

# STATE CLASS GRAPH FOR FUZZY TIME PETRI NETS

J. Cardoso<sup>1</sup>, Xiaoyu Mao<sup>2</sup> and Robert Valette<sup>3</sup>

<sup>1</sup> IRT-UT1, 21, allée de Brienne, 31042 Toulouse France

<sup>2</sup> Almende / University of Maastricht, The Netherlands

<sup>3</sup> LAAS-CNRS, 31077 Toulouse, France

e-mail: jcardoso@univ-tlse1.fr, xmao@almende.com, robert@laas.fr

## KEYWORDS

Petri nets, time Petri nets, analysis, fuzzy Petri nets

## ABSTRACT

The objective of this paper is the formal verification of quantitative temporal properties of embedded systems. Our approach is different from the current state of the art where the verification of the properties is done after a complete design with numerical values for all parameters. The application domain is that of concurrent engineering design, where essential parameters may be ill-known. The formal mathematical framework used to take into account these ill-known constraints is the fuzzy set theory. The use of fuzzy theory for temporal constraints and Petri net theory for the dynamic aspect in the construction of a graph of states classes allows ensuring some properties during design phases where the value of some parameters is not well defined.

## INTRODUCTION

Property verification is typically based on model checking. When continuous time is involved, the number of states is infinite and it is necessary to cover them by means of a finite number of state classes.

We presented in (Mao 05) the tool GraphC that generates, for t-Time Petri Nets where temporal intervals are associated with transitions (see figure 1.a), a graph of classes as the one in figure 5. The state space exactly (in a quantitative way) defines the set of constraints that have to be verified by the transition firings (event occurrences in the real system). The graph of classes is constructed in such a way that the constraints which have to be verified by the occurrence dates for any event sequence in the real system are directly derived by concatenating the constraints attached to the arcs covered by the corresponding sequence in the graph of classes. Each path in this graph is associated with a sequence which can effectively be fired in the Petri net and which verifies the temporal constraints. It must be underlined that this approach allows the analysis of properties such that, for example, “a state is reachable”. But above all, it allows determining the set of temporal constraints the sequence leading to this state must meet.

The problem with using only imprecise temporal parameters delimited by intervals  $[a, b]$ , is that we can only say

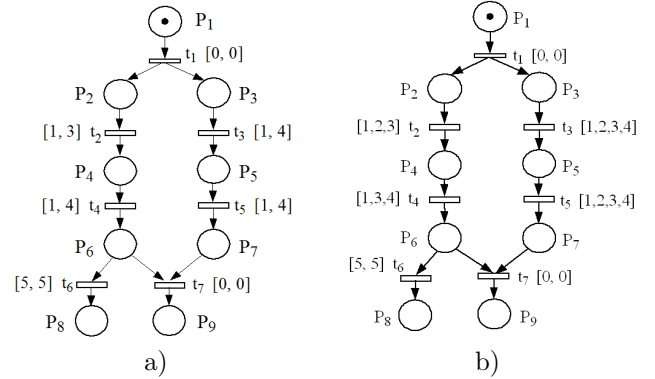


Figure 1: a) t-time and b) fuzzy time Petri nets

Yes or No for a property verification. We would like to say not only *Yes* or *No* but also “*It is Possible with a possibility measure*” i.e. with a *quantitative* view.

This kind of information can be obtained by using a Fuzzy Time Petri Net (Cardoso 98) like the one represented in the figure 1.b, where a temporal constraint is represented by a fuzzy interval of time – representing a possibility distribution of a firing date associated with each transition. For example, the fuzzy interval associated with  $t_3$  is  $[1\ 2\ 3\ 4]$ . It means that all instants of time between the *core*  $[2\ 3]$  are the best candidates to the firing interval of  $t_3$ , but the ones between the *support*  $[1\ 4]$  are not excluded.

Possibility theory is a very straightforward theory to manage incomplete information; it is particularly useful when there is no available statistical data. In this case, models requiring statistical data, as Stochastic Petri nets or Markov chains, become meaningless. It is more interesting to deal with available data (even if ill-known) than trying to use information that does not exist. In fact, if some verification can be done with values that are ill known at an earlier design phase, it allows refining temporal constraints and providing in this way flexibility in the achievement of goals in the system.

For example, let us consider reconfiguration policies concerning computer and buses. Such systems are used to control critical equipments in planes or cars. The correctness of their design depends on the duration of the cycle time of each equipment. Proving the properties in the proposed graph, obtained from a fuzzy time

Petri net, allows delaying the moment at which a precise (or less imprecise) quantitative value is assigned to some parameters.

A first attempt to do this is (Cardoso 05), where the generated graph of a Fuzzy time Petri Net was an extension of the graph of classes in linear mode (Berthomieu 04) that allows LTL property model checking. The first difference in relation to this work is the kind of the abstract class. The one used in this work is based on (Mao 05) where each path in the state space is associated with a sequence effectively fireable in the Petri net. The second difference is that in (Cardoso 05) the graph was generated directly by calculating the Possibility and Necessity measures of the transition firing using the fuzzy intervals associated with the transitions. This implies intersections and subtractions between fuzzy sets, and sometimes the resulting fuzzy sets are not trapezoids and it is necessary to approximate them by trapezoids in order to have a feasible calculus. The approach proposed in this paper consists in transforming a Fuzzy time Petri Net into several t-Time Petri Nets using the concept of  $\alpha$ -cut. In fact, a fuzzy set can be considered as a collection of nested (classical) sets called  $\alpha$ -cuts. For each t-Time Petri Net a graph of classes is generated using the tool GraphC. The issue is then to compose all these graphs into a Fuzzy Graph of Classes, using an algorithm presented in the paper. The paper is organised as follows: the first Section introduces the basic concepts needed in the proposed approach, as fuzzy dates,  $\alpha$ -cut and fuzzy Petri nets. The second Section presents a brief description of GraphC and introduces an illustrative example. The third Section explains how the GraphC and the fuzzy intervals are combined into a Fuzzy GraphC and the advantage of using alpha cuts. Finally, the last Section presents some conclusions and future work.

## BASIC NOTIONS

In this section the main notions used in the proposed approach are presented.

### Fuzzy date

A date  $a$  has *only one value*, which may be ill-known and the fuzzy set of its possible values is a disjunctive set (Yager 84). The available knowledge about a date  $a$  is represented by a possibility distribution  $\pi_a(\tau)$ ,  $\tau \in \mathcal{T}$  (the universal set of time instants), delimited by the fuzzy set  $A$ , represented by the quadruple  $[\underline{a}, a_*, a^*, \bar{a}]$  (Dubois 89) which represents a trapezoid.

The greater  $\pi_a(\tau)$ , the greater the possibility that  $a$  is equal to  $\tau$ . In the time interval  $[\underline{a}, \bar{a}]$  (called support),  $0 < \pi_a \leq 1$ . In the time interval  $[a_*, a^*]$  (called core),  $\pi_a(\tau) = 1$ . Outside the interval  $[\underline{a}, \bar{a}]$ ,  $\pi_a(\tau) = 0$ . The wider the support of the fuzzy set, the higher the uncertainty. There are three particular cases: a) triangular form,  $a_* = a^* = a$ , denoted by  $[\underline{a}, a, \bar{a}]$ ; b) imprecise

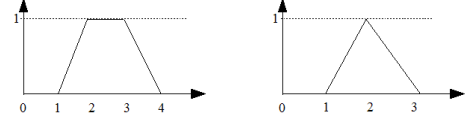


Figure 2: Fuzzy dates : a) a trapezoid ; b) a triangle.

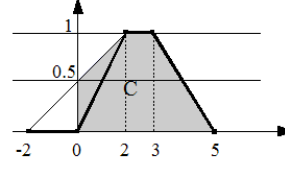


Figure 3: A non trapezoidal fuzzy set

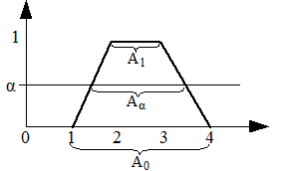


Figure 4: Example of an  $\alpha$ -cut

case,  $\underline{a} = a_*$  and  $a^* = \bar{a}$ , denoted by  $[\underline{a}, \bar{a}]$ ; c) precise case,  $\underline{a} = a_* = a^* = \bar{a}$ .

Let us consider the example depicted by figure 1.b. Transitions  $t_3$ ,  $t_4$  and  $t_5$  are associated with  $[1 \ 2 \ 3 \ 4]$  (fig. 2.a) and transition  $t_2$  is associated with the fuzzy temporal interval  $[1 \ 2 \ 3]$  (fig. 2.b). In the latest case, the values of the support  $[1 \ 3]$  correspond to possible values and the ones of the core  $[2 \ 2]$  correspond to the normal behaviour. The intervals  $[0 \ 0]$  and  $[5 \ 5]$  correspond to precise firing dates.

All operations (addition, subtraction, union, intersection, complement) on fuzzy sets can be done using the extension principle in an easy way if the fuzzy sets are trapezoids or triangles. Otherwise, approximations are required in order to have feasible calculus. The difficulty of such calculus results from the fact that the intersection of two trapezoids is not necessarily a trapezoid.

Let us consider the fuzzy set  $[-2 \ 2 \ 3 \ 5]$  of figure 3.a, and let us assume that it is a quantitative constraint between the firing of two transitions which have to be ordered in a sequence. It is thus necessary to compute its intersection with the interval  $[0 \ \infty)$ . The fuzzy set  $C$  represented by points  $(0,0)$ ,  $(0,0.5)$ ,  $(2,1)$ ,  $(3,1)$  and  $(5,0)$  is obtained and it is not a trapezoid.

It is possible to avoid this problem and consequently to avoid approximating non trapezoid fuzzy sets by trapezoid fuzzy sets by using the concept of alpha-cut (Dubois 88). In this paper we will proceed in this way.

### Alpha cut of a fuzzy set

**Definition 1** Given a fuzzy set  $A$ , the alpha-cut (or  $\alpha$ -cut) set of  $A$  is defined by  $A_\alpha = \{\tau \mid \mu_A(\tau) \geq \alpha\}$ ,  $\alpha \in ]0, 1]$ .

As the set  $A_\alpha$  is a crisp set, the alpha-cut sets are a mean to convert a fuzzy set into a collection of crisp sets, "facilitating" the computations. The set  $A_1$  is obtained for the core, with level  $\alpha = 1$  and the set  $A_0$  is obtained for the support, with level  $\alpha = \epsilon > 0$  (see fig. 4).

**Property 1** The  $\alpha$ -cut of  $A$  are nested sets of  $\mathcal{T}$ : if  $\alpha_1 \geq \alpha$ , then  $A_{\alpha_1} \subseteq A_\alpha$ . The level  $\alpha = 1$  corresponds to the core, and  $\alpha = 0$  to the universal set.

For all fuzzy sets  $A$  and  $B$  of  $\mathcal{T}$ , and for all  $\alpha$ -cuts of  $]0, 1]$ , it is equivalent to directly carry out fuzzy operations on  $A$  and  $B$ , and then construct the  $\alpha$ -cut, or construct first the  $\alpha$ -cut and then to carry out on these  $\alpha$ -cut the corresponding classical operations. So we have:

- $(A \cup B)_\alpha = A_\alpha \cup B_\alpha$
- $(A \cap B)_\alpha = A_\alpha \cap B_\alpha$
- if  $A \subseteq B$  then  $A_\alpha \subseteq B_\alpha$

**Property 2** A fuzzy set can be represented by its  $\alpha$ -cuts. If the nested  $\alpha$ -cuts of a unknown fuzzy set  $A$  are known for all thresholds  $\alpha$ , the membership function of  $A$  can be constructed considering the thresholds from the largest to the smallest.

In the approach presented in this paper, this property allows to rebuild the fuzzy constraints for the state classes and the analysed scenarios.

## Fuzzy Time Petri nets

A t-time Petri net (Merlin 74) allows taking watchdogs into account in a natural way. In this model, an interval  $[a_i, b_i]$  is associated with each transition  $t_i$  and can be considered as a way of defining an *imprecise* enabling duration.

In a Time Fuzzy Petri net (TFPN) (Cardoso 98) the imprecise enabling duration  $[a_i, b_i]$  is extended to a fuzzy duration defined by a possibility distribution.

**Definition 2** A Fuzzy Time Petri Net (FTPN) is a 3-tuple  $\langle \mathcal{N}, M_0, I \rangle$  where:

- $\mathcal{N} = \langle P, T, Pre, Post \rangle$  is a Petri net,
- $M_0$  : is the initial marking,
- $I : T \rightarrow (Q^+ \cup 0) * (Q^+ \cup \infty) * (Q^+ \cup \infty) * (Q^+ \cup \infty)$  is the static interval function represented by a fuzzy set.

The fuzzy static interval function  $I$  associates with each transition  $t_i$  a temporal interval  $[a_i, a_{i*}, a_i^*, \bar{a}_i]$  (see the FTPN of figure 1.b and fuzzy sets of figure 2) that represents the set of its possible firing dates counting from its enabling date.

## Simple temporal network (STN)

**Definition 3 (Simple temporal network)** A simple temporal network (STN) (Dechter 91)  $N$  is composed of a finite set  $V$  of variables  $v_i$  and a finite set  $C$  of **binary** constraints  $C_{ij}(v_i, v_j)$  defined as convex intervals  $[c_{mij}, c_{Mij}]$  delimiting the possible distance between two variables  $v_i$  and  $v_j$  of  $V$ . Each  $C_{ij}$  is therefore

equivalent to:  $c_{mij} \leq v_j - v_i \leq c_{Mij}$   $v_i, v_j \in V$ .

A STN is complete if a constraint  $C_{ij}$  is associated with each pair of variables  $v_i$  and  $v_j$ . If no constraints is defined, it can be assumed that  $C_{ij}(v_i, v_j) = (-\infty + \infty)$  for the sake of completeness.

Figure 6.a depicts an STN with  $V = \{x_1, x_2, x_3\}$  and constraints  $C_{12} = C_{13} = [1, 3]$  and  $C_{32} = [0, 2]$ .

**Definition 4 (STN inclusion)** A STN  $N^1 = (V^1, C^1)$  is included (or nested) in a STN  $N^2 = (V^2, C^2)$ , noted  $N^1 \subseteq N^2$  if and only if:

- they are assumed to be complete;
- they are based on the same set of variables:  $V^1 = V^2 = V$ ;
- $\forall v_i, v_j \in V, C_{i,j}^1(v_i, v_j) \subseteq C_{i,j}^2(v_i, v_j)$ , i.e. the constraints interval between the arcs of  $N^1$  are included (or nested) into the constraints interval of the corresponding arcs of  $N^2$ .

**Definition 5 (A fuzzy STN)** A fuzzy Simple Temporal Network (STN)  $N_F = (V, C_F)$  is a STN where the constraints are expressed by means of fuzzy intervals.

Each alpha-cut (the same alpha level for all the constraints) is therefore a classical STN.

## THE TOOL GRAPHC

The tool GraphC generates a graph of classes proposed in (Mao 05) that allows obtaining the exact temporal constraints for a sequence. The reader should refer to (Mao 05) for a detailed description of its construction. In this section only the main lines are given.

Typically a state graph class of a Petri net is constructed in order to check some property expressed in LTL or CTL. In the case of the tool GraphC the objective is to exactly characterise the temporal constraints which have to be verified by the transition firings in order to implement some scenario i.e. some sequence. It is why sets of constraints are associated with the arcs, and not only with the classes. The second difference (a consequence of the first one) is that in place of a set of clocks and inequality matrices, the temporal constraints are expressed under the form of Simple Temporal Networks (STN) which are complete and minimal (Floyd-Warshall algorithm).

Before presenting the definitions of a class and of the set of constraints associated with an arc between two classes, let us introduce an example in order to illustrate the graph generated by GraphC and later, by the Fuzzy GraphC.

Let us consider the Petri net of figure 1.a, as the representation of a main process that call a distant process. The main process is described by transitions  $t_1, t_2, t_4, t_6$  and  $t_7$ . The distant process is described by transitions



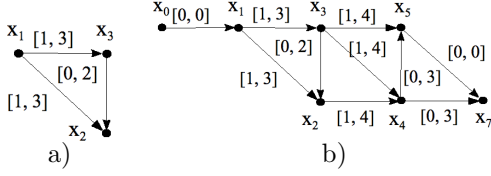


Figure 6: a) The STN  $Nt_{2,3}$ , b) The STN of  $s_0$

Temporal Networks attached to the state classes and to the arcs become Fuzzy Simple Temporal Networks. Any problem due to non trapezoid fuzzy sets (particularly for the execution of Floyd-Warshall algorithm) is avoided by operating on the graphs generated for each alpha-cut separately.

### Decomposing a FTPN into TPN

First of all, all fuzzy time intervals  $I(t_i), i = 1, \dots, 7$  of the Petri net of fig. 1 are converted into ten  $\alpha$ -sets,  $I_\alpha(t_i)$ . Each set  $I_\alpha(t_i), i = 1, \dots, 7$  represents a t-time Petri net (since the time intervals are imprecise or crisp), noted  $\alpha$ -Petri net for short.

The 0-Petri net of fig. 1.a is the graphical representation for  $\alpha = 0$  (the support). For example, the time intervals for  $\alpha = 1, 0.9, 0.5$  and  $0.1$  are:

- for  $\alpha = 1$  (the core):  $I_1(t_1) = I_1(t_7) = [0 \ 0], I_1(t_2) = [2 \ 2], I_1(t_3) = I_1(t_5) = [2 \ 3], I_1(t_4) = [3 \ 3], I_1(t_6) = [5 \ 5]$ .
- for  $\alpha = 0.9$ :  $I_{0.9}(t_1) = I_{0.9}(t_7) = [0 \ 0], I_{0.9}(t_2) = [1.9 \ 2.1], I_{0.9}(t_3) = I_{0.9}(t_5) = [1.9 \ 3.1], I_{0.9}(t_4) = [2.8 \ 3.1]$  and  $I_{0.9}(t_6) = [5 \ 5]$ .
- for  $\alpha = 0.5$ :  $I_{0.5}(t_1) = I_{0.5}(t_7) = [0 \ 0], I_{0.5}(t_2) = [1.5 \ 2.5], I_{0.5}(t_3) = I_{0.5}(t_5) = [1.5 \ 3.5], I_{0.5}(t_4) = [2.0 \ 3.5]$  and  $I_{0.5}(t_6) = [5 \ 5]$ .
- for  $\alpha = 0.1$ :  $I_{0.1}(t_1) = I_{0.1}(t_7) = [0 \ 0], I_{0.1}(t_2) = [1.1 \ 2.9], I_{0.1}(t_3) = I_{0.1}(t_5) = [1.1 \ 3.9], I_{0.1}(t_4) = [1.2 \ 3.9]$  and  $I_{0.1}(t_6) = [5 \ 5]$ .

According to Property 1, for all  $\alpha$  sets  $I_\alpha(t)$  for a transition  $t_i$ , they are nested:  $I_1(t_i) \subseteq I_{0.9}(t_i) \dots \subseteq I_{0.1}(t_i) \subseteq I_0(t_i)$ .

The following notations are used in the remainder of the paper:  $G^\alpha = (\mathcal{N}^\alpha, \mathcal{A}^\alpha)$  is a graph generated by an  $\alpha$ -Petri net;  $[a]^\alpha$  an arc of a graph  $G^{\alpha_1}$ ,  $[C]^\alpha$  a class of a graph  $G^{\alpha_1}$ , and  $s^{\alpha_1}$  a sequence of a graph  $G^{\alpha_1}$ . A class  $[C]^\alpha$  is noted  $[(M, Nc)]^\alpha$  or  $([M]^\alpha, [Nc]^\alpha)$ . An arc  $[a]^\alpha$  is labelled by a transition  $t_i$  and the STN  $[Nt_{i,c}]^\alpha$  delimiting this firing.

For each  $\alpha$ -Petri net, a state class graph  $G^\alpha$  is constructed; fig. 5 corresponds to  $G^0$  (support). The other graphs for the other values of  $\alpha$  are depicted in figure 7 and all classes for all these graphs are represented in Table 1. The first two columns indicate the marking and the STN constraints (in this example there is just a pair of nodes) for all classes. The following columns

indicate the name of the classes for each  $\alpha$ -cut, and the value of  $\alpha$ . The name of the classes of the composed fuzzy graph  $G_F$  is listed in the last column. Each line indicates: the marking  $M_i$ , the STN  $Nc_i$ , and, for each set of columns the name of the classes in  $G^\alpha$  and the constraint value. An empty value in a column labeled by  $\alpha$  and a line labelled by  $C_{F_i}$  indicates that there is no class included in  $C_{F_i}$  for this  $G^\alpha$ . We can see in figure 7 that some graphs have the same structure (as  $G^{0.9}, G^{0.8}, G^{0.7}$  and  $G^{0.6}$ ); in these case, the name of the class (and arcs) are the same but their STN can be different as it can be seen in Table 1. The list of restricted classes with their "original" (or mother) class ( $G^1$  and  $G^{0.9}$  have no restricted classes) is :  $[C_{17}]^{0.5} = [C_2]_r^{0.5}$ ,  $[C_{19}]^{0.3} = [C_2]_r^{0.3}$ ,  $[C_{20}]^{0.3} = [C_3]_r^{0.3}$ ,  $[C_{22}]^{0.2} = [C_2]_r^{0.2}$ ,  $[C_{23}]^{0.2} = [C_3]_r^{0.2}$ ,  $[C_{24}]^{0.2} = [C_{10}]_r^{0.2}$ ,  $[C_{25}]^{0.2} = [C_5]_r^{0.2}$ ,  $[C_{26}]^{0.2} = [C_{22}]_r^{0.2}$ ,  $[C_{27}]^{0.2} = [C_{16}]_r^{0.2}$ ,  $[C_{25}]^0 = [C_2]_r^0$ ,  $[C_{22}]^0 = [C_3]_r^0$ ,  $[C_{24}]^0 = [C_5]_r^0$  and  $[C_{26}]^0 = [C_{16}]_r^0$  (see def. 8).

The next step is to build the fuzzy state class graph  $G_F$  from all these graphs  $G^\alpha$  as described in the sequel.

### Composing several GraphC $G^\alpha$ in a Fuzzy GraphC $G_F$

#### Principle

The objective is to derive a Fuzzy GraphC  $G_F$  from all  $G^\alpha$  for each  $\alpha$ -Petri net. In order to do this, it is necessary to match all the  $G^\alpha$  in order to find how their classes can be grouped in order to derive the classes of  $G_F$  and then reconstruct for each class (and arc) the fuzzy STN representing them. Although the problem of matching two graphs is an NP complete problem, in this case the problem is much simpler because the firing sequences are necessarily preserved.

It follows indeed from the alpha cut definition that  $G^{\alpha_1} \subseteq G^\alpha$  for  $\alpha_1 \geq \alpha$ , the only problem is to find which classes of  $G^{\alpha_1}$  are contained in those of  $G^\alpha$ , since the structure of the graphs (number of nodes and arcs) can be different due to the restricted classes that can appear or disappear when  $\alpha$  changes (no matter if it increases or decreases). In other words, in order to have the same graph structure for all the  $G^\alpha$ , it is sometimes necessary for some  $G^\alpha$  to introduce a restricted class which is not actually restricted for the current value of  $\alpha$  because it exists for some other values of  $\alpha$ . Similarly, two classes may be equivalent within one  $G^\alpha$  and not within other ones. They have then to contribute for this  $\alpha$ -cut to two different classes in the Fuzzy GraphC as if they were not equivalent.

#### Matching the state class graphs of two $\alpha$ -cuts

The algorithm for finding the corresponding state classes in two state class graphs ( $G^{\alpha_1}$  and  $G^\alpha$ ) in order to build the fuzzy state class graph is the algorithm 1. The function IsIncludedClass is based on definition 10:

**Definition 10 (Classes inclusion)** A class  $[C]^\alpha = [(M, Nc)]^\alpha$  (of graph  $G^{\alpha_1}$ ) with  $[Nc = (V, C)]^\alpha$  is

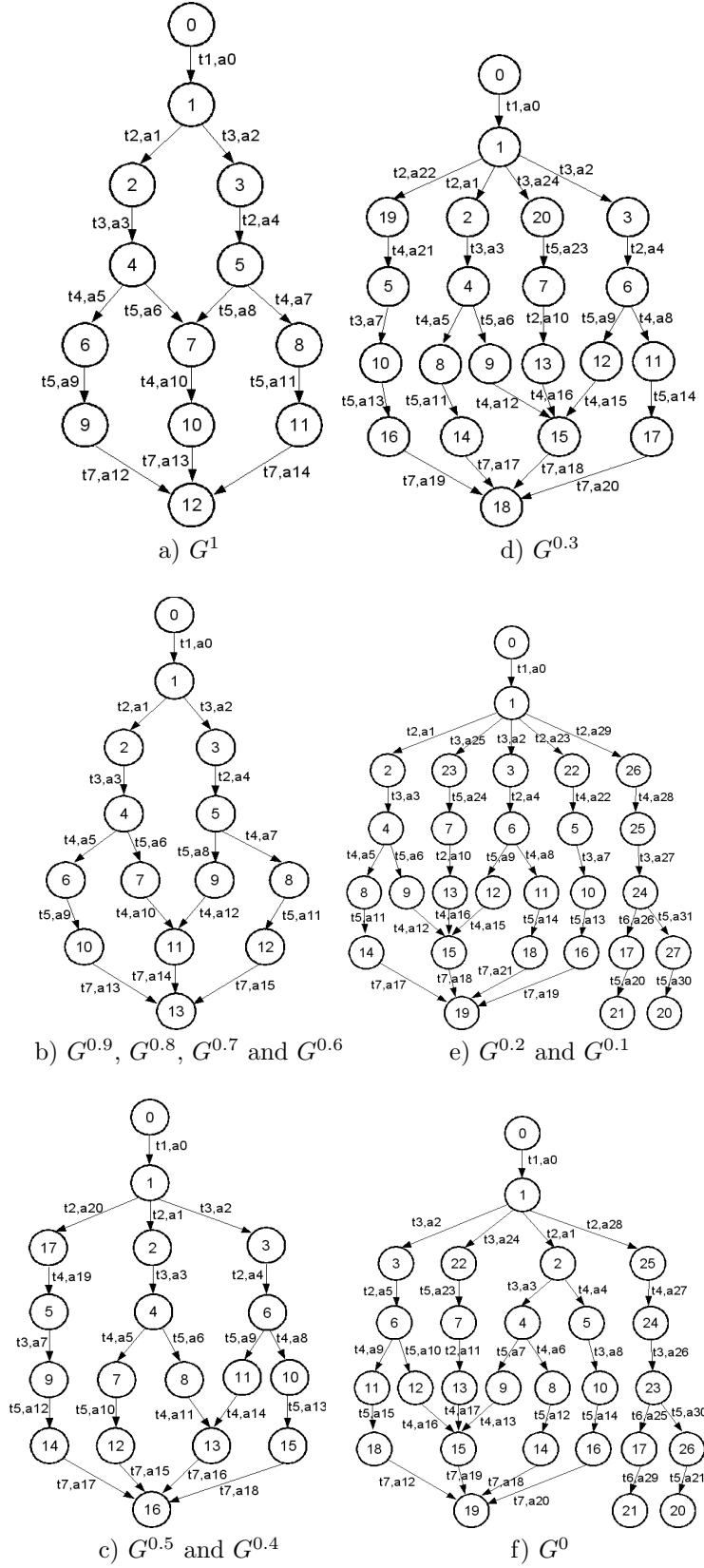


Figure 7: Graphs of classes  $G^\alpha$

$M_i$	$N_{C_i}$	$C_i$	$\alpha = 1$	$\alpha = 0.9$	$\alpha = 0.8$	$\alpha = 0.7$	$\alpha = 0.6$	$\alpha = 0.5$	$\alpha = 0.4$	$\alpha = 0.3$	$\alpha = 0.2$	$\alpha = 0.1$	$C_i$	$\alpha = 0$	$C_{F_i}$
$p_1$	$x_0$	$C_0$											$C_0$	$[1\ 3]$	$C_{F_0}$
$p_2 p_3$	$x_1$	$C_1$	$[2\ 2]$										$C_1$	$[1\ 3]$	$C_{F_1}$
$p_3 p_4$	$C_{1,2}$	$C_2$	$[2\ 2]$										$C_2^*$	$[0\ 2]$	$C_{F_2}$
$p_4 p_5$	$C_{1,3}$	$C_3$	$[0\ 1]$										$C_3$	$[0\ 2]$	$C_{F_3}$
$p_4 p_5$	$C_{2,3}$	$C_4$	$[0\ 1]$										$C_4$	$[0\ 2]$	$C_{F_4}$
$p_4 p_5$	$C_{3,2}$	$C_5$	$[0\ 0]$										$C_5$	$[0\ 2]$	$C_{F_5}$
$p_5 p_6$	$C_{3,4}$	$C_6$	$[2\ 3]$										$C_6$	$[0\ 4]$	$C_{F_6}$
$p_5 p_6$	$C_{2,5}$	$C_7^+$	$[2\ 3]$										$C_7$	$[1\ 4]$	$C_{F_7}$
$p_5 p_6$	$C_{3,4}$	$C_8^+$	$[3\ 3]$										$C_8$	$[1\ 4]$	$C_{F_8}$
$p_4 p_7$	$C_{2,5}$	$C_9^+$	$[2\ 3]$										$C_9$	$[1\ 4]$	$C_{F_9}$
$p_6 p_7$	$C_{4,5}$	$C_9$	$[0\ 1]$										$C_{10}$	$[0\ 4]$	$C_{F_{10}}$
$p_6 p_7$	$x_4$	$C_{10}$	$[0\ 0]$										$C_{11}$	$[0\ 3]$	$C_{F_{11}}$
$p_6 p_7$	$x_7$	$C_{12}$	$[0\ 0]$										$C_{14}$	$[0\ 4]$	$C_{F_{14}}$
$p_9$		$C_{13}$											$C_{15}$	$[0\ 3]$	$C_{F_{15}}$
$p_3 p_6$	$C_{1,4}$												$C_{19}$	$[2\ 4]$	$C_{F_{19}}$
$p_5 p_6$	$C_{4,3}$												$C_5$	$[2\ 4]$	$C_{F_{14}}$
$p_6 p_7$	$x_2$												$C_{10}$	$[0\ 2]$	$C_{F_{14}}$
$p_6 p_7$	$C_{1,5}$												$C_{16}$	$[1\ 5]$	$C_{F_{16}}$
$p_7 p_8$	$x_6$												$C_{22}^*$	$[1\ 3]$	$C_{F_{17}}$
$p_7 p_8$	$x_5$												$C_{22}$	$[1\ 2]$	$C_{F_{18}}$
$p_5 p_6$	$C_{4,3}$												$C_7$	$[2\ 3]$	$C_{F_{20}}$
$p_3 p_6$	$C_{1,4}$												$C_{17}$	$[3\ 4]$	$C_{F_{21}}$
$p_3 p_4$	$C_{1,2}$												$C_{20}$	$[3\ 4]$	$C_{F_{22}}$
													$C_{21}$	$[1\ 2]$	$C_{F_{23}}$
													$C_{24}$	$[2\ 3]$	$C_{F_{24}}$
													$C_{25}$	$[1\ 2]$	$C_{F_{25}}$
													$C_{26}$	$[1\ 2]$	$C_{F_{26}}$
													$C_{27}$	$[5\ 5]$	$C_{E_{27}}$

Table 1: Nested classes for all graphs  $G^\alpha$ .

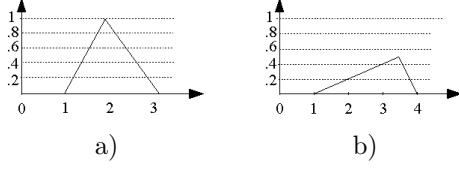


Figure 8: The fuzzy constraint associated with a class:  
a)  $C_{F_{13}}$  of  $C_{F_3}$ , b)  $C_{F_{14}}$  of  $C_{F_{14}}$

included (or nested) into a class  $[C]^\alpha = [(M, Nc)]^\alpha$  (of graph  $G^\alpha$ ) with  $[Nc = (V, C)]^\alpha$ , noted  $[C]^{\alpha_1} \subseteq [C]^\alpha$ , if:

- they have the same marking,  $[M]^{\alpha_1} = [M]^\alpha$ ;
- $[Nc]^{\alpha_1}$  is included in  $[Nc]^\alpha$  (according def. 4);
- they are reached from the initial state class by the same firing sequences;
- from them the same firing sequences can be fired.

The execution of this algorithm provides a list ListCl with all pairs of classes ( $[C]^{\alpha_1}, [C]^\alpha$ ),  $[C]^{\alpha_1} \subseteq [C]^\alpha$  and a list ListArc with all pairs of arcs ( $[a]^{\alpha_1}, [a]^\alpha$ ),  $[a]^{\alpha_1} \subseteq [a]^\alpha$ . Table 1 was constructed based on ListCl. It can be observed that the constraints for a given class are such that  $C_{ij}^{\alpha_1} \subseteq C_{ij}^\alpha$  if  $\alpha_1 \geq \alpha$  (each class is included in the class at its right side in the same line). A fuzzy class  $C_F$  is "constructed" from all its  $\alpha$  cuts (the nested crisp sets) in a same line (see Table 1), using Property 2, as for example,  $C_{F_3}$  whose STN is represented in fig. 8.a. The obtained fuzzy class can be not normalised, for example if there is no class  $[C_i]^1$  (the core) on its line. It is the case of class  $C_{F_{14}}$  represented in fig. 8.b, since the maximum value is  $\alpha = 0.5$ .

In the example considered here, the structure of the Fuzzy GraphC  $G_F$  is that of the GraphC for the  $\alpha$ -cuts 0.1 and 0.2 (figure 7.e). This illustrates the fact that it is not necessarily the structure of  $G^0$ , the GraphC for the support, that defines the structure of  $G_F$  although it is the one describing the less constrained system.

#### Particular cases of class inclusion

It can be observed in Table 1 that there are two types of class inclusion  $[C]^{\alpha_1} \subseteq [C]^\alpha$ :

1.  $G^{\alpha_1}$  and  $G^\alpha$  have exactly the same structure. It is the case for: 1)  $G^{0.5}$  and  $G^{0.4}$ , 2)  $G^{0.9}$ ,  $G^{0.8}$ ,  $G^{0.7}$  and  $G^{0.6}$  and 3)  $G^{0.2}$  and  $G^{0.1}$ .
2.  $G^{\alpha_1}$  and  $G^\alpha$  do not have the same structure. As mentioned informally above, there are two interesting cases (see Table 1):
  - Case 1: for  $\alpha_1 > \alpha$ ,  $[C_j]^{\alpha_1} \subseteq [C_k]^\alpha$  and  $[C_j]^{\alpha_1} \subseteq [C_i]^\alpha$ , i.e. a class of  $G^{\alpha_1}$  is contained in two classes of  $G^\alpha$ . It is the case for  $G^1$  and  $G^{0.9}$ :  $[C_7^+]^1$  is nested in both classes  $[C_7]^{0.9}$  and  $[C_9]^{0.9}$ .
  - Case 2: for  $\alpha_1 > \alpha$ ,  $[C_j]^{\alpha_1} \subseteq [C_k]^\alpha$  and  $[C_i]^{\alpha_1} \subseteq [C_k]^\alpha$ , i.e. two classes of  $G^{\alpha_1}$  are contained in a same class of  $G^\alpha$ . It is the case for  $G^{0.1}$  and  $G^0$ :

$[C_2]^{0.1} \subseteq [C_2^*]^0$  and  $[C_{22}]^{0.1} \subseteq [C_2^*]^0$ . By the way,  $[C_{22}]^{0.1}$  is a restricted class of  $[C_2]^{0.1}$ .

#### Algorithm 1 Matching of two graphs ( $G^{\alpha_1}, G^\alpha$ ), $\alpha_1 \geq \alpha$

```

InclusionGraph( $G^{\alpha_1}, G^\alpha$ )
nodesIC = add( $G^{\alpha_1}$ .InitialClass,  $\alpha_1$ ) {Treat the initial class  $[C_0]^{\alpha_1}$ }
ListCl.add(( $G^{\alpha_1}$ .InitialClass,  $\alpha_1$ ), ( $G^\alpha$ .InitialClass,  $\alpha$ )) {The initial classes are always nested  $[C_0]^{\alpha_1} \subseteq [C_0]^\alpha$ }
while nonEmpty(nodesIC) do
   $n_1$  = nodesIC.get(0) {Extract a node from nodesIC ( $G^{\alpha_1}$ )}
   $n$  =  $n_1$ .IncludeClass {Treat the corresponding class  $[C]^\alpha$ , with  $[C]^{\alpha_1} \subseteq [C]^\alpha$ }
  for i=1 to  $n_1$ .outputArc.size do
    ( $\text{arc}_1, \alpha_1$ ) = ( $n_1$ .outputArc.get(i),  $\alpha_1$ ) {take an output arc of  $[C]^{\alpha_1}$ }
    for j=1 to  $n$ .outputArc.size do
      ( $\text{arc}, \alpha$ ) = ( $n$ .outputArc.get(j),  $\alpha$ ) {take an output arc of  $[C]^\alpha$ }
      {If both arcs in  $G^{\alpha_1}$  and  $G^\alpha$  are labelled by the same transition}
      if  $\text{arc}_1$ .trans =  $\text{arc}$ .trans then
        {Verify if the target classes are included}
        if IsIncludedClass(( $\text{arc}_1$ .targetCl,  $\alpha_1$ ), ( $\text{arc}$ .targetCl,  $\alpha$ )) then
          ListCl.add(( $\text{arc}_1$ .targetCl.getId,  $\alpha_1$ ), ( $\text{arc}$ .targetCl.getId,  $\alpha$ )) {add the pair  $[C_T]^{\alpha_1} \subseteq [C_T]^\alpha$ }
          nodesIC = add( $\text{arc}_1$ .targetCl) {add this class to be analysed later if it is not yet}
          if IsIncludedSTN(( $\text{arc}_1$ .STN,  $\alpha_1$ ), ( $\text{arc}$ .STN,  $\alpha$ )) then
            ListArc.add(( $\text{arc}_1$ .getId,  $\alpha_1$ ), ( $\text{arc}$ .getId,  $\alpha$ )) {add the pair  $[\text{arc}_1]^{\alpha_1} \subseteq [\text{arc}]^\alpha$ }
          end if
        end if
      end if
    end for
  ListArc = MostRestrictedIncl(ListArc,  $\text{arc}_1$ ) {Keep the most restricted inclusion for a same arc}
  ListCl = MostRestrictedIncl(ListCl,  $\text{arc}_1$ .targetCl) {Keep the most restricted inclusion for a same class}
end for
nodesIC.remove(0) {Remove the class already treated}
end while

```

#### Analysing a sequence on $G_F$

If we consider the firing of sequence  $s_2 = t_1; t_2; t_4; t_3; t_6; t_5$  in fig. 1.b, corresponding to the loss of a late response, the state class reached is  $C_{F_{23}}$  in  $G_F$  (corresponding to node  $C_{21}$  in figures 7.e and 7.f). In Table 1 it appears that  $C_{F_{23}}$  has no corresponding classes for  $\alpha > 0.2$  and consequently this means that the possibility of this behaviour is 0.2. For lack of space, the STN  $Nt_{i,c}$ , delimiting the firing of  $t_i$  from class  $C$ , for the transitions in the sequence  $s_2$  are given in textual form (see def. 3):

- $Nt_{1,0} : C_{01}(x_0, x_1)$  (arc a0);
  - $Nt_{2,1} : C_{12}(x_1, x_2)$  (arc a29);
  - $Nt_{4,26} : C_{12}(x_1, x_2), C_{14}(x_1, x_4)$  and  $C_{24}(x_2, x_4)$  (arc a28);
  - $Nt_{3,25} : C_{14}(x_1, x_4), C_{13}(x_1, x_3)$  and  $C_{43}(x_4, x_3)$  (arc a27);
  - $Nt_{6,24} : C_{43}(x_4, x_3), C_{46}(x_4, x_6)$  and  $C_{36}(x_3, x_6)$  (arc a26);
  - $Nt_{5,17} : C_{36}(x_3, x_6), C_{35}(x_3, x_5)$  and  $C_{65}(x_6, x_5)$  (arc a20).
- The STN  $Ns_2$  for  $\alpha$ -cut .1 and .2 is obtained by the union  $Nt_{1,0} \cup Nt_{2,1} \cup Nt_{4,26} \cup Nt_{3,24} \cup Nt_{6,23} \cup Nt_{5,17}$ ; they have the same structure depicted in figure 9.a. The STN  $Ns_2$  for the core (0-cut) is obtained by the union  $Nt_{1,0} \cup Nt_{2,1} \cup Nt_{4,26} \cup Nt_{3,25} \cup Nt_{6,24} \cup Nt_{5,17}$  (respectively, arcs a0, a28, a27, a26, a25, a21 in fig. 7.f) and its structure is also as in figure 9.a. The constraints values  $C_{ij}$  for all  $\alpha$ -cuts are given in Table 2.

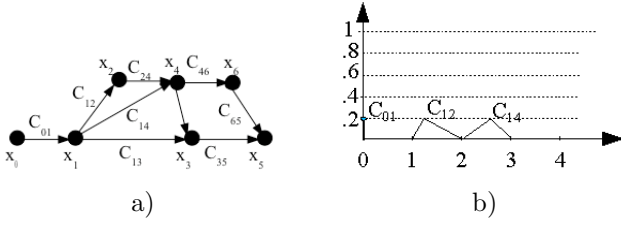


Figure 9: Sequence  $s_2 = t_1 t_2 t_4 t_3 t_6 t_5$ : a) STN, b) Fuzzy constraints  $C_{01}$ ,  $C_{12}$  and  $C_{14}$

arc/ $\alpha$	$C_{01}$	arc/ $\alpha$	$C_{12}$	$C_{14}$	$C_{24}$
$a_0/0$	[0.0 0.0]	$a_{27}/0$	[1.0 2.0]	[2.0 3.0]	[1.0 2.0]
$a_0/.1$	[0.0 0.0]	$a_{28}/.1$	[1.1 1.6]	[2.3 2.8]	[1.2 1.7]
$a_0/.2$	[0.0 0.0]	$a_{28}/.2$	[1.2 1.2]	[2.6 2.6]	[1.4 1.4]

$t_1$		$t_4$			
arc/ $\alpha$	$C_{12}$	arc/ $\alpha$	$C_{14}$	$C_{13}$	$C_{43}$
$a_{28}/0$	[1.0 2.0]	$a_{26}/0$	[2.0 3.0]	[3.0 4.0]	[1.0 2.0]
$a_{29}/.1$	[1.1 1.6]	$a_{27}/.1$	[2.3 2.8]	[3.4 3.9]	[1.1 1.6]
$a_{29}/.2$	[1.2 1.2]	$a_{27}/.2$	[2.6 2.6]	[3.8 3.8]	[1.2 1.2]

$t_3$			$t_6$				
arc/ $\alpha$	$C_{43}$	$C_{46}$	$C_{36}$	arc/ $\alpha$	$C_{36}$	$C_{35}$	$C_{65}$
$a_{25}/0$	[1.0 2.0]	[5.0 5.0]	[3.0 4.0]	$a_{21}/0$	[3.0 4.0]	[3.4 3.9]	[0.0 1.0]
$a_{26}/.1$	[1.1 1.6]	[5.0 5.0]	[3.4 3.9]	$a_{20}/.1$	[3.4 3.9]	[3.4 3.9]	[0.0 0.5]
$a_{26}/.2$	[1.2 1.2]	[5.0 5.0]	[3.8 3.8]	$a_{20}/.2$	[3.8 3.8]	[3.8 3.8]	[0.0 0.0]

Table 2: Constraints values for the STN of sequence  $s_2$  of graphs  $G^0$ ,  $G^{0.1}$  and  $G^{0.2}$ .

The fuzzy STN  $N_{F}s_2$  (def. 5) is obtained from Table 2 according to property 2. The (triangular) fuzzy constraints such obtained are:  $C_{13} = [3 \ 3.8 \ 4]$ ,  $C_{24} = [1 \ 1.4 \ 2]$ ,  $C_{35} = [3.4 \ 3.8 \ 3.9]$ ,  $C_{36} = [3 \ 3.8 \ 4]$ ,  $C_{43} = [1 \ 1.2 \ 2]$ ,  $C_{46} = [5 \ 5 \ 5]$ , and  $C_{65} = [0 \ 0 \ 1]$ ;  $C_{01}$ ,  $C_{12}$  and  $C_{14}$  are depicted in fig. 9.b. We can see that the core of all these constraints is empty and the height is .2. It means that the *possibility* that transition  $t_6$  could be fired is .2 and so the *necessity* (certainty) is zero.

## CONCLUSION

This paper has shown that it is possible to use the tool GraphC to analyse a Fuzzy Time Petri net. By decomposing the fuzzy intervals attached to the transitions into a set of  $\alpha$ -cuts, generating the state class graphs for each one, and by recomposing the  $\alpha$ -cuts, it is possible to derive the fuzzy constraints of the fuzzy simple temporal networks attached to the classes and to the arcs of the fuzzy GraphC.

In place of just answering *yes* or *no* for a property to be verified, it is possible to derive a possibility degree (quantitative view) that the property is not verified. The more important is that the sequences leading to the states expressing the property violation are clearly characterised: the temporal constraints existing among the firing dates are delimited in a complete and minimal way. It will be an important aid for the designer to

modify, if necessary, the values of the parameters.

The tool GraphC that generates a graph of classes for a t-time Petri net can be downloaded at <http://graphc.sourceforge.net>. Currently the algorithm 1 is being implemented in order to automatically generate a fuzzy graph of classes. Further work consists in exploring another way to generate the fuzzy graph of classes: its direct generation without using  $\alpha$ -cuts.

## REFERENCES

- [Berthomieu 04] B. Berthomieu, P.O. Ribet, F. Vernadat : The tool TINA: construction of abstract state spaces for Petri nets and time Petri nets, IJPR, Vol.42, N<sup>o</sup>14, pp.2741-2756, 15th July 2004.
- [Cardoso 98] J. Cardoso : Time Fuzzy Petri nets. *Fuzziness in Petri nets*, J. Cardoso and H. Camargo (Ed), Studies in Fuzziness, Physica Verlag, pp 115-145, 1998.
- [Cardoso 05] J. Cardoso, S. Cousy, G. Juanole : Extending time Petri nets to fuzzy time Petri nets: definition of the graph of fuzzy state class, 16th IFAC World Congress, July 2005, Prague.
- [Dechter 91] R. Dechter, I. Meiri, J. Pearl : Temporal constraint networks, Artificial Intelligence, vol 49, p.61-91, 1991.
- [Dubois 88] D. Dubois, H. Prade. *Possibility theory: an approach to computerized processing of uncertainty* Plenum Press, New York, 1988, 263 p.
- [Dubois 89] D. Dubois, H. Prade. Processing fuzzy temporal knowledge. *IEEE Trans. on Syst. Man and Cyber.*, 14, 1989, n<sup>o</sup> 4.
- [Mao 05] X. Mao, J. Cardoso, R. Valette : A New Graph of Classes for the Preservation of Quantitative Temporal Constraints. ATVA 2005, Taipei, Taiwan, October 4-7, 2005, LNCS 3707, pp.278-292 Springer-Verlag.
- [Merlin 74] P. Merlin. *A Study of the recoverability of Computer Systems*. Phd thesis. University of California, Irvine, 1974.
- [Yager 84] R.R. Yager. On different classes of linguistic variables defined via fuzzy subsets. *Kibernetes*, 13:103-110, 1984.
- [Yoneda 98] T. Yoneda, H. Ryuba, CTL Model checking of time Petri nets using geometric regions, *IEICE Trans. inf. & Syst.*, Vol E81-D, No. 3, pp.297-396, 1998.