq 2 \end{cases}$$
  3. $\chi_0 = \{x_0\}$;
  4. $trans_0(x_0) = \{(T, <1>), (T, <2>), (T, <3>), (T, <4>)\}$.

- $Cl_1 = \langle m_1, D_1, \chi_1, trans_1 \rangle$ with :

  1. $m_1(Thinking) = <2> + <3> + <4>$, $m_1(Forks) = <3> + <4>$ and $m_1(Eating) = <1>$;
  2. D_1 is defined by :
  $$\begin{cases} 0 \leq \theta_{T,<3>} \leq 2 \\ 0 \leq \theta_{P,<1>} \leq 2 \end{cases}$$
  3. $\chi_1 = \{x_0, x_1\}$;
  4. $trans_1(x_0) = \{(T, <3>)\}$ and $trans_1(x_1) = \{(P, <1>)\}$.

- $Cl_2 = \langle m_2, D_2, \chi_2, trans_2 \rangle$ with :

  1. $m_2(Thinking) = <1> + <3> + <4>$, $m_2(Forks) = <1> + <4>$ and $m_2(Eating) = <2>$;
  2. D_2 is defined by :
  $$\begin{cases} 0 \leq \theta_{T,<4>} \leq 2 \\ 0 \leq \theta_{P,<2>} \leq 2 \end{cases}$$
  3. $\chi_2 = \{x_0, x_1\}$;
  4. $trans_2(x_0) = \{(T, <4>)\}$ and $trans_2(x_1) = \{(P, <2>)\}$.

- $Cl_3 = \langle m_3, D_3, \chi_3, trans_3 \rangle$ with :

  1. $m_3(Thinking) = <1> + <2> + <4>$, $m_3(Forks) = <1> + <2>$ and $m_3(Eating) = <3>$;
  2. D_3 is defined by :
  $$\begin{cases} 0 \leq \theta_{T,<1>} \leq 2 \\ 0 \leq \theta_{P,<3>} \leq 2 \end{cases}$$
  3. $\chi_3 = \{x_0, x_1\}$;
  4. $trans_3(x_0) = \{(T, <1>)\}$ and $trans_3(x_1) = \{(P, <3>)\}$.

- $Cl_4 = \langle m_4, D_4, \chi_4, trans_4 \rangle$ with :

  1. $m_4(Thinking) =< 1 > + < 2 > + < 3 >$, $m_4(Forks) =< 2 > + < 3 >$ and $m_4(Eating) =< 4 >$;
  2. D_4 is defined by :
  $$\begin{cases} 0 \leq \theta_{T,<2>} \leq 2 \\ 0 \leq \theta_{P,<4>} \leq 2 \end{cases}$$
  3. $\chi_4 = \{x_0, x_1\}$;
  4. $trans_4(x_0) = \{(T, < 2 >)\}$ and $trans_4(x_1) = \{(P, < 4 >)\}$.

- $Cl_5 = \langle m_5, D_5, \chi_5, trans_5 \rangle$ with :

  1. $m_5(Thinking) =< 1 > + < 2 > + < 3 > + < 4 >$, $m_5(Forks) =< 1 > + < 2 > + < 3 > + < 4 >$ and $m_5(Eating) = \emptyset$;
  2. D_5 is defined by :
  $$\begin{cases} 0 \leq \theta_{T,<1>} \leq 2 \\ 0 \leq \theta_{T,<2>} \leq 2 \\ 0 \leq \theta_{T,<3>} \leq 2 \\ 0 \leq \theta_{T,<4>} \leq 2 \end{cases}$$
  3. $\chi_5 = \{x_0, x_1\}$;
  4. $trans_5(x_0) = \{(T, < 3 >)\}$ and $trans_5(x_1) = \{(T, < 1 >), (T, < 2 >), (T, < 4 >)\}$.

- $Cl_6 = \langle m_6, D_6, \chi_6, trans_6 \rangle$ with :

  1. $m_6(Thinking) =< 2 > + < 4 >$, $m_6(Forks) = \emptyset$ and $m_6(Eating) =< 1 > + < 3 >$;
  2. D_6 is defined by :
  $$\begin{cases} 0 \leq \theta_{P,<1>} \leq 2 \\ 0 \leq \theta_{P,<3>} \leq 2 \end{cases}$$
  3. $\chi_6 = \{x_0, x_1\}$;
  4. $trans_6(x_0) = \{(P, < 3 >)\}$ and $trans_6(x_1) = \{(P, < 1 >)\}$.

- $Cl_7 = \langle m_7, D_7, \chi_7, trans_7 \rangle$ with :

  1. $m_7(Thinking) =< 1 > + < 2 > + < 3 > + < 4 >$, $m_7(Forks) =< 1 > + < 2 > + < 3 > + < 4 >$ and $m_7(Eating) = \emptyset$;
  2. D_7 is defined by :
  $$\begin{cases} 0 \leq \theta_{T,<1>} \leq 2 \\ 0 \leq \theta_{T,<2>} \leq 2 \\ 0 \leq \theta_{T,<3>} \leq 2 \\ 0 \leq \theta_{T,<4>} \leq 2 \end{cases}$$

3. $\chi_7 = \{x_0, x_1\}$;
4. $trans_7(x_0) = \{(T, <4>)\}$ and $trans_7(x_1) = \{(T, <1>), (T, <2>), (T, <4>)\}$.

- $Cl_8 = \langle m_8, D_8, \chi_8, trans_8 \rangle$ with :

1. $m_8(Thinking) = <1> + <3>$, $m_8(Forks) = \emptyset$ and $m_8(Eating) = <2> + <4>$;
2. D_8 is defined by :

$$\begin{cases} 0 \leq \theta_{P,<2>} \leq 2 \\ 0 \leq \theta_{P,<4>} \leq 2 \end{cases}$$

3. $\chi_8 = \{x_0, x_1\}$;
4. $trans_8(x_0) = \{(P, <4>)\}$ and $trans_8(x_1) = \{(P, <2>)\}$.

- $Cl_9 = \langle m_9, D_9, \chi_9, trans_9 \rangle$ with :

1. $m_9(Thinking) = <1> + <2> + <3> + <4>$, $m_9(Forks) = <1> + <2> + <3> + <4>$ and $m_9(Eating) = \emptyset$;
2. D_9 is defined by :

$$\begin{cases} 0 \leq \theta_{T,<1>} \leq 2 \\ 0 \leq \theta_{T,<2>} \leq 2 \\ 0 \leq \theta_{T,<3>} \leq 2 \\ 0 \leq \theta_{T,<4>} \leq 2 \end{cases}$$

3. $\chi_9 = \{x_0, x_1\}$;
4. $trans_9(x_0) = \{(T, <1>)\}$ and $trans_9(x_1) = \{(T, <2>), (T, <3>), (T, <4>)\}$.

- $Cl_{10} = \langle m_{10}, D_{10}, \chi_{10}, trans_{10} \rangle$ with :

1. $m_{10}(Thinking) = <1> + <2> + <3> + <4>$, $m_{10}(Forks) = <1> + <2> + <3> + <4>$ and $m_{10}(Eating) = \emptyset$;
2. D_{10} is defined by :

$$\begin{cases} 0 \leq \theta_{T,<1>} \leq 2 \\ 0 \leq \theta_{T,<2>} \leq 2 \\ 0 \leq \theta_{T,<3>} \leq 2 \\ 0 \leq \theta_{T,<4>} \leq 2 \end{cases}$$

3. $\chi_{10} = \{x_0, x_1\}$;
4. $trans_{10}(x_0) = \{(T, <2>)\}$ and $trans_{10}(x_1) = \{(T, <1>), (T, <3>), (T, <4>)\}$.

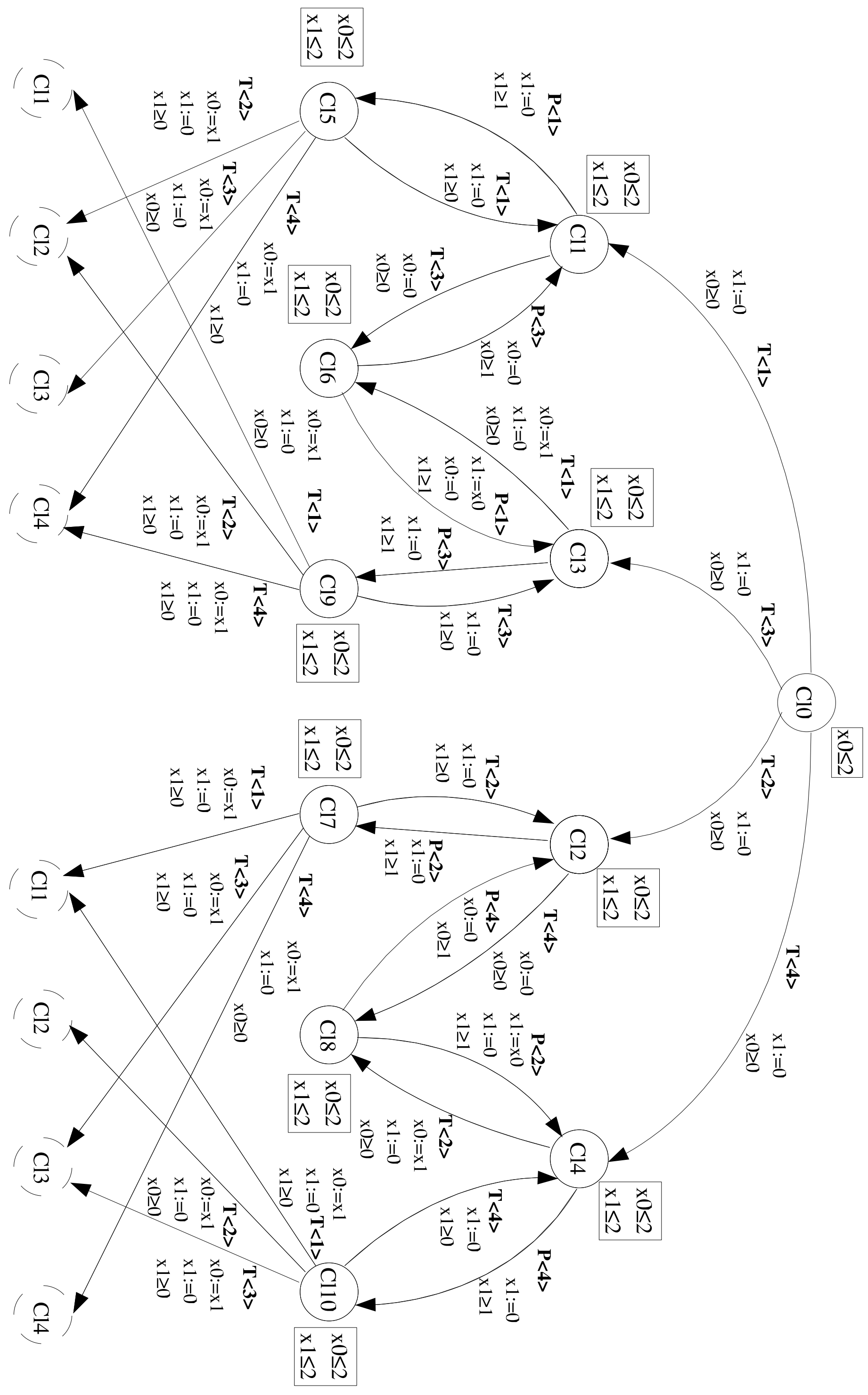


Figure 3.2: The associated state class timed automaton

Chapter 4

Marking class Timed Automaton of the Transitions Time Well-formed Nets

4.1 Introduction

The strucuture of the state class timed automaton of a Transitions Time Well-formed Net (TTWN), which we have presented in chapter 3, is quite similar to the one of the ordinary reachability graph of the underlying Well-formed net (i.e. the Well-formed net obtained by removing the time constraints). In this chapter, we present another timed automaton which has the same behavior as the TTWN and which is built from the ordinary reachability graph. We first present this method for Transitions Time Petri Net (TTPN) and then we extend it to the TTWN. We conclude the chapter by explaining in which cases this timed automaton can be built and what are the differences with the state class timed automaton.

4.2 Reachability Graph of the untimed Net

In this chapter we will consider a TTPN $\mathcal{T} = \langle P, T, W^-, W^+, m_0, (\alpha, \beta)\rangle$ where :

- $P = \{p_1, ..., p_m\}$ is the finite set of places;
- $T = \{t_1, ..., t_n\}$ is the finite set of transitions such that $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$;
- $W^- \in (\mathbb{N}^P)^T$ is the backward incidence mapping ;
- $W^+ \in (\mathbb{N}^P)^T$ is the forward incidence mapping ;
- $m_0 \in \mathbb{N}^P$ represents the initial marking ;
- $\alpha \in (\mathbb{Q}_{\geq 0})^T$ and $\beta \in (\mathbb{Q}_{\geq 0} \cup \{\infty\})^T$ are the earliest and latest firing time mappings.

The semantics of such a TTPN is described in chapter 1 . We denote by $S_{\mathcal{T}}$ the timed transition system which characterizes the behavior of $\mathcal{T}$. We will “untime” this TTPN in order to obtain a Place/Transition Petri Net $\mathcal{P} = \langle P, T, W^-, W^+, m_0\rangle$. We denote by $\mathcal{R}_{un}(m_0)$ the set of markings that $\mathcal{P}$ can reach from its initial marking m_0 (hence $\mathcal{R}_{un}(m_0) \subseteq \mathbb{N}^P$). When this Petri Net is bounded (i.e. $(\exists k \in \mathbb{N})(\forall p \in P)(\forall m \in \mathcal{R}_{un}(m_0))(m(p) \leq k)$), the behavior of this Petri Net can be represented by a finite-state transition sytem $S_{\mathcal{T}} = \langle Q, q_0, T, \rightarrow\rangle$ where :

- $Q = \mathcal{R}_{un}(m_0)$;
- $q_0 = m_0$;
- the transition relation $\rightarrow$ is defined by, $\forall M, M' \in \mathcal{R}_{un}(m_0), \forall t \in T$:

$$M \xrightarrow{t} M' \Leftrightarrow \begin{cases} M \geq W^-(t) \text{ and} \\ M' = M + W^+(t) - W^-(t) \end{cases}$$

This transition sytem is called the reachability graph of the Petri Net. From this reachability graph, we build a timed automaton which will have the same behavior as the TTPN.

4.3 Marking class timed automaton of a TTPN

4.3.1 Marking classes of a TTPN

In this section we will define the marking classes of a TTPN. A marking class corresponds to a marking of the underlying Petri Net and a set of clocks associated to the enabled transitions for the marking.

Definition 23 *A marking class of a TTPN $\mathcal{T} = \langle P, T, W^-, W^+, m_0, (\alpha, \beta)\rangle$ is a tuple $\langle M, \chi, trans\rangle$ where M is a marking belonging to $\mathcal{R}_{un}(m_0)$, χ is a set of real valued clocks and $trans \in (2^T)^\chi$ maps clocks to sets of transitions.*
We will denote $\mathcal{MC}$ the set of the marking classes of a Transitions Time Petri Net.

The relation $trans$ associates to each clock $x \in \chi$ a set of transitions such that $\forall t \in trans(x)$, x represents the value linked to t. Each transition $t \in T$ must be associated to at most one clock (i.e. $\forall t \in T$, $|trans^{-1}(t)| \leq 1$).

We also need the notion of clock-similarity for the marking classes.

Definition 24 *Two marking classes $Mc = \langle M, \chi, trans\rangle$ and $Mc' = \langle M', \chi', trans'\rangle$ are clock-similar, denoted $Mc \approx Mc'$ if and only if they have the same marking, the same number of clocks and their clocks are mapped to the same transitions, i.e. :*

$$Mc \approx Mc' \Leftrightarrow \begin{cases} M = M' \textit{ and} \\ |\chi| = |\chi'| \textit{ and} \\ \forall x \in \chi, \exists x' \in \chi', trans(x) = trans'(x') \end{cases}$$

4.3.2 Construction of the marking class graph

In this part, we give the algorithm which allows the building of the marking class graph $\Gamma'(\mathcal{T})$ of a TTPN $\mathcal{T}$, which is a transition system $\Gamma'(\mathcal{T}) = \langle \mathcal{MC}, Mc_0, T, \rightarrow_{mc}\rangle$. The initial marking class $Mc_0 = \langle m_0, \{x_0\}, trans_0\rangle$ is defined by :

- m_0 is the initial marking of $\mathcal{T}$;
- The set of clocks χ_0 of Mc_0 is composed of a single clock x_0;

- $trans_0$ is defined by $trans_0(x_0) = \{t \in T \mid t \text{ is enabled for } m_0\}$.

To build the marking class graph, we use a breadth-first graph generation algorithm. This algorithm begins by computing the sons of the initial marking class, then it proceeds by progressively computing the sons of each computed marking class. The algorithm terminates when it cannot compute more new marking classes. This algorithm can be written as follows :

```
• The variables are $MSG$ (the marking state graph itself) and a FIFO queue $New$
  (which contains the newly computed marking classes);
• The variables are initialized to : $MSG := \{Mc_0\}$ and $New := Mc_0$;
• WHILE $New$ is NOT empty DO
  – $C := remove(New)$ (The first marking class of $New$ is taken, $C := \langle M_C, \chi_C, trans_C \rangle$);
  – For all transitions $t$ enabled for $M_C$ (i.e. $M_C \geq W^-(t)$) DO
    ∗ (Computation of the marking class $C' = \langle M_{C'}, \chi_{C'}, trans_{C'} \rangle$, son of $C$ )
    ∗ $M_{C'} = M_C + W^+(t) - W^-(t)$;
    ∗ Computation of $\chi_{C'}$ and $trans_{C'}$ :
      1. For each clock $x \in \chi_C$, remove from $trans_C(x)$ all the transitions $t_k$
         such that $t_k$ is enabled from $M_C$ and is not from $M_C - W^-(t)$, to obtain
         a relation $trans'$;
      2. The clocks whose image by $trans'$ is empty are removed from $\chi_C$, to
         obtain a set of clocks $\chi'$;
      3. For all transitions $t_k$ which verify $\uparrow enabled(t_k, M_C, t) = True$ DO :
         + IF a clock $x$ has already been created for the computation of $C'$
         + THEN $t_k$ is added to $trans'(x)$;
         + ELSE a new clock $x_n$ is created; $n$ is the smallest available index
           among the clocks of $\chi'$ and $trans'(x_n) = t_k$;
      4. ENDDO; $\chi_{C'}$ and $trans_{C'}$ are then obtained from the resulting set $\chi'$
         and function $trans'$;
    ∗ IF there is a marking class $C''$ in $MSG$ such that $C' \approx C''$
    ∗ THEN $MSG := MSG \cup \{C \xrightarrow{t}_{mc} C''\}$;
    ∗ ELSE $MSG := MSG \cup \{C \xrightarrow{t}_{mc} C'\}$ and $add(New, C')$;
    ∗ ENDIF;
  – ENDDO;
• ENDDO.
```

We remark that the construction of this graph can be done by following the different paths in the reachability graph of the underlying Petri Net adding clock set χ' and relation $trans'$ and possibly ”unlooping” some loops of the reachability graph when a marking is reached many times with asoociated marking classes which are not clock-similar.

4.3.3 The marking class timed automaton

From the marking class graph defined previously, it is possible to build a timed automaton $\Gamma(\mathcal{T})$ which will have the same behavior as the TTPN $\mathcal{T}$, as we will show in the next section.

Definition 25 *Let $\mathcal{T} = \langle P, T, W^-, W^+, m_0, (\alpha, \beta)\rangle$ be a TTPN and $\Gamma'(\mathcal{T}) = \langle \mathcal{MC}, Mc_0, T, \rightarrow_{mc}\rangle$ its associated marking class graph. The marking class timed automaton $\Gamma(\mathcal{T})$ associated to $\mathcal{T}$ is the timed automaton $\langle L, L^0, \Sigma, X, I, E\rangle$ defined by :*

- *$L = \mathcal{MC}$ is the set of the marking classes ;*
- *$L^0 = \{Mc_0\}$, where Mc_0 is the initial marking class ($Mc_0 = \langle m_0, \{x_0\}, trans_0\rangle$);*
- *$X = \bigcup_{\langle M,\chi,trans\rangle \in \mathcal{MC}} \chi$;*
- *$\Sigma = T$;*
- *E is the set of switches defined by :*

$$\forall C_i = \langle M_i, \chi_i, trans_i\rangle \in \mathcal{MC}$$

$$\forall C_j = \langle M_j, \chi_j, trans_j\rangle \in \mathcal{MC}$$

$$\exists C_i \xrightarrow{t_i}_{mc} C_j \Leftrightarrow \exists (s_i, a, \phi, \lambda, \rho, s_j) \in E \text{ such that}$$

$$\begin{cases} s_i = C_i \text{ and} \\ s_j = C_j \text{ and} \\ a = t_i \text{ and} \\ \phi = (trans_i^{-1}(t_i,) \geq \alpha(t_i)) \text{ and} \\ \lambda = \{trans_j^{-1}(t_k) \mid \uparrow enabled(t_k, M_i, t_i) = True\} \text{ and} \\ \forall x \in \chi_i, \forall x' \in \chi_j, \text{ such that } trans_j(x') \subseteq trans_i(x) \text{ and } x' \notin \lambda, \rho(x') = x \end{cases}$$

- *$\forall C_i = \langle M_i, \chi_i, trans_i\rangle \in \mathcal{MC}$, $I(C_i) = \bigwedge_{x \in \chi_i, t \in trans_i(x)} (x \leq \beta(t))$.*

4.4 Properties of the marking class timed automaton

Now that we have a method to build the marking class timed automaton, we will give some properties over this timed automaton.

4.4.1 Bisimulation

In this section, we will define a binary relation between the states of the TTPN $\mathcal{T}$ and the states of its associated marking class timed automaton and we will prove that this relation is a bisimulation.

Definition 26 *Let $\mathcal{T} = \langle P,T,W^-,W^+,m_0,(\alpha,\beta)\rangle$ be a TTPN and $\Gamma(\mathcal{T})$ its associated marking class timed automaton. We consider $\bar{Q}_{\mathcal{T}}$ the set of reachable states of $\mathcal{T}$ and Q_A the set of states of $\Gamma(\mathcal{T})$. We define the relation $\simeq_{mc} \subseteq \bar{Q}_{\mathcal{T}} \times Q_A$ by, $\forall s = (M,\nu_{\mathcal{T}}) \in \bar{Q}_{\mathcal{T}}$, $\forall a = (Mc,\nu_A) \in Q_A$ (with $Mc = \langle M_a,\chi_a,trans_a\rangle$) :*

$$s \simeq_{mc} a \Leftrightarrow \begin{cases} M = M_a \textit{ and} \\ \\ \forall t \in T \textit{ such that } t \textit{ is enabled from } M, \\ \nu_{\mathcal{T}}(t) = \nu_A(x) \textit{ with } x \in \chi_a \textit{ such that } t \in trans_a(x) \end{cases}$$

Given the definition of the relation $\simeq_{mc}$, we have the following results.

Theorem 8 *For all $(s,a) \in \bar{Q}_{\mathcal{T}} \times Q_A$, if $s \simeq_{mc} a$ then :*

1. *$\forall \delta \in \mathbb{R}_{\geq 0}$, if $\exists s' \in \bar{Q}_{\mathcal{T}}$ such that $s \xrightarrow{\delta} s'$, then $\exists a' \in Q_A$ such that $a \xrightarrow{\delta} a'$ and $s' \simeq_{mc} a'$;*

2. *$\forall t \in T$, if $\exists s' \in \bar{Q}_{\mathcal{T}}$ such that $s \xrightarrow{t} s'$, then $\exists a' \in Q_A$ such that $a \xrightarrow{t} a'$ and $s' \simeq_{mc} a'$.*

Proof Let $s = (M,\nu_{\mathcal{T}})$ be a state in $\bar{Q}_{\mathcal{T}}$ and $a = (Mc,\nu_A)$ with $Mc = \langle M_a,\chi_a,trans_a\rangle$ be a state in Q_A such that $s \simeq_{mc} a$.
We consider $\delta \in \mathbb{R}_{\geq 0}$ and we suppose that there exists $s' \in \bar{Q}_{\mathcal{T}}$ such that $s \xrightarrow{\delta} s'$. Then $s' = (M,\nu'_{\mathcal{T}})$ with $\nu'_{\mathcal{T}} = \nu_{\mathcal{T}} + \delta$. For all $t \in T$, such that t is enabled from M, we have then (by definition of continuous transitions), $\nu_{\mathcal{T}}(t) + \delta \leq \beta(t)$. Since $s \simeq_{mc} a$, we can deduce that for all $t \in T$ such that t is enabled from M, and for $x \in \chi_a$ such that $t \in trans_a(x)$, we have $\nu_A(x) + \delta \leq \beta(t)$. For all $x \in \chi_a$, for all $t \in T$, if $t \in trans_a(x)$ then t is enabled from $M_a = M$ (by construction of the relation $trans$). Then we have $\bigwedge_{x\in\chi_a, t\in trans_a(x)}(\nu_A(x) + \delta \leq \beta(t))$, which means that $\nu_A + \delta$ satisfies $I(Mc) = \bigwedge_{x\in\chi_a, t\in trans_a(x)}(x \leq \beta(t))$. We deduce that there exist $a' \in Q_A$ such that $a \xrightarrow{\delta} a'$ with $a' = (Mc,\nu'_A)$ and $\nu'_A = \nu_A + \delta$.
The markings which appear in a' and s' are equal because the marking class of a' is the same as the one of a, the marking of s' is the same marking as the one from s and the marking of s and the marking of the marking class of a are equal (due to the fact that $s \simeq_{mc} a$). We consider $t \in T$ such that t is enabled from M, and denote x the clock in χ_a such that $t \in trans_a(x)$; then $\nu_{\mathcal{T}}(t) = \nu_A(x)$, and consequently $\nu_{\mathcal{T}}(t) + \delta = \nu_A(x) + \delta$. Since $\nu'_{\mathcal{T}} = \nu_{\mathcal{T}} + \delta$ and $\nu'_A = \nu_A + \delta$, and since the marking classes in a and in a' are the same, we can deduce that for all $t \in T$ such that t is enabled from M, for $x \in \chi_a$ such that $t \in trans_a(x)$, we have $\nu'_{\mathcal{T}}(t) = \nu'_A(x)$; hence $s' \simeq_{mc} a'$.

Now we consider $t \in T$, and we suppose that there exists $s' = (M',\nu'_{\mathcal{T}}) \in \bar{Q}_{\mathcal{T}}$ such that $s \xrightarrow{t} s'$. We can deduce that t is enabled from M, consequently there exists an edge of the form $Mc \xrightarrow{t}_{mc} Mc'$ in the marking class graph associated to the TTPN, and hence there exists a switch $e = (Mc,t,\phi,\lambda,\rho,Mc')$ in the associated marking class timed automaton. Furthermore, since we have $s \xrightarrow{t} s'$ for the TTPN, we can deduce that $\nu_{\mathcal{T}}(t) \geq \alpha(t)$. We consider $x \in \chi_a$ such that $t \in trans(x)$. Since $s \simeq_{mc} a$, we have $\nu_A(x) \geq \alpha(t)$, and since the guard ϕ on the switch e is by construction $x \geq \alpha(t)$, we conclude that ν_A satisfies ϕ. We can conclude that there exists $a' = (Mc',\nu'_A) \in Q_A$ such that $a \xrightarrow{t} a'$ and $Mc' = \langle M'_a,\chi'_a,trans'_a\rangle$ with, by construction of Mc', $M'_a = M_a + W^+(t) - W^-(t) = M + W^+(t) - W^-(t) = M'$.

Now we want to prove that $\forall t' \in T$ such that t' is enabled from M', for $x' \in \chi'_a$ such that $t \in trans'_a(x')$, we have $\nu'_A(x') = \nu'_{\mathcal{T}}(t')$. We consider a transition $t' \in T$ such that t' is enabled from M'. Two cases are possible :

1. $\uparrow enabled(t', M, t) = False$. This means that t' is enabled from M, and we deduce that for $x \in \chi_a$ such that $t \in trans_a(x)$, we have $\nu_{\mathcal{T}}(t) = \nu_A(x)$. By definition of $s \xrightarrow{t} s'$, we have $\nu'(t') = \nu(t')$. We denote x' the clocks of χ'_a such that $t' \in trans'_a(x')$. By construction of the marking class timed automaton, since t' is not newly enabled, we deduce that $trans'_a(x') \subseteq trans_a(x)$ (in fact, during the construction of the amrking class graph, when a set $trans(x)$ that contains enabled transtions which are not newly enabled, is built, no transition t is added to this set; only removal of transitions is performed, and here t' is not removed because the associated transition is not disabled by the firing of t). We deduce that the renaming function ρ of the switch is such that $\rho(x') = \rho(x)$. Hence we have, by construction of $a \xrightarrow{t} a'$, $\nu'_A(x') = \nu_A(\rho(x')) = \nu_A(x)$. Consequently $\nu'_A(x') = \nu'_{\mathcal{T}}(t')$.

2. $\uparrow enabled(t', M, t) = True$. We have by definition of $s \xrightarrow{t} s'$, $\nu'_{\mathcal{T}}(t') = 0$. By construction of the marking class graph, a new clock x' has been created for Mc' such that $t' \in trans'_a(x')$ and such that $x' \in \lambda$. Hence we have $\nu'_A(x') = \nu'_{\mathcal{T}}(t') = 0$.

We conclude that $s' \simeq_{mc} a'$.
□

Theorem 9 *For all $(s, a) \in \bar{Q}_{\mathcal{T}} \times Q_A$, if $s \simeq_{mc} a$ then :*

1. *$\forall \delta \in \mathbb{R}_{\geq 0}$, if $\exists a' \in Q_A$ such that $a \xrightarrow{\delta} a'$, then $\exists s' \in \bar{Q}_{\mathcal{T}}$ such that $s \xrightarrow{\delta} s'$ and $s' \simeq_{mc} a'$;*

2. *$\forall t \in T$, if $\exists a' \in Q_A$ such that $a \xrightarrow{t} a'$, then $\exists s' \in \bar{Q}_{\mathcal{T}}$ such that $s \xrightarrow{t} s'$ and $s' \simeq_{mc} a'$.*

Proof Let $s = (M, \nu_{\mathcal{T}})$ be a state in $\bar{Q}_{\mathcal{T}}$ and $a = (Mc, \nu_A)$ with $Mc = \langle M_a, \chi_a, trans_a \rangle$ be a state in Q_A such that $s \simeq_{mc} a$.
We consider $\delta \in \mathbb{R}_{\geq 0}$ and we suppose that there exists $a' \in Q_A$ such that $a \xrightarrow{\delta} a'$. Then $a' = (Mc, \nu'_A)$ with $\nu'_A = \nu_A + \delta$. For all $x \in \chi_a$, for all $t \in trans_a(x)$ we have then (by definition of continuous transitions), $\nu_A(x) + \delta \leq \beta(t)$. Since $s \simeq_{mc} a$, we can deduce that for all $t \in T$ such that t is enabled from M, and for $x \in \chi_a$ such that $t \in trans_a(x)$, we have $\nu_{\mathcal{T}}(t) + \delta \leq \beta(t)$. We deduce that there exists $s' \in \bar{Q}_{\mathcal{T}}$ such that $s \xrightarrow{\delta} s'$ with $s' = (Mc, \nu'_{\mathcal{T}})$ and $\nu'_{\mathcal{T}} = \nu_{\mathcal{T}} + \delta$.
The markings which appear in a' and s' are equal because the marking class of a' is the same as the one of a, the marking of s' is the same marking as the one from s, and the marking of s and the marking of a are equal (due to the fact that $s \simeq_{mc} a$). We consider $t \in T$ such that t is enabled from M, and denote x the clock in χ_a such that $t \in trans_a(x)$; then $\nu_{\mathcal{T}}(t) = \nu_A(x)$, and consequently $\nu_{\mathcal{T}}(t) + \delta = \nu_A(x) + \delta$. Since $\nu'_{\mathcal{T}} = \nu_{\mathcal{T}} + \delta$ and $\nu'_A = \nu_A + \delta$, since the marking classes in a and in a' are the same, we can deduce that for all $t \in T$ such that t is enabled from M, for $x \in \chi_a$ such that $t \in trans_a(x)$, we have $\nu'_{\mathcal{T}}(t) = \nu'_A(x)$; hence $s' \simeq_{mc} a'$.

Now we consider $t \in T$, and we suppose that there exists $a' = (Mc', \nu'_A) \in Q_A$ such that $a \xrightarrow{t} a'$ with $Mc' = \langle M'_a, \chi'_a, trans'_a \rangle$.This implies that there exists in the marking class timed automaton an edge of the form $(Mc, t, \phi, \lambda, \rho, Mc')$. We can deduce that t is enabled from M and we denote x the clock in χ_a such that $t \in trans_a(x)$. Furthermore, since $a \xrightarrow{t} a'$, ν_A satisfies $I(Mc)$ (the invariant of the location linked to Mc) and also ϕ. Since $I(Mc) = \bigwedge_{y \in \chi_a, t \in trans_a(y)} (y \leq \beta(t))$ and $\phi = x \geq \alpha(t)$, and since $s \simeq_{mc} a$, we conclude that $\alpha(t) \leq \nu_{\mathcal{T}}(t) \leq \beta(t)$. Consequently there exists $s' = (M', \nu'_{\mathcal{T}}) \in \bar{Q}_{\mathcal{T}}$ such that $s \xrightarrow{t} s'$ and $M' = M + W^+(t) - W^-(t) = M_a + W^+(t) - W^-(t) = M'_a$.
Now we want to prove that $\forall t' \in T$ such that t' is enabled from M', for $x' \in \chi'_a$ such that $t \in trans'_a(x')$, we have $\nu'_A(x') = \nu'_T(t')$. The proof of this property is the same as the one proposed in the previous theorem for the equivalent property. We conclude that $s' \simeq_{mc} a'$.
$\square$

From this two theorems, we can conclude the following result.

Theorem 10 *The binary relation $\simeq_{mc} \subset \bar{Q}_{\mathcal{T}} \times Q_A$ is a bisimulation.*

If we consider a TTPN $\mathcal{T} = \langle P, T, W^-, W^+, m_0, (\alpha, \beta) \rangle$ and its associated marking class timed automaton $\Gamma(\mathcal{T})$, since we have by construction $(m_0, \bar{0}) \simeq_{mc} (Mc_0, \bar{0})$, we conclude that a marking m is reachable from m_0 in $\mathcal{T}$ if and only if there exists a state of $\Gamma(\mathcal{T})$ whose associated marking is m, furthermore by finding all the states of $\Gamma(\mathcal{T})$ which accept m as marking and the sequences which lead to them, we can obtain all the sequences in $\mathcal{T}$ which lead to m; this will allow us to verify real-time properties of $\mathcal{T}$ by verifiyng them on $\Gamma(T)$.

4.4.2 Boundedness

In order to build the marking class timed automaton of a Transitions Time Petri Net, the number of marking classes has to be bounded, otherwise the construction of the marking class graph will not terminate.

Theorem 11 *A Transitions Time Petri Net has a bounded number of marking classes if and only if the underlying Petri Net (i.e. the Petri Net obtained by untiming the TTPN) is bounded.*

Proof From the construction of the marking state class graph, we conclude that, for each marking present in the reachability graph of the underlying Petri Net, there exists at least one marking class in the marking class graph. Since two different classes with the same marking differ only with regard to the partition of the set of the enabled transitions, and since the number of transitions is finite, we conclude that if the underlying Petri Net is bounded, it has a bounded number of markings, and consequently the number of marking classes of the Transitions Time Petri Net is bounded. If the underlying Petri Net is not bounded, then it has an infinite number of markings and the TTPN has also an infinite number of marking classes.
$\square$

From this theorem, we deduce that we will be able to build the marking class timed automaton only for TTPN which have bounded underlying Petri Nets. In constrast to the case of the boundedness of Transitions Time Petri Nets, the boundedness of a Petri Net is decidable [16] .

Theorem 12 *A Petri Net $\langle P,T,W^-,W^+,m_0\rangle$ is not bounded if and only if there exist two markings $M,M' \in \mathcal{R}_{un}(m_0)$ which fulfill the following conditions :*

1. *M' is reachable from M ;*
2. *$M' > M$.*

Consequently, it is possible to build the marking class graph on the fly: if a pair of markings which verify the previous conditions are encountered then the algorithm stops, because the underlying Petri Net is not bounded.

4.5 Extension to the Transitions Time Well-formed Nets

Just as we had extended in chapter 3 the state class timed automaton of a TTPN to TTWN, we can adapt the marking state class timed automaton to TTWN. In this section, we will only give the definitions, algorithms and properties linked to the marking class timed automaton of a TTWN, because the proofs and the explanations are the same as the ones for the TTPN.

4.5.1 Marking class graph of a TTWN

For a TTWN $\mathcal{T} = \langle P,T,W^-,W^+,Cl,\mathcal{C},\Phi,(\alpha,\beta),m_0\rangle$, we denote $\mathcal{R}_{un}(m_0)$ the set of the markings reachable in the untimed Well-formed net $\langle P,T,W^-,W^+,Cl,\mathcal{C},\Phi,m_0\rangle$ from the initial marking m_0.

Definition 27 *A marking class of a TTWN $\mathcal{T} = \langle P,T,W^-,W^+,Cl,\mathcal{C},\Phi,(\alpha,\beta),m_0\rangle$ is a tuple $\langle M,\chi,trans\rangle$ where M is a marking belonging to $\mathcal{R}_{un}(m_0)$, χ is a set of real valued clocks and $trans \in (2^{\mathcal{CT}})^{\chi}$ is a relation, which maps clocks with sets of $\mathcal{CT}$.*
We will denote $\mathcal{MC}$ the set of the marking classes of a TTWN.

Just as for the TTPN, each pair (t,c) of $\mathcal{CT}$ must be associated to at most one clock in the relation $trans$.

Definition 28 *Two marking classes $Mc = \langle M,\chi,trans\rangle$ and $Mc' = \langle M',\chi',trans'\rangle$ of a TTWN are clock-similar, denoted $Mc \approx Mc'$ if and only if they have the same marking, the same number of clocks and their clocks are mapped to the same transitions, i.e. :*

$$Mc \approx Mc' \Leftrightarrow \begin{cases} M = M' \\ |\chi| = |\chi'| \\ \forall x \in \chi, \exists x' \in \chi', trans(x) = trans'(x') \end{cases}$$

We will also denote $\Gamma'(\mathcal{T})$ the marking class graph of a TTWN $\mathcal{T}$, which is a transition system $\Gamma'(\mathcal{T}) = \langle \mathcal{MC}, Mc_0, \mathcal{CT}, \rightarrow_{mc}\rangle$. The initial marking class $Mc_0 = \langle m_0, \{\chi_0\}, trans_0\rangle$ is defined by :

- m_0 is the initial marking of $\mathcal{T}$;
- The set of clocks χ_0 of Mc_0 is composed of a single clock x_0;
- $trans_0$ is defined by $trans_0(x_0) = \{t \in T \mid t \text{ is enabled for } m_0\}$.

The algorithm to compute the marking class graph can be written as follows :

```
• The variables are $MSG$ (the marking state graph itself) and a FIFO queue $New$
  (which contains the newly computed marking classes);
• The variables are initialized to : $MSG := \{Mc_0\}$ and $New := Mc_0$;
• WHILE $New$ is NOT empty DO
    – $C := remove(New)$ (The first marking class of $New$ is taken, $C := \langle M_C, \chi_C, trans_C \rangle$);
    – For all pairs $(t_i, c_{i,j}) \in \mathcal{CT}$ such that $t_i$ is enabled for $c_{i,j}$ in $M_C$ (i.e. $\forall p \in P, M_C(p) \geq W^-(p,t_i)(c_{i,j})$ and $\Phi(t_i)(c_{i,j}) = True$) DO
        ∗ (Computation of the marking class $C' = \langle M_{C'}, \chi_{C'}, trans_{C'} \rangle$ son of $C$ )
        ∗ For all $p \in P, M_{C'}(p) = M_C(p) + W^+(p,t_i)(c_{i,j}) - W^-(p,t_i)(c_{i,j})$;
        ∗ Computation of $\chi_{C'}$ and $trans_{C'}$ :
            1. For each clock $x \in \chi_C$, remove from $trans_C(x)$ all the pairs $(t_k, c_{k,l})$
               such that $t_k$ is enabled for $c_{k,l}$ in $M_C$ and not for the marking $M'$ de-
               fined by $\forall p \in P$, $M'(p) = M_C(p) - W^-(p,t_i)(c_{i,j})$, to obain a relation
               $trans'$;
            2. The clocks whose image by $trans'$ is empty are removed from $\chi_C$, to
               obtain a set of clocks $\chi'$;
            3. For all pairs $(t_k, c_{k,l}) \in \mathcal{CT}$ which verify $\uparrow enabled((t_k, c_{k,l}), M_C, (t_i, c_{i,j})) =$
               $True$ DO :
                + IF a clock $x$ has already been created for the computation of $C'$
                + THEN $(t_k, c_{k,l})$ is added to $trans'(x)$;
                + ELSE a new clock $x_n$ is created; $n$ is the smallest available index
                  among the clocks of $\chi'$ and $trans'(x_n) = (t_k, c_{k,l})$;
            4. ENNDO; $\chi_{C'}$ and $trans_{C'}$ are then obtained from the resulting set $\chi'$
               and function $trans'$;
        ∗ IF there is a marking class $C''$ in $MSG$ such that $C' \approx C''$
        ∗ THEN $MSG := MSG \cup \{C \overset{(t_i, c_{i,j})}{\rightarrow}_{mc} C''\}$;
        ∗ ELSE $MSG := MSG \cup \{C \overset{(t_i, c_{i,j})}{\rightarrow}_{mc} C'\}$ and $add(New, C')$;
        ∗ ENDIF;
    – ENDDO;
• ENDDO.
```

4.5.2 Marking class timed automaton of a TTWN

The marking class $\Gamma(\mathcal{T})$ can then be defined.

Definition 29 *Let $\mathcal{T} = \langle P, T, W^-, W^+, Cl, \mathcal{C}, \Phi, (\alpha, \beta), m_0 \rangle$ be a TTWN and $\Gamma'(\mathcal{T}) = \langle \mathcal{MC}, Mc_0, \mathcal{CT}, \rightarrow_{mc} \rangle$ its associated marking class graph. The marking class timed automaton $\Gamma(\mathcal{T})$ associated to $\mathcal{T}$ is the timed automaton $\langle L, L^0, \Sigma, X, I, E \rangle$ defined by :*

- *$L = \mathcal{MC}$ is the set of the marking classes ;*
- *$L^0 = \{Mc_0\}$, where Mc_0 is the initial marking class ($Mc_0 = \langle m_0, \{x_0\}, trans_0 \rangle$);*
- *$X = \bigcup_{\langle M, \chi, trans \rangle \in \mathcal{MC}} \chi$;*
- *$\Sigma = \mathcal{CT}$;*
- *E is the set of switches defined by :*

$$\forall C_i = \langle M_i, \chi_i, trans_i \rangle \in \mathcal{MC}$$
$$\forall C_j = \langle M_j, \chi_j, trans_j \rangle \in \mathcal{MC}$$
$$\exists C_i \overset{(t_i, c_{i,j})}{\rightarrow}_{mc} C_j \Leftrightarrow \exists (s_i, a, \phi, \lambda, \rho, s_j) \in E \text{ such that}$$

$$\begin{cases} s_i = C_i \text{ and} \\ s_j = C_j \text{ and} \\ a = (t_i, c_{i,j}) \text{ and} \\ \phi = (trans_i^{-1}((t_i, c_{i,j})) \geq \alpha(t_i)) \text{ and} \\ \lambda = \{trans_j^{-1}((t_k, c_{k,l})) \mid \uparrow enabled((t_k, c_{k,l}), M_i, (t_i, c_{i,j})) = True\} \text{ and} \\ \forall x \in \chi_i, \forall x' \in \chi_j, \text{ such that } trans_j(x') \subseteq trans_i(x) \text{ and } x' \notin \lambda, \rho(x') = x \end{cases}$$

- *$\forall C_i = \langle M_i, \chi_i, trans_i \rangle \in \mathcal{MC}$, $I(C_i) = \bigwedge_{x \in \chi_i, (t,c) \in trans_i(x)} (x \leq \beta(t))$.*

4.5.3 Properties

The properties which were found for the marking class timed automaton of a TTPN can be extended to the marking class timed automaton of a TTWN. The proofs are substantially the same, the main difference being that we consider pairs (t, c) which belong to $\mathcal{CT}$ instead of considering only transitions.

Bisimulation

Definition 30 *Let $\mathcal{T} = \langle P, T, W^-, W^+, Cl, \mathcal{C}, \Phi, (\alpha, \beta), m_0 \rangle$ be a TTWN and $\Gamma(\mathcal{T})$ its associated marking class timed automaton. We consider $\bar{Q}_{\mathcal{T}}$ the set of reachable states of $\mathcal{T}$ and Q_A the set of states of $\Gamma(\mathcal{T})$. We define the relation $\simeq_{mc} \subseteq \bar{Q}_{\mathcal{T}} \times Q_A$ by, $\forall s = (M, \nu_{\mathcal{T}}) \in \bar{Q}_{\mathcal{T}}$, $\forall a = (Mc, \nu_A) \in Q_A$ (with $Mc = \langle M_a, \chi_a, trans_a \rangle$) :*

$$s \simeq_{mc} a \Leftrightarrow \begin{cases} M = M_a \text{ and} \\ \\ \forall (t, c) \in \mathcal{CT} \text{ such that } t \text{ is enabled for } c \text{ in } M, \\ \nu_{\mathcal{T}}((t, c)) = \nu_A(x) \text{ with } x \in \chi_a \text{ such that } (t, c) \in trans_a(x) \end{cases}$$

Theorem 13 *For all $(s,a) \in \bar{Q}_{\mathcal{T}} \times Q_A$, if $s \simeq_{mc} a$ then :*

1. *$\forall \delta \in \mathbb{R}_{\geq 0}$, if $\exists s' \in \bar{Q}_{\mathcal{T}}$ such that $s \xrightarrow{\delta} s'$, then $\exists a' \in Q_A$ such that $a \xrightarrow{\delta} a'$ and $s' \simeq_{mc} a'$;*
2. *$\forall (t,c) \in \mathcal{CT}$, if $\exists s' \in \bar{Q}_{\mathcal{T}}$ such that $s \xrightarrow{(t,c)} s'$, then $\exists a' \in Q_A$ such that $a \xrightarrow{(t,c)} a'$ and $s' \simeq_{mc} a'$.*

Theorem 14 *For all $(s,a) \in \bar{Q}_{\mathcal{T}} \times Q_A$, if $s \simeq_{mc} a$ then :*

1. *$\forall \delta \in \mathbb{R}_{\geq 0}$, if $\exists a' \in Q_A$ such that $a \xrightarrow{\delta} a'$, then $\exists s' \in \bar{Q}_{\mathcal{T}}$ such that $s \xrightarrow{\delta} s'$ and $s' \simeq_{mc} a'$;*
2. *$\forall (t,c) \in \mathcal{CT}$, if $\exists a' \in Q_A$ such that $a \xrightarrow{(t,c)} a'$, then $\exists s' \in \bar{Q}_{\mathcal{T}}$ such that $s \xrightarrow{(t,c)} s'$ and $s' \simeq_{mc} a'$.*

Theorem 15 *The binary relation $\simeq_{mc} \subset \bar{Q}_{\mathcal{T}} \times Q_A$ is a bisimulation.*

Boundedness

With regards to the the number of marking classes of TTWN, we have the following theorem.

Theorem 16 *A TTWN has a bounded number of marking classes if and only if the underlying Well-formed Net (i.e. the Well-formed Net obtained by untiming the TTWN) is bounded.*

With the regards to the boundedness of a Well-formed Net, the following theorem can be deduced from the equivalent theorem for Petri Nets.

Theorem 17 *A Well-formed Net $\mathcal{T} = \langle P, T, W^-, W^+, Cl, \mathcal{C}, \Phi, m_0 \rangle$ is not bounded if and only if there exist two markings $M, M' \in \mathcal{R}_{un}(m_0)$ which fulfill the following conditions :*

1. *M' is reachable from M;*
2. *$\forall p \in P$, $M'(p) \geq M(p)$ and $\exists p \in P$, $\exists c \in \mathcal{C}(p)$ such that $M'(p)(c) > M(p)(c)$.*

4.6 An example

As in chapter 3, we present here an exemple based on the problem of the dining philosophers to which we added time constraints. The TTWN corresponding to this problem is given in Figure 3.1.
The ordinary reachability graph of the underlying Well-formed Net is given by the Figure 4.1.

In this ordinary reachability graph, the markings correspond to the following list :

- $M_0(Thinking) = <1> + <2> + <3> + <4>$, $M_0(Forks) = <1> + <2> + <3> + <4>$ and $M_0(Eating) = \emptyset$;

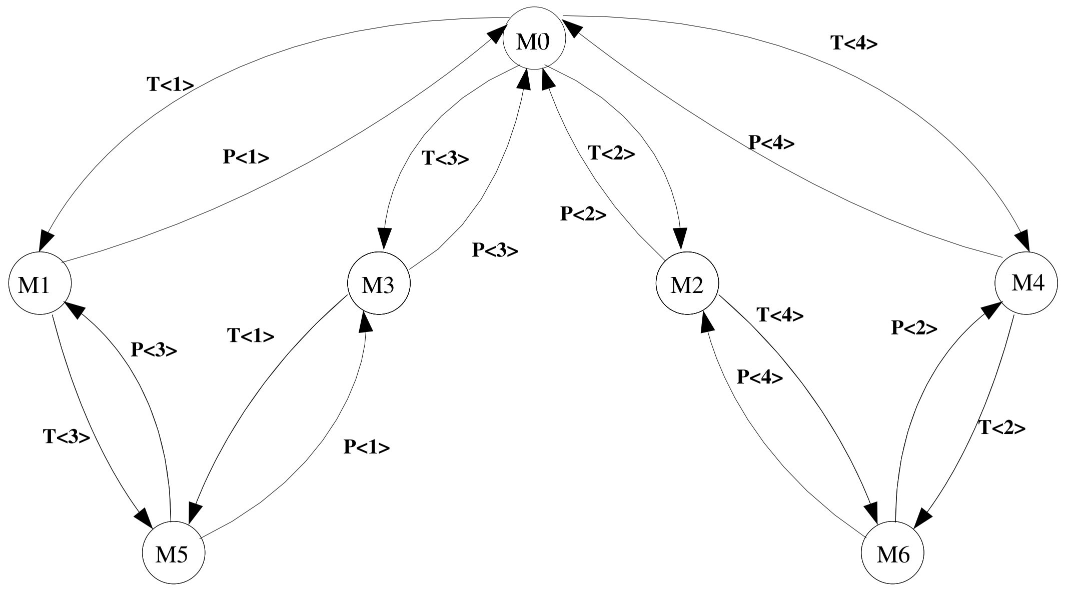


Figure 4.1: Reachability graph for the problem of the philosophers

- $M_1(Thinking) = <2> + <3> + <4>$, $M_0(Forks) = <3> + <4>$ and $M_1(Eating) = <1>$;
- $M_2(Thinking) = <1> + <3> + <4>$, $M_1(Forks) = <1> + <4>$ and $M_2(Eating) = <2>$;
- $M_3(Thinking) = <1> + <2> + <4>$, $M_3(Forks) = <1> + <2>$ and $M_3(Eating) = <3>$;
- $M_4(Thinking) = <1> + <2> + <3>$, $M_4(Forks) = <2> + <3>$ and $M_4(Eating) = <4>$;
- $M_5(Thinking) = <2> + <4>$, $M_4(Forks) = \emptyset$ and $M_5(Eating) = <1> + <3>$;
- $M_6(Thinking) = <1> + <3>$, $M_4(Forks) = \emptyset$ and $M_6(Eating) = <2> + <4>$.

The marking class timed automaton of this TTWN is given by the Figure 4.2.

The description of the different marking classes which feature on this timed automaton is given by :

- $Mc_0 = \langle M_0, \chi_0, trans_0 \rangle$ with :
  1. $\chi_0 = \{x_0\}$;
  2. $trans_0(x_0) = \{(T, <1>), (T, <2>), (T, <3>), (T, <4>)\}$.
- $Mc_1 = \langle M_1, \chi_1, trans_1 \rangle$ with :
  1. $\chi_1 = \{x_0, x_1\}$;

2. $trans_1(x_0) = \{(T, <3>)\}$ and $trans_1(x_1) = \{(P, <1>)\}$.

- $Mc_2 = \langle M_2, \chi_2, trans_2 \rangle$ with :
  1. $\chi_2 = \{x_0, x_1\}$;
  2. $trans_2(x_0) = \{(T, <4>)\}$ and $trans_2(x_1) = \{(P, <2>)\}$.
- $Mc_3 = \langle M_3, \chi_3, trans_3 \rangle$ with :
  1. $\chi_3 = \{x_0, x_1\}$;
  2. $trans_3(x_0) = \{(T, <1>)\}$ and $trans_3(x_1) = \{(P, <3>)\}$.
- $Mc_4 = \langle M_4, \chi_4, trans_4 \rangle$ with :
  1. $\chi_4 = \{x_0, x_1\}$;
  2. $trans_4(x_0) = \{(T, <2>)\}$ and $trans_4(x_1) = \{(P, <4>)\}$.
- $Mc_5 = \langle M_0, \chi_5, trans_5 \rangle$ with :
  1. $\chi_5 = \{x_0, x_1\}$;
  2. $trans_5(x_0) = \{(T, <3>)\}$ and $trans_5(x_1) = \{(T, <1>), (T, <2>), (T, <4>)\}$.
- $Mc_6 = \langle M_5, \chi_6, trans_6 \rangle$ with :
  1. $\chi_6 = \{x_0, x_1\}$;
  2. $trans_6(x_0) = \{(P, <3>)\}$ and $trans_6(x_1) = \{(P, <1>)\}$.
- $Mc_7 = \langle m_0, \chi_7, trans_7 \rangle$ with :
  1. $\chi_7 = \{x_0, x_1\}$;
  2. $trans_7(x_0) = \{(T, <4>)\}$ and $trans_7(x_1) = \{(T, <1>), (T, <2>), (T, <4>)\}$.
- $Mc_8 = \langle M_6, \chi_8, trans_8 \rangle$ with :
  1. $\chi_8 = \{x_0, x_1\}$;
  2. $trans_8(x_0) = \{(P, <4>)\}$ and $trans_8(x_1) = \{(P, <2>)\}$.
- $Mc_9 = \langle M_0, \chi_9, trans_9 \rangle$ with :
  1. $\chi_9 = \{x_0, x_1\}$;
  2. $trans_9(x_0) = \{(T, <1>)\}$ and $trans_9(x_1) = \{(T, <2>), (T, <3>), (T, <4>)\}$.
- $Mc_{10} = \langle M_0, \chi_{10}, trans_{10} \rangle$ with :
  1. $\chi_{10} = \{x_0, x_1\}$;
  2. $trans_{10}(x_0) = \{(T, <2>)\}$ and $trans_{10}(x_1) = \{(T, <1>), (T, <3>), (T, <4>)\}$.

4.7 Advantages and drawbacks of the marking class timed automaton

4.7.1 Advantages

The computation of the marking class timed automaton is easier than the computation of the extended state class timed automaton, because all the operations which are linked to firing domains, featured in the algorithm to obtain the extended state timed auomaton, are suppressed. Furthermore, there already exist many tools to compute the reachability graph of a Petri Net which is the first task required when obtaining the marking class timed automaton. This method of computing thr marking class timed automaton from the reachability graph coud also be released for TTWN to find a way in the future to compute a timed automaton with symbolic markings. Even if some symmetries are suppressed with the introducing of time constraints, it could be interesting to find a method that computes from the symbolic reachability graph of the underlying Well-formed Net a timed automaton which has the same behavior as the TTWN. In the case of the TTWN, suppressing the computing of firing domain may be advantageaous, because the size of the firing domain depends on transition-colour pair, which means that the size of this firing domain could potentially be large and consequently the diverse operations on this firing domain (such as Fourier-Motzkin method) could take time and memory. Furthermore, the algorithm for the computation of the marking class graph is easier to implement than the algorithm for the extended state class graph because, we do not have to make domain comparaisons, for instance.

4.7.2 Drawbacks

Even if the marking class timed automaton brings some advantages, there are also some drawbacks that should not be neglected. First, the extended state class timed automaton is included in the marking class timed automaton (if the extended state classes are replaced by marking classes, ie the domains are suppressed).However, it is possible that the marking class timed automaton has more locations (for instance markings that are reached in the underlying Petri Net, but never in the TTPN) and also more switches, and that these locations and switches are never used for the behavior of the system.Therefore the marking class timed automaton can be bigger than the extended state class timed automaton.

For some TTWN, it is also possible to build the state class timed automaton whereas the construction of the marking class timed automaton cannot be done. It is possible for a bounded TTPN that the underlying Petri Net is not bounded; furthermore it is possible that a TTPN verifies the sufficient conditions for boundedness that were presented in chapter 3, yet its underlying Petri Net is not bounded. For instance if we consider the TTPN shown in Figure 4.3, its underlying Petri Net is not bounded since it is possible to put an infinity of tokens in the place $P3$, and hence it is not possible to build its marking class timed automaton. However, when the time constraints are considered, we see that the transition $T2$ is never fired for the TTPN. The TTPN is in fact bounded, and it is possible to build its state class timed automaton which is shown in Figure 4.4. The description of the extended state classes of this TTPN is the following :

- $Cl_0 = \langle m_0, D_0, \chi_0, trans_0 \rangle$ with :

  1. $m_0(P1) = 1$ and $m_0(P2) = m_0(P3) = m_0(P4) = 0$;
  2. D_0 is defined by :
  $$\begin{cases} 0 \leq \theta_1 \leq 2 \\ 3 \leq \theta_2 \leq 4 \end{cases}$$
  3. $\chi_0 = \{x_0\}$;
  4. $trans_0(x_0) = \{T1, T2\}$.

- $Cl_1 = \langle m_1, D_1, \chi_1, trans_1 \rangle$ with :

  1. $m_1(P2) = 1$ and $m_1(P1) = m_1(P3) = m_1(P4) = 0$;
  2. D_1 is defined by :
  $$\begin{cases} 1 \leq \theta_3 \leq 2 \end{cases}$$
  3. $\chi_1 = \{x_0\}$;
  4. $trans_1(x_0) = \{T3\}$.

- $Cl_2 = \langle m_2, D_2, \chi_2, trans_2 \rangle$ with :

  1. $m_2(P1) = m_2(P2) = m_2(P3) = m_2(P4) = 0$;
  2. D_2 is defined by the empty set of constraints;
  3. $\chi_2 = \emptyset$.

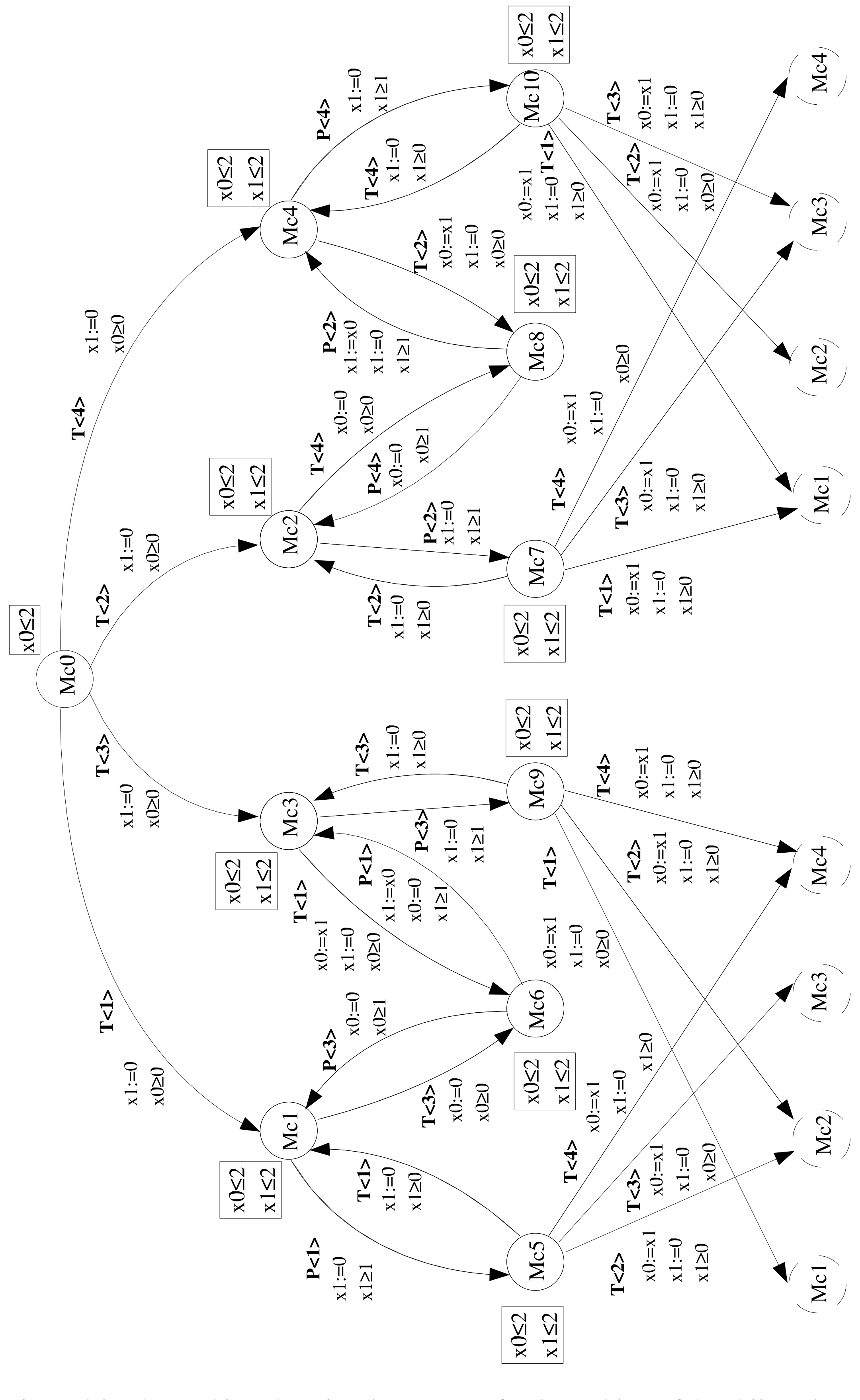


Figure 4.2: The marking class timed automaton for the problem of the philosophers

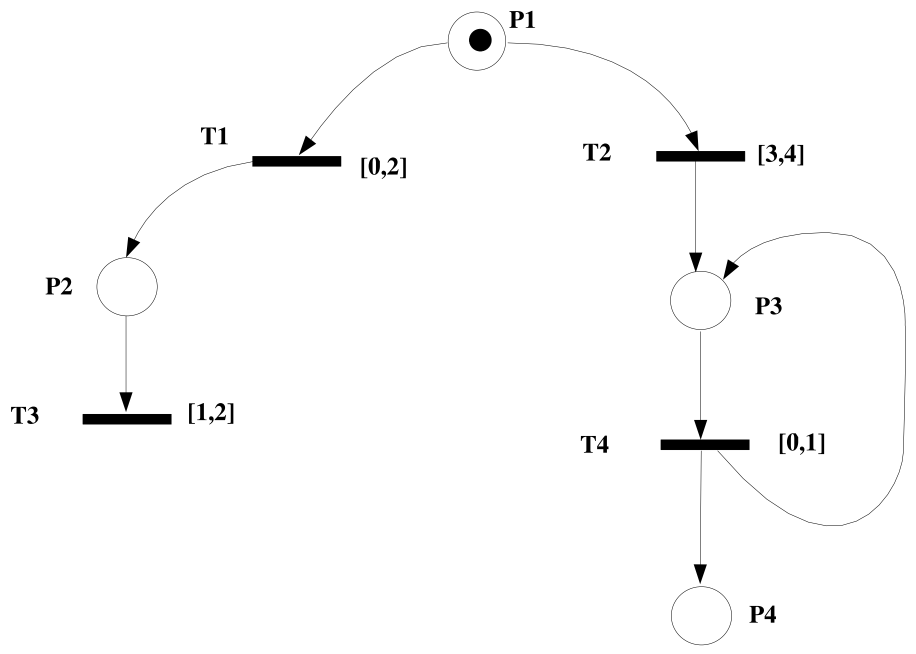


Figure 4.3: An example of bounded TTPN with an unbounded underlying Petri Net

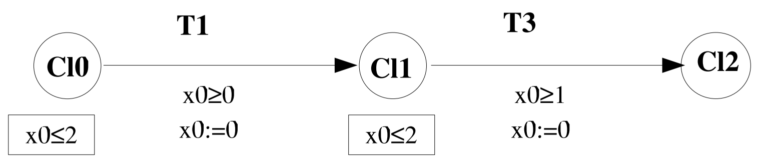


Figure 4.4: The state class timed automaton associated with the TTPN of 4.3

Chapter 5

GSPN2TA : A tool for computing Marking Class Timed Automata

5.1 Introduction

In this chapter, we present the tool that we have implemented for the computation of the marking class timed automaton of Transitions Time Petri Net, where the underlying Petri Net is described with the tool GreatSPN. We will first describe how the tool can be used, then we will see how it has been implemented. Finally we will give the results that we obtained and make a comparison with the tool Romeo, which can compute the state class timed automaton of Transitions Time Petri Nets.

5.2 Using GSPN2TA

5.2.1 GreatSPN, Kronos and Uppaal

GreatSPN

GreatSPN is a software package for the modelling, validation and performance evaluation of distributed systems using Generalized Stochastic Petri Nets and their colored extension, the Stochastic Well-formed Nets. GreatSPN was developed by the Performance Evaluation group of the University of Torino. GreatSPN offers a graphical interface which can be used to create Petri Net models and can be used to compute the reachability graph. When the Petri Net is drawn and saved , it is stored into two files with the extensions .net and .def. It is also possible to obtain a description of the reachability graph with a function which can be called in command line, and which is called showRG.

Information on how to obtain GreatSPN is given at http://www.di.unito.it/ greatspn/.
The version of GreatSPN that we used is *GreatSPN2.0*.

Kronos and Uppaal

Kronos [24] and Uppaal [3] are tools which allow the verification of real-time systems. Kronos was developed by the laboratory Verimag, and Uppaal was jointly developed by

Uppsala University and Aalborg University. These two tools are designed to verify systems that can be modelled with timed automata. The timed automata produced by our tool are described in the input format of these two tools.

The automata given to Kronos are described in a file with the extension *.tg*. Kronos accepts renaming of clocks, and it is possible to label each location with a set of identifiers. We will use this latter fact to associate in the automata the description of the markings with the corresponding marking class.

The automata given to Uppaal are described in a XML file. Uppaal does not allow the renaming of clocks and, it appears to be less natural to associate identifiers to the locations of a timed automata. This latter point is problematic because the marking class timed automaton given to Uppaal does not include information about the markings of the marking classes. This is why when the marking class timed automaton is produced in the input format of Uppaal, another file is produced which contains this information. Despite these difficulties, we were nevertheless motivated to give a translation to Uppaal because its performance is competitive.

Kronos is available at http://www-verimag.imag.fr/TEMPORISE/kronos/ and Uppaal at http://www.uppaal.com The version of Kronos to which we refer is *Kronos2.5* and the version of Uppaal to which we refer is *UPPAAL3.4*.

5.2.2 User's manual

The tool that we have developed allows the computation of a marking class timed automaton of a Transitions Time Petri Net. The marking class timed automaton computed can be produced into the input format of Kronos, in which case a file with the file extension *.tg* is created, or into the input format of Uppaal, in which case a XML file and a file with the file's extension *.ta_desc* which contains the description of the firings and of the markings are produced. In both cases, the user has to define the Transitions Time Petri Net. In the following part, we will suppose that our Transitions Time Petri Net is called net.

Description of the Transitions Time Petri Net

The description of the net is done in two steps :

1. Description of the underlying Petri Net with GreatSPN (file net.net and net.def are produced);

2. Association of time constraints to the transitions in the file net.tcons.

When the user describes the underlying Petri Net using GreatSPN, he has to define all the transitions as exponentially timed distributed transitions; furthermore all the names of the entities (places and transitions) should be of the form [a-zA-Z][a-zA-Z1-9]*, which means that they have to begin with a letter and all their characters have to be alphanumeric. When the user has drawn the underlying Petri Net with GreatSPN, he saves it with the name net and he obtains two files net.net and net.def. These files can also be created in other ways because their format are described in [14].

To add the time constraints to the Petri Net, the user can run the program addTime net that is included in the tool we developed. This tool parses the file net.net and finds all the transitions; it is then possible to add to each transition an earliest and a latest firing time. The earliest firing time has to be a positive integer and the latest firing time can be a positive integer or -1 can be input if the latest firing time is equal to infinity. The fact that the firing times are represented with integers and not with rationals is not problematic since from a set of rational numbers it is always possible to obtain an equivalent set of integers by multiplying all the numbers by the greatest denominator. When the user has finished to describe the firing times, the file net.tcons is then created. The form of this file is described by the table 5.1 (in this table the term newline refers to the character of newline and the term empty is used to say that there is no character). All the transitions of the

LIST_TRANS	:=	empty | TRANS LIST_TRANS
TRANS	:=	*name_trans earliest_firing_time latest_firing_time* newline

Table 5.1: Form of the .tcons file

underlying Petri Net have to appear in this file. This file can as well be created manually but if an error is made during its creation the tool will not then be able to compute the timed automaton correctly.

Obtaining of the marking class timed automaton

Once the Transitions Time Petri Net has been described by the user, the program to compute the marking class timed automatoncan be executed. The tool offers two programs, respectively called Net2ta_Kronos and Net2ta_Uppaal. The first program, Net2ta_Kronos, computes the marking class timed automaton with eventual renamings of clocks and produces the file net.tg which can be analysed by Kronos. The second program, Net2ta_Uppaal, computes the marking class timed automaton without renaming of clocks and produces the file net.xml which can be analysed by Uppaal, and the file net.ta_desc, whose form is given by the table 5.2.

Once the user has obtained the marking class timed automaton in the required format,

MARK_CLASS_DESC	:=	/***MARKINGS DESCRIPTION***/ newline newline MARK_CLASS_LIST #***TRANSITIONS DESCRIPTION***# newline newline TRANS_LIST newline $***END OF DESCRIPTION ***$
MARK_CLASS_LIST	:=	empty | MARK_CLASS newline MARK_CLASS_LIST
MARK_CLASS	:=	Marking class [id*id_marking_class*] : newline MARK_LIST
MARK_LIST	:=	empty | MARK MARK_LIST
MARK	:=	->[*name_place*] *number_tokens* tokens newline
TRANS_LIST	:=	empty | TRANS TRANS_LIST
TRANS	:=	[id*id_marking_class_src*] -> *name_trans* ->[id*id_marking_class_dest*] newline

Table 5.2: Form of the .ta_desc file

he can use the tool (Uppaal or Kronos) to analyse the properties of the timed automaton. With Kronos the user can use the tool in a relatively direct way, because the markings and the fired transitions are given with the timed automaton as properties of the locations and labels of the switches. In fact in the file produced for Kronos, each location is associated with a list of properties which characterizes the marking of the location; each element of this list has the form *id_num* where id represents the name of a place with a strictly positive number of tokens and num represents the number of tokens present in the place. However, with Uppaal the user has to refer to the file net.ta_desc to have the correspondance between the properties of the given timed automaton and the marking class timed automaton. It is to be noted that since Kronos does not accept locations without successors, we create a switch labeled no_trans which loops in the marking classes without any successor.

An important point concerning the tool that we have implemented is that if the underlying Petri Net is not bounded, then the reachability graph will not be computed by GreatSPN and the marking class timed automaton will not be computed; instead an error message will be given to the user.

5.3 Operation of the tool

In this section, we will give a description of the operation of the tool GSPN2TA and of the different data structures which have been used for its implementation.

5.3.1 General operation

When the user calls the programs Net2ta_Kronos or Net2ta_Uppaal, the program showRG of GreatSPN is executed in order to compute the reachability graph of the underlying Petri Net contained in the file net.def and net.net. This reachability graph is then stored in a file called net.rg_desc, which can also be consulted by the user. Once the reachability graph ahs been computed, the program callq the program net2ta_kronos (or net2ta_uppaal), which has the following behavior :

1. Loading the described net (analysis of net.net);
2. Loading the time constraints (analysis of net.tcons);
3. Loading the reachability graph (analysis of the file net.rg_desc);
4. Computation of the marking class timed automaton (with or without renamings of clocks);
5. Writing of the timed automaton in the corresponding files.

We will give in the following sections a description of these different steps.

5.3.2 Loading the net and the time constraints

To obtain the informations stored in the file net.net, the program parses the file by calling the function ”net_struct load_net(FILE* file)” (implemented in the file load_net.c) which takes as argument the file descriptor of the file net.net and which returns a net_struct (implemented in the file structure.h). A net_struct is a data structure in which the data of the Transitions Time Petri Net are stored. The figure 5.1 shows the form of this data structure.

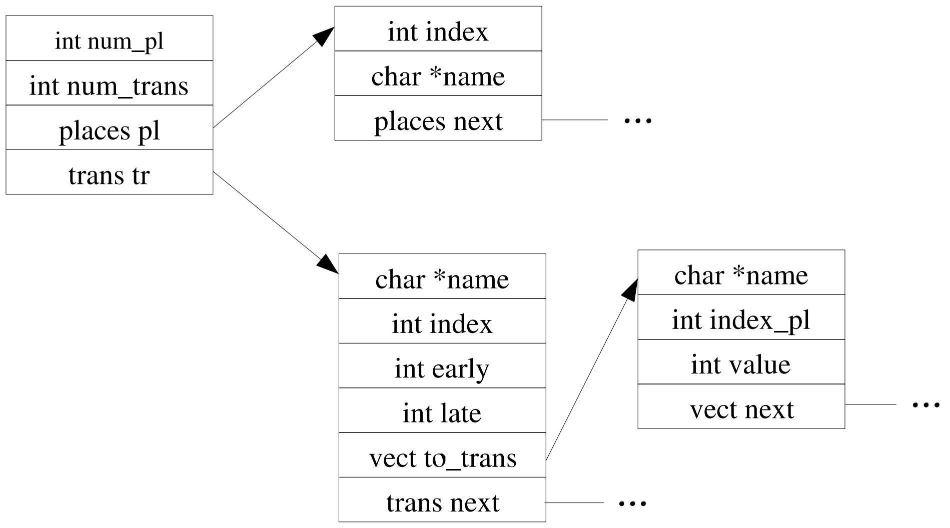


Figure 5.1: The data structure net_struct

The date structure net_struct has the following fields :

- num_pl stores the number of places of the net ;
- num_trans stores the number of transitions of the net ;
- pl stores the list of the places of the net; it is of the type places which has the fields :
  - index stores the index of the place;
  - name stores the name of the place;
  - next stores the next element of the list (NULL if there is no next element).
- tr stores the list of the transitions of the net; it is of the type trans which has the fields :
  - name stores the name of the transition;
  - index stores the index of the transition;
  - early stores the earliest firing time of the transition;
  - late stores the latest firing time of the transition;

- to_trans stores the list of input arcs of the transitions; it is of the type vect which has the fields :
    - ∗ name which is here always equal to NULL (in fact it is used in an other data structure);
    - ∗ index_pl stores the index of the place to which the transition is linked;
    - ∗ value stores the value of the input arc;
    - ∗ next stores the next input arc (NULL if there is no next element).
- next stores the transition (NULL if there is no next element).

The function load_net does not fill the fields early and late of the transitions because the relevant data is not written in the file net.net; in fact, these fields are filled by calling the function ”void load_time_cons(net_struct net,FILE *file)” (implemented in the file load_net.c) which takes as argument the net_struct built with the function load_net and the file descriptor of the file net.tcons. The list of input arcs associated to a transition groups only the arcs which are associated with a strictly positive value. The output arcs of the transitions are not stored because they are not useful to build the marking class timed automaton given that the reachability graph is already computed.

5.3.3 Loading of the reachability graph

To store the reachability graph of the underlying Petri Net which is described in the file net.rg_desc, the program calls the function ”rg load_rg(FILE *file, net_struct net)” (implemented in the file load_rg.c). This function takes as arguments the net_struct built previously and the file descriptor of the file net.rg_desc, and returns a data structure rg (implemented in structure.h). Figure 5.2 shows the form of this data structure.

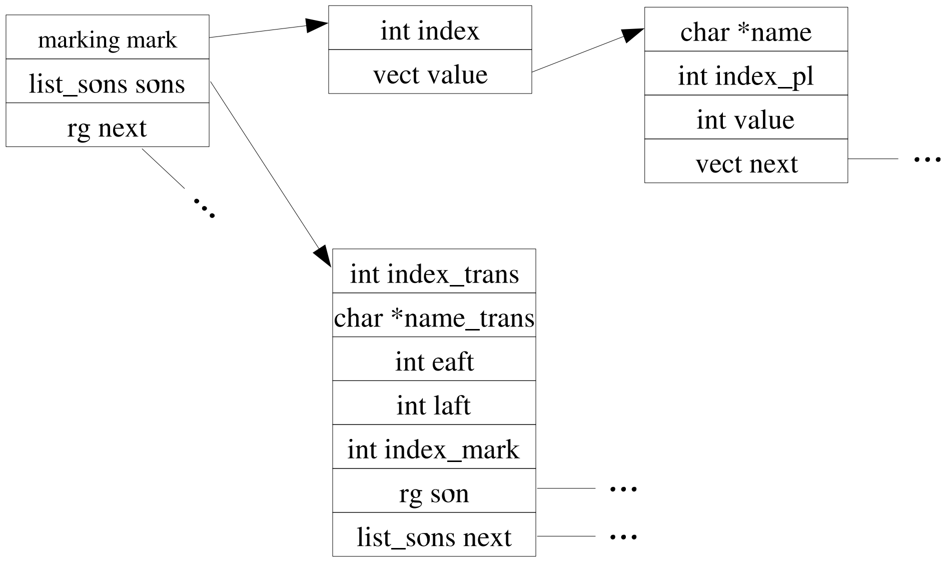


Figure 5.2: The data structure rg

A data structure rg corresponds to a list of the different markings of the reachability graph and their successors. It has the following fields :

- mark stores the marking; it is of the type marking which has the fields :
  - index stores the index of the marking;
  - value stores the value of the marking as a list of places and values (only the places that have a strictly positive number of tokens are stored); it is of the type vect which has the fields :
    - name stores the name of the place;
    - index_pl stores the index of the place;
    - value stores the number of tokens;
    - next stores the next place (NULL if there is no next element).
- sons stores the list of the sons of the marking; it is of the type list_sons which has the fields :
  - index_trans stores the index of the transition which arrives at the son;
  - name_trans stores the index of the transition which arrives at the son;
  - eaft stores the earliest firing time of the transition which arrives at the son;
  - laft stores the latest firing time of the transition which arrives at the son;
  - index_mark stores the index of the marking of the son;
  - son stores the son; it is of the type rg;
  - next stores the next son (NULL if there is no next element).
- next stores the next element of the reachability graph (NULL if there is no next element).

The function load_rg which is generated by flex (or lex) is in fact a lexical analyser of the file net.rg_desc. This function does not fill the field son of the list of sons because to build the data structure rg, it parses only once the file. In fact, to make then the link between the list of sons and the element to which they correspond the function ”update_rg(rg r)” (which is implemented in update_rg.c) is called. The goal of the field son is to mimimize the search of elements in the reachability graph. If we had not introduced this field, then the program which wanted to access the properties of a son of an element of the reachability graph would have to look for this son in the reachability graph knowing its index; instead, with the field son, the program obtains directly the pointer to the son in the reachability graph.

5.3.4 Computation of the marking class timed automaton

When the net and the reachability graph have been stored in data structures, the program calls the function ”ta build_ta_rename(net_struct net, rg r)” (which is implemented in build_ta_rename.c) for the construction of a marking class timed automaton with renaming of clocks, or the function ”ta build_ta(net_struct net, rg r)” (which is implemented in build_ta.c) for the construction of a marking class timed automaton without renaming of clocks. These two functions take as arguments the data structure which contains the description of the net and the data structure which contains the description of the reachability graph, and returns a data structure ta (implemented in structure_ta.h) which stores a timed automaton and whose form is given by the figure 5.3.

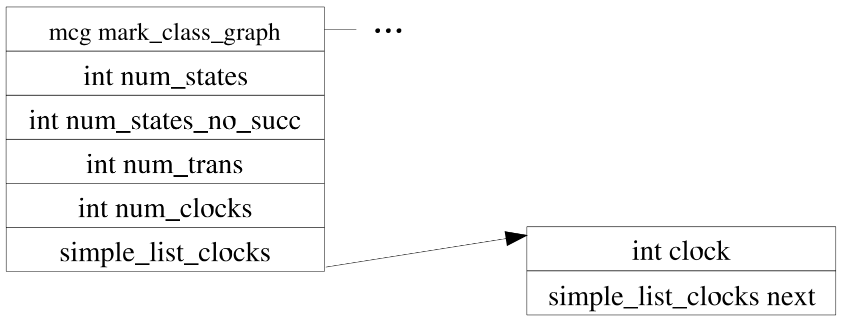


Figure 5.3: The data structure ta

The data structure ta has the following fields :

- marking_class_graph stores the marking class graph; it is of the type mcg that we will describe later in this section;
- num_states stores the number of locations;
- num_states_no_succ the number of locations with no sons;
- num_trans stores the number of switches;
- num_clocks stores the number of clocks;
- clocks stores the list of clocks; it is of the type simple_list_clocks which has the fields :
  - clock the index of the clock;
  - next the next clock in the list (NULL if there is no next element).

For Kronos each location must have a son, and the exact number of switches must be given. To solve the problem of the locations which do not have any sons, we decide to make them their own successor with a switch labelled with the name no_trans. Consequently the number of switches given to Kronos will correspond to the number of switches of the marking class timed automaton to which is added the number of locations without successor, which motivates the presence of a field which stores the number of locations without successors. In the data structure ta, the first field stores the marking class graph which is of type mcg (implemented in structure_ta.h) whose form is shown in figure 5.4.

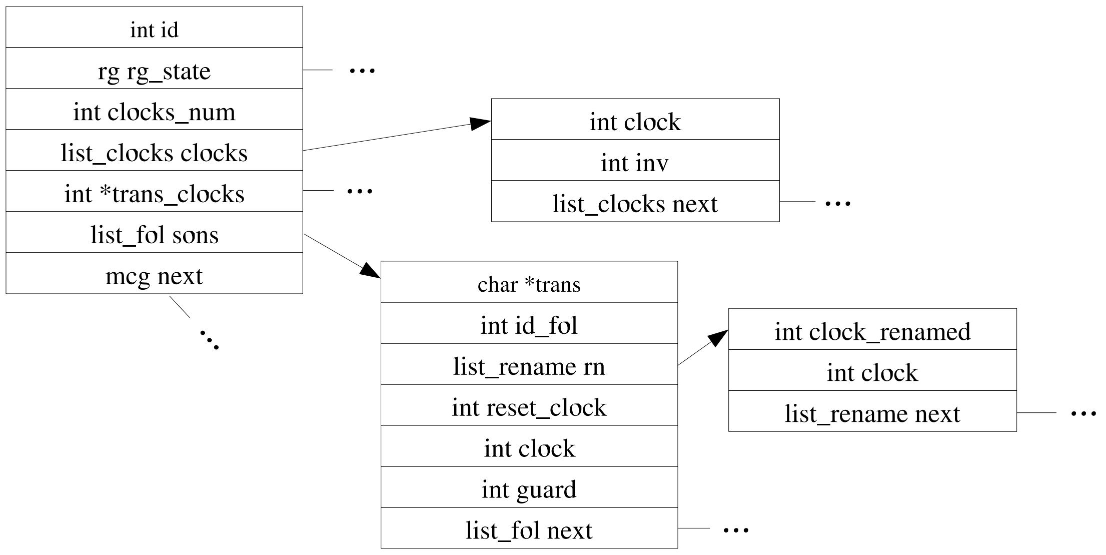


Figure 5.4: The data structure mcg

The data structure mcg is a list of marking classes and it has the following fields :

- id stores the index of the marking class;
- rg_state stores the state of the reachability graph which corresponds to the marking class; it is of type rg (see above for more informations);
- clocks_num stores the number of clocks of the marking class;
- clocks stores the list of the clocks of the marking class; it is of type list_clocks which has the following fields :
  - clock stores the index of the clocks;
  - inv stores the invariant assoiated with the clock in the location of the marking class timed automaton corresponding to the marking class;
  - next stores the next clock in the list (NULL if there is no next element).
- trans_clocks stores a table whose size is equal to the number of transitions in the net; to each enabled transition in this marking class an index of a clock is associated in the table, while for the other transitions the value in the table is equal to -2;
- sons stores the list of the sons of the marking class; it is of the type list_fol which has the following fields :
  - trans stores the name of the transition whose firing arrives at the successor;
  - id_fol stores the index of the successor;
  - rn stores the list of the renaming of clocks that are associated to the switch which leads to the successor; when the timed automaton is supposed to be without renaming of clocks this list is empty; this list is of type list_rename which has the following fields :

        * clock_renamed stores the index of the clock that will be renamed;
        * clock stores the index of the clock used for the renaming;
        * next stores the next element in the list (NULL if there is no next element).
    - reset_clock stores the clock to be reset with the firing of the transition; if there is no clock to be reset, the value stored is -1;
    - clock stores the clock associated to the transition fired;
    - guard stores the guard of the clock associated to the transition fired (obtained from the earliest firing time of the transition);
    - next stores the next successor of the marking class (NULL if there is no next element).

- next stores the next marking class (NULL if there is no next element).

The algorithm to compute the marking class timed automaton corresponds to the one given in chapter 4, except for the fact that the reachability graph is used to determine the enabled and newly enabled transitions and the value of the markings. The addition of the clocks at the automaton is done directly when the marking classes are computed. For the construction of the marking class timed automaton without renaming of clocks, instead of using the relation of bisimilarity to regroup the marking classes, the equality of marking classes is tested.

5.3.5 Writing in the file

Once the marking class timed automaton has been computed, the program calls the function ”void ta2ta_kronos(FILE *tg,ta t)” (respectively ”void ta2ta_uppaal(FILE *xml,FILE *ta_desc,ta t)”), implemented in ta2ta_kronos.c (resp. in ta2ta_uppaal.c), and which writes the timed automaton in the file net.tg (resp. in the files net.xml and net.ta_desc). These two functions take as arguments the file descriptors of the files in which they will write the timed automaton and the timed automaton itself contained in the data structure ta. Note that these functions can be easily adapted knowing the form of the data structure ta in order to produce another type of files, for instance if an user want to obtain the timed automaton in the input format of an other tool.

5.4 Comparison with Romeo

5.4.1 Presentation of the tool Romeo

The software Romeo permits the state space computation of Transititions Time Petri Nets, on-the-fly model-checking and translations from Transitions Time Petri Nets to timed automata with an equivalent behavior. It has been developed at the IRCCyN (Institut de Recherche en Communication et Cybernétique de Nantes). The software Romeo incorporates two tools in which we are interested :

1. The tool GPN;

2. The tool MERCUTIO.

GPN is a tool which computes the state class timed automaton of a Transitions Time Petri Net in the input format of Uppaal or of Kronos. MERCUTIO also computes a timed automaton whose behavior is equivalent to a given Transitions Time Petri Net, but to construct it, instead of computing the reachable markings of the Time Transitions Petri Net using the method of the extended state classes (as GPN), it uses the zone graph method we have presented in chapter 1. The exact method is given in [13] where it is also shown that MERCUTIO can be more efficient than GPN.

5.4.2 Comparison

Since both MERCUTIO and GPN compute a timed automaton with an equivalent behavior to the one of the given Time Transitions Petri Net, as our tool does, we have decided to compare these tools with the tool we have implemented. We consider two models (found in [10]) which could be easily extended to test the behavior of the different tools. The first model is a representation of the problem of the philosophers without colour. Figure 5.5 shows the i^{th} element of a model with n philosophers. The model we called philo*k* has k philosophers. The second model is a representation of the Local Area Network called the slotted ring. Figure 5.6 shows the i^{th} element of a model with n elements in the LAN. The model we call RS*k* has k elements in the LAN. It is to be noted that the two models have an underlying Petri Net which is bounded;if this was not case, it would not have been possible to test our tool with these models.

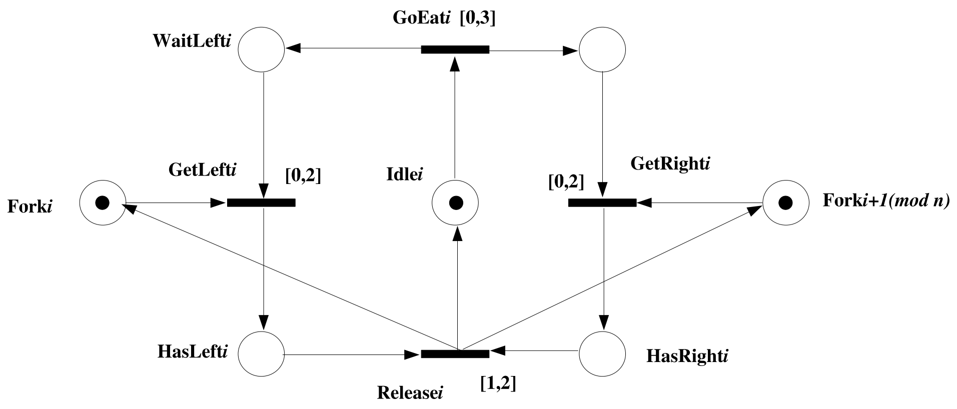


Figure 5.5: The i^{th} philosopher

The results obtained by testing the tools on these models are given the table 5.3. Due to a difficulty in obtaining Kronos models using MERCUTIO (empty files were returned to the user when MERCUTIO was tested during the preparation of this report), only the translation to Uppaal models was performed with MERCUTIO. The tests have been realized on Intel Celeron 2.60 GHz with 192 MB of RAM. The values that feature in the table 5.3 are the mean of the computed times after having run the different tools many times. When the symbol *** occurs, it signifies that the program needed more than 3 hours to

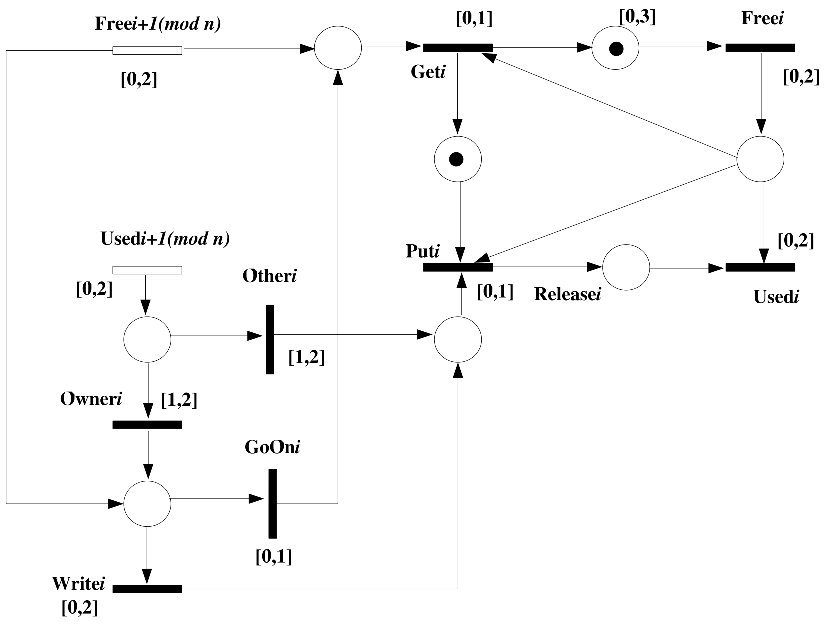


Figure 5.6: The i^{th} element in the slotted ring

conclude and that it has been stopped.

The results clearly show that the computation of the marking class timed automaton

Models	GSPN2TA (kronos)	GSPN2TA (uppaal)	GPN (kronos)	GPN (uppaal)	MERCUTIO
philo3	0.95sec	13.17sec	2.8sec	7.29sec	0.59sec
philo4	6.74sec	***	29.95sec	***	19.69sec
philo5	8min12sec	***	***	***	1h30min
RS3	0.36sec	4.06sec	0.41 sec	1.26sec	0.27sec
RS4	12min04sec	***	***	***	***

Table 5.3: Time to compute the different timed automata

using renaming of clocks can be more efficient than the other methods presented. However when the renaming of clocks is suppressed, our tool is less efficient. Even if the choice was made in Romeo to favourite the computation of timed automata without renaming of clocks, the option with renaming of clocks should not be neglicted, because it permits the reduction of the number of locations of the timed automaton. The results we obtain in this case are quite encouraging and suggest that the development of the theory of the marking class timed automaton should be continued.

Chapter 6

Conclusions and perspectives for the Marking Class Timed Automaton

6.1 Introduction

In this chapter, we comment on the method of the marking class timed automaton for TTWN presented previously and we consider a possible extensions of the symbolic approach developed for Well-formed Nets.

6.2 Efficiency of the method of the Marking Class Timed Automaton

6.2.1 Conclusions on the Marking Class Timed Automaton

In this report, we presented the method of the marking class timed automaton which permits the construction of a timed automaton which has an equivalent behavior to that of a Transitions Time Petri Net. There exist other methods to compute such a timed automaton, such as the method based on the extended state class graph or that based on the zone graph. For these two methods, it appears that the algorithms developed use reachability techniques which will be again used to analyse the produced timed automata. Consequently, for these two methods during the building of the timed automata and during its analysis similar computations are performed. Instead, the method we proposed uses only the reachability graph of the underlying Petri Net to build the marking class timed automaton. It is true that our method may produce a timed automaton with more locations, but when the introduced time constraints do not modify the behavior of the underlying Net, the timed automaton produced is the same as the one obtained with the other two methods. Hence, since the method we proposed works only for Transitions Time Petri Nets (resp. TTWN) which have a bounded underlying Petri Net (resp. Well-formed Net), it can be used to find real-time properties for models that already have been modelled without time constraints and on which a part of verification (construction of the reachability graph) has already been done.

6.2.2 Improvement of the tool

We have implemented a tool which can compute the marking class timed automaton and in comparison to other tools which also construct timed automata from Transitions Time Petri Nets, our tool gave encouraging results. These results are encouraging anf can be improved. In fact, we observed that our tool, for some examples, was taking z realtively large amount of time to build the marking class timed automaton even though the reachability graph was already computed. This problem is linked to the data structures which store the elements; in fact in the phase of building the timed automaton from the reachability graph almost no calculations are performed. This could be improved using more compact data structures such as BDDs or by improving the presented data structures in order to get better time performace. In order to deal with some Transitions Time Petri Nets which are not bounded, a possible improvement could be to adapt a method developed for the non-bounded Petri Net which consists in replacing an infinite sequence of growing markings by a special marking [15].

6.3 Toward a Symbolic Marking Class Timed Automaton

If at the beginning of the theory of the TTWN, we decided to add time constraints to the transitions of the Well-formed Nets and not to all the ordinary coloured nets, it was to ensure the fact that we will then be able to use the symmetric properties of a Well-formed Net. In this section, we will begin by defining the symmetries of a TTWN and by oresenting how they determine sets of equivalent marking classes; we will then present an adaptation of the algorithm of Huber et al. [17] and, finally we consider the problems raised by the adaptation of the symbolic approach to TTWN. In this section, we use definitions which were presented in chapter 2 and in [8] .

6.3.1 Equivalence between the Marking Classes

Symmetries

We consider a TTWN $\mathcal{T} = \langle P, T, W^-, W^+, Cl, \mathcal{C}, \Phi, (\alpha, \beta), m_0 \rangle$ with the set of colour classes $Cl = \{C_1, ..., C_k\}$. For all $i \in [|1,k|]$ the colour class C_i is divided in n_i static subclasses (ie $C_i = \biguplus_{q=1,\ldots,n_i} D_{i,q}$).

Definition 31 *A permutation $s = \otimes_{i=1}^{n} s_i$ on $\otimes_{i=1}^{k} C_i$ is a symmetry on the TTWN $\mathcal{T}$ if and only if :*

- *If C_i is not ordered, s_i is a permutation on C_i such that $\forall q \in [|1,n_i|]$, $s_i(D_{i,q}) = D_{i,q}$;*
- *If C_i is ordered, s_i is a rotation on C_i such that $\forall q \in [|1,n_i|]$, $s_i(D_{i,q}) = D_{i,q}$ (this implies that if $n_i > 1$ then the only allowed rotation is the identity).*

We will denote ξ the set of the symmetries on a net $\mathcal{T}$. For all colour domains $C_d = \otimes_{i=1}^{k}(C_i)^{e_i}$, $\forall c \in C_d$, $s(c)$ is defined by : $s(\otimes_{i=1}^{k} \otimes_{j=1}^{e_i} c_i^j) = \otimes_{i=1}^{k} \otimes_{j=1}^{e_i} s_i(c_i^j)$.

Directly from this definition and from the definitions of a permutation and of a rotation, we can deduce that $(\xi, \circ)$ is a group which has for neutral element the identity function.

Definition 32 *We consider a marking M of the TTWN $\mathcal{T}$ and a symmetry s on $\mathcal{T}$. Then the marking $s.M$ is defined by :*

$$\forall p \in P, \forall c \in \mathcal{C}(p), s.M(p)(c) = M(p)(s(c))$$

We then have the following property.

Property 1 *The application of an admissible symmetry to a marking and to the colour instance of transition preserves the enabling of the transition .For all transitions $t \in T$, for all colours c of $\mathcal{C}(t)$, and for all symmetries s of ξ, the transition t is enabled for the colour c in the marking M if and only if the transition t is enabled for the colour $s(c)$ in the marking $s.M$. Furthermore, M' is the marking obtained from the firing of the enabled transition t for the colour $c \in \mathcal{C}(t)$ from the marking M if and only if $s.M'$ is the marking obtained from the firing of the enabled transition t for the colour $s(c)$ from the marking $s.M$.*

Proof This result directly comes from the theory of the Well-formed Nets ([22],[9]).
□
We also need another definition to use symmetries.

Definition 33 *Consider $\mathcal{CT} = \{(t,c) | t \in T \wedge c \in \mathcal{C}(t)\}$. For all symmetries $s \in \xi$, for all subsets U of $\mathcal{CT}$, the subset $s.U$ is defined by :*

$$(t,c) \in U \Leftrightarrow (t, s(c)) \in s.U$$

Equivalence relation

We can then define an equivalence relation for the marking classes.

Definition 34 *Consider two marking classes $Mc = \langle M, \chi, trans \rangle$ and $Mc' = \langle M', \chi', trans' \rangle$. Mc and Mc' are equivalent for the equivalence relation $\cong_{mc}$, denoted by $Mc \cong_{mc} Mc'$, if and only if :*

$$\exists s \in \chi \text{ such that } \begin{cases} M' = s.M \text{ and} \\ \chi = \chi' \text{ and} \\ \forall x \in \chi, trans(x) = s.trans(x') \end{cases}$$

The equivalence classes of this relation are called the symbolic marking classes. For each marking class Mc, we will denote by $\hat{M}c$ its symbolic marking class.

We want then to use this equivalence relation to build a timed automaton with fewer locations than the marking class timed automaton, ideally with one location for each symbolic marking class which features in the marking class timed automaton.

6.3.2 The algorithm of Huber et al. and the problem it raises

The algorithm of Huber et al. as presented in [9] allows the representation of each symbolic marking by one of its elements and determines if two elements are equivalent. The test for the equivalence of markings consists of an exhaustive search for a symmetry mapping one marking class onto the other. We will call the graph built with this algorithm

Equivalent Marking Class Graph (EMCG). To begin the construction of this graph, we need the initial marking class Mc_0 as it is defined in chapter 4 . The algorithm to build this graph can be written as follows :

```
• The variables are $EMCG$ (the equivalent marking class graph itself), a FIFO queue
  $New$ (which contains the newly computed marking classes) and a boolean $New\_found$
  (to know if a newly equivalent class has been reached);
• The variables are initialized to : $EMSG := \{Mc_0\}$ and $New := Mc_0$;
• WHILE $New$ is NOT empty DO
  – $C := remove(New)$ (The first marking class of $New$ is taken, $C := \langle M_C, \chi_C, trans_C \rangle$);
  – For all pair $(t_i, c_{i,j}) \in \mathcal{CT}$ such that $t_i$ is enabled for $c_{i,j}$ in $M_C$ (i.e. $\forall p \in
    P, M_C(p) \geq W^-(p,t_i)(c_{i,j})$ and $\Phi(t_i)(c_{i,j}) = True$) DO
    ∗ (Computation of the marking class $C' = \langle M_{C'}, \chi_{C'}, trans_{C'} \rangle$, son of $C$ )
    ∗ For all $p \in P, M_{C'}(p) = M_C(p) + W^+(p,t_i)(c_{i,j}) - W^-(p,t_i)(c_{i,j})$;
    ∗ Computation of $\chi_{C'}$ and $trans_{C'}$ :
      1. For each clock $x \in \chi_C$, remove from $trans_C(x)$ all the pairs $(t_k, c_{k,l})$
         such that $t_k$ is enabled for $c_{k,l}$ in $M_C$ and is not for the marking $M'$ de-
         fined by $\forall p \in P, M'(p) = M_C(p) - W^-(p,t_i)(c_{i,j})$, to obtain a relation
         $trans'$;
      2. The clocks whose image by $trans'$ is empty are removed from $\chi_C$, to
         obtain a set of clocks $\chi'$;
      3. For all pairs $(t_k, c_{k,l}) \in \mathcal{CT}$ which verify $\uparrow enabled((t_k, c_{k,l}), M_C, (t_i, c_{i,j})) =
         True$ DO :
         + IF a clock $x$ has already been created for computing $C'$
         + THEN $(t_k, c_{k,l})$ is added to $trans'(x)$;
         + ELSE a new clock $x_n$ is created; $n$ is the smallest available index
           among the clocks of $\chi'$ and $trans'(x_n) = (t_k, c_{k,l})$;
      4. ENDDO; $\chi_{C'}$ and $trans_{C'}$ are then obtained from the resulting set $\chi'$
         and function $trans'$;
    ∗ $New\_found := true$
    ∗ For all $s \in \xi$ DO
      + $C'' := \langle s.M_{C'}, \chi_{C'}, s.trans_{C'} \rangle$
      + IF $C'' \in EMSG$
      + THEN $New\_found := false$ and Goto Cont
      + ENDIF
    ∗ ENDDO
    ∗ Cont:
    ∗ IF $New\_found = True$
    ∗ THEN $add(New, C')$
    ∗ ENDIF
```

```
        * $EMCG := EMCG \cup \{C \overset{(t_i, c_{i,j})}{\rightarrow}_{mc} C'\}$
    – ENDDO;
• ENDDO.
```

We note that this algorithm does not take into account the clock-similarity of the marking classes. A consequence of this is that the timed automaton which would be built from such a graph would not include renamings of clocks. It also appears, that the construction of the timed automaton from this graph following the rules evocated in chapter 4 would give an automaton with locations which will not have successors if their marking class is equivalent to another one in the net. The analysis of this timed automaton will also be more complicated because of these locations.We do not explore this method further for the same reasons that motivated the developement of the theory of the Symbolic Reachability Graph for Well-formed Nets. In fact, the equivalence test is costly in terms of time, because to test if an equivalent marking class is equivalent to another one, we have to search among all the symmetries, one which permits us to conclude that there is equivalence.

6.3.3 The symbolic theory and the problem it raises

For the Well-formed Nets, the theory of the symbolic reachability marking graph solves the different problems described above. Firstly, each symbolic marking is not represented by one of its elements, but it is given in a symbolic form, from which there exists a canonical form so that the test of equality of two marking classes of equivalence which are equal can be done easily. Secondly, in this theory, there is no need to test the firing of a transition for all the colours of its domain because the firing is done symbolically, which leads to a saving in terms on time.

A possible representation of a symbolic marking

If we want to extend this theory to the case of TTWN, the first thing that needs to be done is to define a way of representing symbolic marking classes. In our case, we use the same notation as for the Well-formed Nets ([8], [22]), to which we add a part for the relation $trans$.

Definition 35 *A symbolic representation of a symbolic marking $\hat{M}c$ is a 6-tuple $\tilde{M}c = \langle\{\{Z_i^j\}_{j=1,\dots,\hat{n}_i}\}_{i=1,\dots,k}, St, Card, Marq, \chi, \mathcal{TR}\rangle$ where :*

- *$\{Z_i^j\}_{j=1,\dots,\hat{n}_i}$ is a set of dynamic subclasses associated to C_i; we denote $\hat{C}_i = \{Z_i^j\}_{j=1,\dots,\hat{n}_i}$ and, by extension, if $C_d = \otimes_{i=1}^{k}(C_i)^{e_i}$ is a domain colour, then $\hat{C}_d = \otimes_{i=1}^{k}(\hat{C}_i)^{e_i}$;*
- *St is a function which maps to each dynamic subclass Z_i^j a static subclass $C_{i,q}$ of C_i;*
- *$Card$ is a function which maps each dynamic subclass with a positive integer (which represents its cardinality) such that:*

$$\Sigma_{St(Z_i^j)=C_{i,q}} Card(Z_i^j) = |C_{i,q}|$$

- *Marq is a function which defines the distribution of the dynamic subclasses in the places of the net: for all $p \in P$, we have $Marq(p) \in Bag(\hat{C}(p))$;*
- *χ is a set of indexed clocks;*
- *$\mathcal{TR}$ is a function which associates to each clock of χ a subset of $\{(t,\hat{c}) | t \in T \wedge \hat{c} \in \hat{C}(t)\}$, representing the distribution of the dynamic subclasses for the enabled transition taking account of the clocks.*

A symbolic representation represents a set of ordinary marking classes.

Definition 36 *We consider a symbolic representation $\tilde{M}c = \langle \{\{Z_i^j\}_{j=1,\dots,\hat{n}_i}\}_{i=1,\dots,k}, St, Card, Marq, \chi, \mathcal{TR}\rangle$. A marking class $Mc = \langle M, \chi', trans\rangle$ is represented by $\tilde{M}c$ if and only if there exists, for all colour classes C_i, a function Ψ_i: $C_i \rightarrow \hat{C}_i$, such that:*

1. *The number of colours represented by a dynamic subclass is equal to its cardinality :*
$$|\Psi_i^{-1}(Z_i^j)| = Card(Z_i^j)$$
2. *The set of colours represented by a dynamic subclass is included in only one static subclass :*
$$\forall q \in [|1, n_i|], \forall Z_i^j \in \hat{C}_i, St(Z_i^j) = q \Rightarrow \Psi_i^{-1}(Z_i^j) \subseteq C_{i,q}$$
3. *If C_i is an ordered class, then the order on C_i is respected, which means that there exists $c_h \in \Psi_i^{-1}(Z_i^j)$ such that:*
$$\oplus c_h \in \Psi_i^{-1}(Z_i^{(j+1) mod |\hat{C}_i|})$$
$$\forall c \neq c_h \in \Psi_i^{-1}(Z_i^j), \oplus c \in \Psi_i^{-1}(Z_i^j)$$
4. *The set of clocks are the same,i.e. :*
$$\chi = \chi'$$
5. *The colours are distributed according to the distribution of the dynamic subclasses :*
   - *$\forall p \in P$, $\forall \otimes_{i=1}^k \otimes_{j=1}^{e_i} c_i^j \in \mathcal{C}(p)$ we have :*
   $$M(p)(\otimes_{i=1}^k \otimes_{j=1}^{e_i} c_i^j) = Marq(p)(\otimes_{i=1}^k \otimes_{j=1}^{e_i} \Psi_i(c_i^j))$$
   - *$\forall x \in \chi$, $\forall (t, \otimes_{i=1}^k \otimes_{j=1}^{e_i} c_i^j) \in \mathcal{CT}$ we have :*
   $$(t, \otimes_{i=1}^k \otimes_{j=1}^{e_i} c_i^j) \in trans(x) \Leftrightarrow (t, \otimes_{i=1}^k \otimes_{j=1}^{e_i} \Psi_i(c_i^j)) \in \mathcal{TR}(x)$$

The conditions on the functions Ψ_i ensure that the marking classes represented by a symbolic representation are exactly the elements of the corresponding symbolic markings. We have the following property for this representation.

Property 2 *Let Mc be a marking class. There exists at least one symbolic representation $\tilde{M}c$ of Mc, and every symbolic representation of Mc represents exactly $\hat{M}c$.*

Proof We consider a marking class Mc. We can at least build the symbolic representation $\tilde{M}c$ where each dynamic subclass has a cardinality equal to 1, where we associate to each dynamic subclass Z_i^j a different element of the colour class C_i. The construction of the functions *St* and *Marq* are then the exact representation of the distribution of each colour. As noted in [22], the conditions on the different functions Ψ_i are linked to the conditions on the possible symmetries (in fact, a symmetry respects the static subclasses and the order of the ordered classes). This in combination with Property, which referred to the enabling of the transitions (and consequently the construction of the function $\mathcal{TR}$) allows us to say that the marking classes represented by a symbolic marking class $\tilde{M}c$ are exactly the marking classes of $\hat{M}c$.
□

Problems raised

First, we would like to note that the adaptation of the symbolic marking and its representation we have done in the precedent parts could be done in a more convenient way. We chose this adaptation because it is a natural extension of the theory developed for Well-formed Nets. Nevertheless some problems appear which prevented further development of the symbolic representation and which show that another representation could be better.

The first problem concerns the construction of a canonical representation of our symbolic representation. In fact, there could be different symbolic representations for a same symbolic marking class. It is useful to have a canonical form for testing equality of two symbolic markings in addition to an algorithm to build this canonical form from any symbolic representations.

The second problem concerns the building of the set $\mathcal{TR}$ and the symbolic firing of transitions. In fact, if for the symbolic firing in Well-formed Nets ([22],[8]), to see if a transition is enabled, the symbolic markings are instantiated in order to allow an evaluation of the colour functions and predicates for the dynamical subclasses. This instantiation depends on the symbolic marking for which the evaluation is done, and there is no link between the different instantiations of the different symbolic markings. This inconvenient for us, because to build the different sets $\mathcal{TR}$ we need the history of the enabled transitions. To illustrate this, we consider the underlying Well-formed Net of the example shown in figure 6.1 and its symbolic reachability graph which is briefly described in [22]
.

The initial symbolic marking $\hat{m}_0$ can be represented by

$$\tilde{m}_0 = (Z_1^1).Sready \text{ with } |Z_1^1| = 3$$

(for the examples, we adopt simple denotations to simplify). Then the transition *csend* is firable for the instantiation Ins_1 of $\tilde{m}_0$ which can be described by :

$$Ins_1.\tilde{m}_0 = (Z_1^{1,1} + Z_1^{1,2} + Z_1^{1,0}).Sready$$

$$\text{with } |Z_1^{1,1}| = |Z_1^{1,2}| = |Z_1^{1,0}| = 1$$

The firing of this transition for this instantiation, which associates x to $Z_1^{1,1}$ and y to $Z_1^{1,2}$ generates the symbolic marking $\hat{m}_1$ which can be represented by :

$$\tilde{m}_1 = (Z_1^1 + Z_1^2).Sready + (Z_1^3).Cwait + (\langle Z_1^3, Z_1^1 \rangle).Mess$$

$$\text{with } |Z_1^1| = |Z_1^2| = |Z_1^3| = 1$$

From this symbolic marking, the transition *csend* is again firable for instance for the instantiations of $\tilde{m}_1$ denoted by Ins_1 and Ins_2.

$$Ins_1.\tilde{m}_1 = (Z_1^{1,1} + Z_1^{2,1}).Sready + (Z_1^{3,0}).Cwait + (\langle Z_1^{3,0}, Z_1^{1,1} \rangle).Mess$$

$$\text{with } |Z_1^{1,1}| = |Z_1^{2,1}| = |Z_1^{3,0}| = 1$$

and :

$$Ins_2.\tilde{m}_1 = (Z_1^{1,1} + Z_1^{2,0}).Sready + (Z_1^{3,1}).Cwait + (\langle Z_1^{3,1}, Z_1^{1,1} \rangle).Mess$$

$$\text{with } |Z_1^{1,1}| = |Z_1^{2,0}| = |Z_1^{3,1}| = 1$$

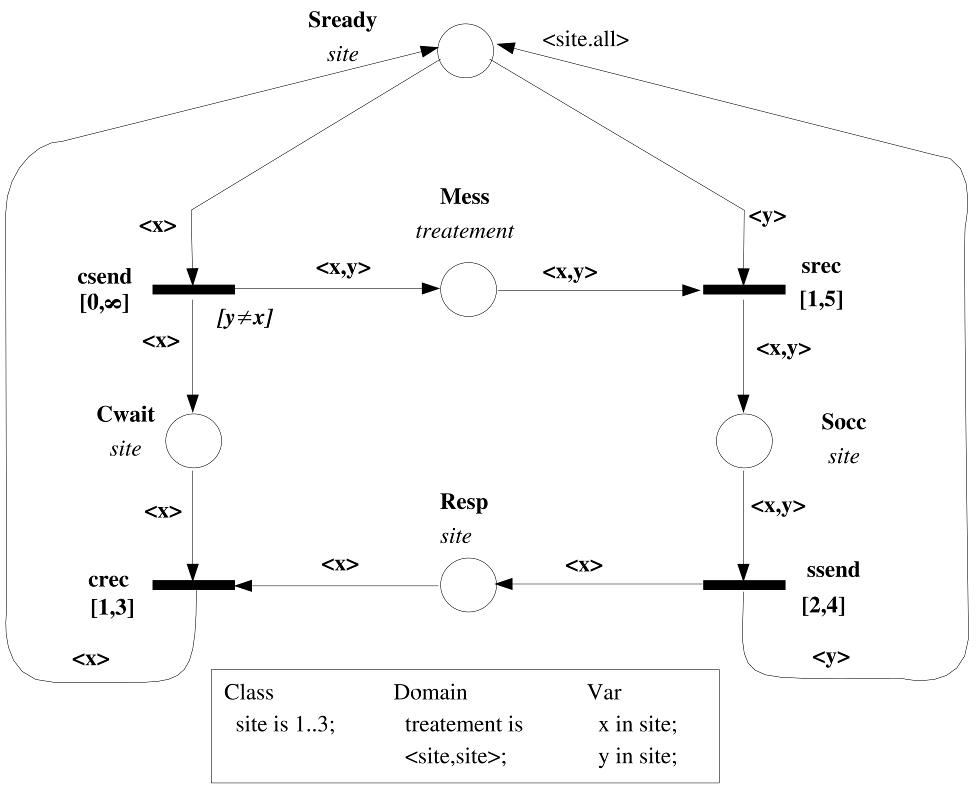


Figure 6.1: AN example of TTWN

For Ins_1, x is associated to $Z_1^{1,1}$ and y to $Z_1^{2,1}$, and for Ins_2, x is associated to $Z_1^{1,1}$ and y to $Z_1^{3,1}$. This part of the symbolic reachability graph of the underlying Well-formed Net is represented on the figure 6.2. This example clearly shows that it is difficult to make a relation between the enabling of *cenv* from $\tilde{m}_0$ and the one from $\tilde{m}_1$. The firing of *cenv*

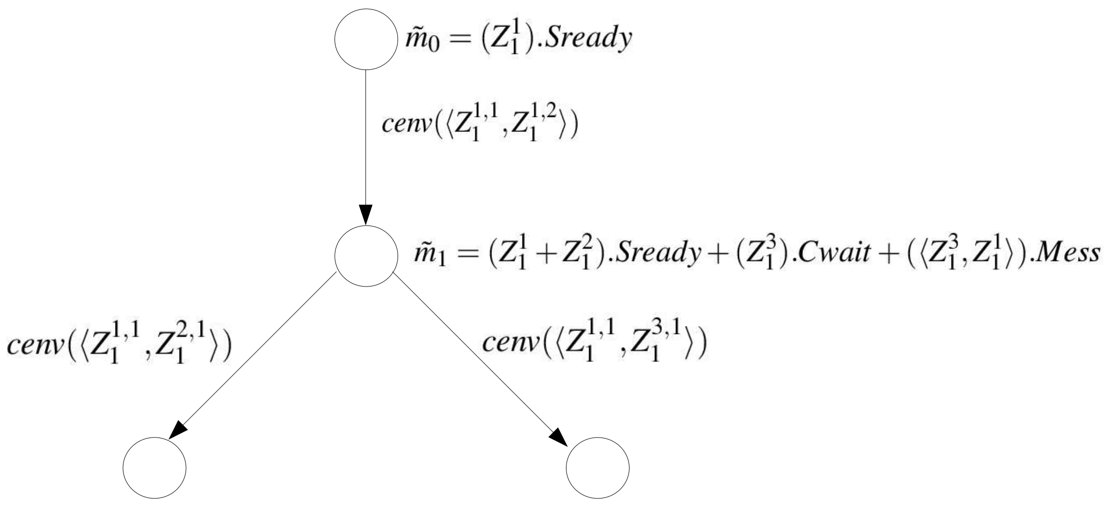

$\tilde{m}_0 = (Z_1^1).Sready$
$cenv(\langle Z_1^{1,1}, Z_1^{1,2}\rangle)$
$\tilde{m}_1 = (Z_1^1 + Z_1^2).Sready + (Z_1^3).Cwait + (\langle Z_1^3, Z_1^1\rangle).Mess$
$cenv(\langle Z_1^{1,1}, Z_1^{2,1}\rangle)$
$cenv(\langle Z_1^{1,1}, Z_1^{3,1}\rangle)$

Figure 6.2: Part of the SRG of the underlying Well-formed net

from $\tilde{m}_1$ was already enabled from $\tilde{m}_0$; this information that does not appear (or is difficult to be extracted) from the symbolic reachability graph is needed to analyse the behavior of the TTWN.

If we take the symbolic representation that we have previously described, the initial symbolic marking class $\hat{M}c_0$ of the TTWN presented in figure 6.1 could be represented by the symbolic representation $\tilde{M}c_0$:

$$\tilde{M}c_0 = (Z_1^1 + Z_1^2 + Z_1^3).Sready$$

$$\text{with } |Z_1^{1,1}| = |Z_1^{2,0}| = |Z_1^{3,1}| = 1, \chi = \{x_0\}$$

and

$$\begin{aligned}\mathcal{TR}(x_0) = \{&(csend, \langle Z_1^1, Z_1^2\rangle), (csend, \langle Z_1^1, Z_1^3\rangle),\\ &(csend, \langle Z_1^2, Z_1^1\rangle), (csend, \langle Z_1^3, Z_1^1\rangle),\\ &(csend, \langle Z_1^2, Z_1^3\rangle), (csend, \langle Z_1^3, Z_1^2\rangle)\}\end{aligned}$$

In this representation, instead of having one representation of enabling of transition, we have six. If we fire "symbolically" this transition for each of the enabled transition then the algorithm is the same as the one of Huber et al.. If we find a canonical form, we do not have to look anymore for all the symmetries but we could test equality only on the canonical form. The symbolic representation of this initial symbolic marking class cannot be reduced, in the sense that it is not possible to represent the symbolic marking with fewer dynamic subclasses. If look again at the symbolic reachability graph of the underlying Well-formed Net, we note that it would be better to have a symbolic representation $\tilde{M}c_0$ of the form :

$$\tilde{M}c_0 = (Z_1^1).Spret$$

$$\text{with } |Z_1^{1,1}| = 3, \chi = \{x_0\}$$

$$\text{and } \mathcal{TR}(x_0) = \{(csend, \langle Z_1^{1,1}, Z_1^{1,2}\rangle)\}$$

However, the problem that we encounter here is how to define formally this symbolic representation and as as we have observed before, how to retain an history of the enabled transitions between the different symbolic markings.

Conclusions on the symbolic theory

In conclusion, we have seen that to adapt the symbolic theory developed for the Well-formed Nets to the construction of a marking class timed automaton for TTWNs, it is necessary to redefine a symbolic representation of the symbolic marking classes to allow a symbolic firing similar to the one found for the Well-formed Nets and which would also allow the construction of the sets $\mathcal{TR}$, which are necessary for the construction of the timed automaton. For the theory to be complete, it is also necessary to define a canonical form of this symbolic representation and an algorithm to construct it.

Bibliography

[1] R. Alur. Timed automata. In *CAV '99: Proceedings of the 11th International Conference on Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*, pages 8–22. Springer-Verlag, 1999.

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[3] G. Behrmann, A. David, and K. G. Larsen. A tutorial on UPPAAL. In *Formal Methods for the Design of Real-Time Sytems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer-Verlag, 2004.

[4] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. Softw. Eng.*, 17(3):259–273, 1991.

[5] P. Bouyer. Real-time and hybrid systems. Presentation Winter School MOVEP'04.

[6] P. Bouyer. Untameable timed automata! In *STACS'03: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2607 of *Lecture Notes in Computer Science*, pages 620–631, 2003.

[7] F. Cassez and O. H. Roux. Structural translations of time Petri nets into timed automata. In *AVOCS'04: Proceedings of the 4th International Workshop on Automated Verification of Critical Systems*, Electronic notes in computer science. Elsevier, 2004.

[8] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. On well-formed coloured nets and their symbolic reachability graph. In *Proc.* 11^{th} *International Conference on Apllication and Theory of Petri Nets*, June 1990. Reprinted in *High-level Petri nets : theory and application*, K. Jensen and G. Rozenberg (editors), Springer Verlag, 1991.

[9] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. A symbolic reachability graph for coloured Petri nets. *Theor. Comput. Sci.*, 176(1-2):39–65, 1997.

[10] G. Ciardo, G. Lüttgen, and R. Siminiceanu. Efficient symbolic state-space construction for asynchronous systems. In *Proceedings of the 21st International Conference on Applications and Theory of Petri Nets*, volume 1825 of *Lecture Notes in Computer Science*, pages 103–122, 2000.

[11] 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*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer-Verlag, 1990.

[12] 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'03: Proceedings of the First International Workshop on Formal Modeling and Analysis of Timed Systems*, volume 2791, pages 246–259, 2003.

[13] G. Gardey, O. H. Roux, and O. F. Roux. State space computation and analysis of time Petri nets. *Theory and Practice of Logic Programming (TPLP). Special Issue on Specification Analysis and Verification of Reactive Systems*, 2005. To appear.

[14] P. E. group Dipartemento di Informatica Universita di Torino. GreatSPN - user's manual (version 2.0.2). User's manual of the software GreatSPN.

[15] S. Haddad. Décidabilité et complexité des problèmes de réseaux de petri. In M. Diaz, editor, *Les réseaux de Petri — Tome 1*, chapter 4, pages 119–158. Hermès, 2001.

[16] S. Haddad and F. Vernadat. Méthodes d'analyse des réseaux de Petri. In M. Diaz, editor, *Les réseaux de Petri — Tome 1*. Hermès, 2001.

[17] P. Huber, A. M. Jensen, L. O. Jepsen, and K. Jensen. Towards reachability trees for high-level petri nets. *Lecture Notes in Computer Science: Advances in Petri Nets 1984*, 188:215–233, 1985. NewsletterInfo: 27.

[18] K. Jensen and G. Rozenberg, editors. *High-level Petri nets: theory and application*. Springer-Verlag, London, UK, 1991.

[19] R. M. Karp and R. E. Miller. Parallel program schemata. *J. Comput. Syst. Sci.*, 3(2):147–195, 1969.

[20] D. Lime and O. H. Roux. Computing the state class time automaton of a transition-time Petri net. Technical Report IRCCyN number 1497 R2003_0, Institut de Recherche en Communication et Cybernétique de Nantes (IRCCyN), 2003.

[21] D. Lime and O. H. Roux. State class timed automaton of a time Petri net. In *PNPM'03: Proceedings of the 10th International Workshop on Petri Nets and Performance Models*, pages 124–133. IEEE Computer Society, Sept. 2003.

[22] F. Pradat-Peyre. Analyse des réseaux de Petri de haut niveau. In M. Diaz, editor, *Les réseaux de Petri — Tome 1*, chapter 8, pages 255–295. Hermès, 2001.

[23] J.-F. Pradat-Peyre and I. Mounier-Vernier. Introduction aux réseaux de Petri et aux réseaux de haut-niveau. Lection notes.

[24] S. Yovine. Kronos: a verification tool for real-time system. *International Journal on Software Tools for Technology Transfer (STTT)*, 1997.

[25] S. Yovine. Model checking timed automata. In *Lectures on Embedded Systems, European Educational Forum, School on Embedded Systems*, volume 1494, pages 114–152. Springer-Verlag, 1998.