State class graph for Fuzzy Time Petri Nets

Janete Cardoso\textsuperscript{1}, Xiaoyu Mao\textsuperscript{2} and Robert Valette\textsuperscript{3}

\textsuperscript{1} IRIT-UT1, 21, allée de Brienne, 31042 Toulouse France
\textsuperscript{2} Almende / University of Maastricht, The Netherlands
\textsuperscript{3} LAAS-CNRS, 31077 Toulouse, France
e-mail: jcardoso@univ-tlse1.fr, xmao@almende.com, robert@laas.fr

ESM 2006
Toulouse, France, 24 October 2006

Plan

1) Introduction and informal presentation of the approach
2) Basic notions
3) GraphC
4) Fuzzy GraphC
5) Conclusion
Property verification is typically based on Model Checking *i.e.* the system is a model of the formula expressing the property (*model* in the *logical* sense).

- Verification is done by state enumeration
- Continuous (dense) time $\Rightarrow$ number of states is infinite
- Group an infinite number of states together into one class of states (abstract class) with a *finite* number of classes
- Petri nets $\Rightarrow$ a class is a marking associated with a set of temporal inequalities

---

There are various kinds of classes:

- All the states which can be reached by the same sequences of events (same *past*)
- Only the states from which exactly the same sequences can be fired (same *branching future*)
- Only the states with exactly the same past and the same future sequence of events (*past+future*) and with the *exact quantitative* temporal constraints over the firing dates

$\Rightarrow$ The third kind of classes (*GraphC*) is considered in this paper.

It is the richest for the designer.
Introduction 3

- **Fuzzy** sets: a way for expressing, in an aggregated way, sets or constraints which fit into each other
- For example:

During design: constraints easy to meet and stronger constraints more expensive to meet

![Fuzzy set diagram](image)

Introduction 4

Goal of the paper:

In place of proving the property for given temporal constraints, determine a set of **consistent** temporal constraints (i.e. requiring the same degree of effort) in order to guarantee that the property is just verified.

For this:

- Model of the system: Time Petri net → Fuzzy Time Petri net
- Classes: Temporal constraints → Fuzzy Temporal constraints

→ How to build the Fuzzy GraphC?
Introduction 4

Goal of the paper:

In place of proving the property for given temporal constraints, determine a set of consistent temporal constraints (i.e. requiring the same degree of effort) in order to guarantee that the property is just verified.

For this:

- Model of the system: Time Petri net ➔ Fuzzy Time Petri net
- Classes: Temporal constraints ➔ Fuzzy Temporal constraints

➔ How to build the Fuzzy GraphC?

Introduction 5

What are the difficulties?

- Direct computation ➔ Fuzzy set intersections
- The result is no longer in a trapezoidal form
- Re-normalization is not exact and cumbersome

Another way to proceed (the one presented here)

- Operate $\alpha$-cut by $\alpha$-cut (a GraphC for each one)
- Recompose the fuzzy constraints defining the same class in all the GraphCs by aggregating the corresponding $\alpha$-cuts

➔ Not straightforward because the GraphCs can be different (different structures) for the different $\alpha$-cuts
Introduction 5

What are the difficulties?

- Direct computation $\rightarrow$ Fuzzy set intersections
- The result is no longer in a trapezoidal form
- Re-normalization is not exact and cumbersome

Another way to proceed (the one presented here)

- Operate $\alpha$-cut by $\alpha$-cut (a GraphC for each one)
- Recompose the fuzzy constraints defining the same class in all the GraphCs by aggregating the corresponding $\alpha$-cuts

$\rightarrow$ Not straightforward because the GraphCs can be different (different structures) for the different $\alpha$-cuts

Plan

1) Introduction and informal presentation of the approach
2) Basic notions
3) GraphC
4) Fuzzy GraphC
5) Conclusion
**Basic notions 1**

**Fuzzy date**

A date $a$ has

*only one value*, which may be ill-known.

The fuzzy set of its possible values is a disjunctive set.

The available knowledge about a date $a$ is represented by a possibility distribution $\pi_a(\tau), \tau \in T$, delimited by the fuzzy set $A$, $[a, a_\ast, a^\ast, \bar{a}]$ which represents a trapezoid.

**Fuzzy constraints** are defined as fuzzy distances between two variables.

---

**Basic notions 2**

**Definition of an $\alpha$-cut**

Given a fuzzy set $A$, the alpha-cut (or $\alpha$-cut) set of $A$ is defined by

$A_\alpha = \{ \tau \mid \pi_A(\tau) \geq \alpha \}, \alpha \in [0, 1]$.

→ 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 $\alpha$-cuts from the largest to the smallest.
**Basic notions 3**

**Definition of a Fuzzy Time Petri net**

A Fuzzy Time Petri Net (FTPn) is a 3-tuple $\langle N, M_0, I \rangle$ where:
- $N = \langle P, T, Pre, Post \rangle$ is a Petri net,
- $M_0$ is the initial marking,
- $I : T \rightarrow (\mathbb{Q}^+ \cup 0)^2 \times (\mathbb{Q}^+ \cup 0 \cup \infty)^2$ is the static interval function represented by a fuzzy set.

$I$ associates with each transition $t$ a duration, delimited by a fuzzy set, during which $t$ has to remain enabled before firing.

**Basic notions 4**

**Definition of a Simple Temporal Network**

A simple temporal network (STN) $N$ 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:

$c_{mij} \leq v_j - v_i \leq c_{Mij} \quad v_i, v_j \in V$

- A STN is complete if a constraint $C_{ij}$ is associated with each pair of variables $v_i$ and $v_j$.
- A Fuzzy Simple Temporal Network is when the intervals are delimited by fuzzy sets.
Plan

1) Introduction and informal presentation of the approach
2) Basic notions
3) GraphC
4) Fuzzy GraphC
5) Conclusion

GraphC 1

- The tool GraphC generates a graph of state classes for a Timed Petri net.
- The classes group together states with same past + future and with the exact quantitative temporal constraints over the firing dates.

Does the response arrives in time or not (t_7 or t_6 fires)?
GraphC 2

GraphC 3

Class (node):
Marking + Simple Temporal Network on firing times

Firing (arc):
Simple Temporal Network on firing times

STN for firing t_2 from C_3
**GraphC 4**

**Classe (node):**
Marking + Simple Temporal Network on firing times

**Firing (arc):**
Simple Temporal Network on firing times

STN for $c_3$

---

**GraphC 5**

**Classe (node):**
Marking + Simple Temporal Network on firing times

**Firing (arc):**
Simple Temporal Network on firing times

STN for firing $t_2$ from $c_3$
GraphC 6

STN for firing the sequence
\( s = t_1 ; t_3 ; t_2 ; t_4 ; t_5 ; t_7 \)

GraphC 7

**Restricted class:**

Stronger constraints to fire a given transition

\( C_{22} \) is a restricted class of \( C_3 \) to fire \( t_5 \) from marking \( p_2, p_5 \)

\( X_1 \ [1, 3] \ X_3 \)

STN of class \( C_3 \)

\( X_1 \ [1, 2] \ X_3 \)

STN of class \( C_{22} \)
Plan

1) Introduction and informal presentation of the approach
2) Basic notions
3) GraphC
4) Fuzzy GraphC
5) Conclusion

Fuzzy GraphC 1

- For each $\alpha$-cut, the structure of the GraphC may be different
- When the constraints turn looser (more slack or more imprecision)
  - new classes, new behaviors, new restricted classes may appear,
- It is also possible that the structure of the GraphC turns simpler when the constraints turn looser

- Let us see the example
Fuzzy GraphC 2

\( \alpha \)-cut 1 (core)

Fuzzy GraphC 3

\( \alpha \)-cut 0.8
Fuzzy Graph C 6

\(\alpha\)-cut 0.2

Fuzzy Graph C 7

\(\alpha\)-cut 0.5 (support)
Fuzzy Graph C 8 - Fuzzy constraint

- Start from the $\alpha$-cut 1.0 (core)
- Consider a new $\alpha$-cut and associate each class of the preceding $\alpha$-cut with a class of this new $\alpha$-cut.
- When needed, classes may be broken down into several ones in order to be associated with a group of restricted class of the same class.
- When needed (new behavior) void classes are introduced for the preceding $\alpha$-cut (state not reachable for $\alpha$-cut 1.0 (core) but reachable for $\alpha$-cut 0+ (support)).
Fuzzy GraphC 10 - Fuzzy constraint

- $\alpha$-cut 0.8

Class $C_3$, $(x_1, x_3)$

Class $C_2$, $(x_1, x_4)$

Fuzzy GraphC 11 - Fuzzy constraint

- $\alpha$-cut 0.6

Class $C_3$, $(x_1, x_3)$

Class $C_2$, $(x_1, x_4)$
Fuzzy GraphC 12 - Fuzzy constraint

- \( \alpha \)-cut 0.5

Class \( C_3 \), \((x_1, x_3)\)

Class \( C_5 \), \((x_1, x_4)\)

Fuzzy GraphC 13 - Fuzzy constraint

- \( \alpha \)-cut 0.4

Class \( C_3 \), \((x_1, x_3)\)

Class \( C_5 \), \((x_1, x_4)\)
**Fuzzy GraphC 14 - Fuzzy constraint**

- $\alpha$-cut 0.2

Class $C_3$, $(x_1, x_3)$

Class $C_5$, $(x_1, x_4)$

**Fuzzy GraphC 15 - Fuzzy constraint**

- $\alpha$-cut 0.5

Class $C_3$, $(x_1, x_3)$

Class $C_5$, $(x_1, x_4)$
As a conclusion for this example:

- The possibility of the undesired path (late response) is 0.2

Plan

1) Introduction and informal presentation of the approach
2) Basic notions
3) GraphC
4) Fuzzy GraphC
5) Conclusion
Conclusion 1

Two main points have to be underlined:

- Our approach allows a quantitative evaluation
  - Verify if the property is true or not
    - Property violation is possible with a degree of 0.2

- Our approach is a rich guide for the designer:
  - How to ensure that the property is verified
  - Choose parameter values and check for the property
    - Determine parameter values such that the property is verified
  - In the example, it is sufficient to restrict all the parameter values to the $\alpha$-cut $0.2^+$ to ensure the property

The End

Thank you for your attention!
Merci pour votre attention!

Any question?
Des questions?
Conclusion 2

The approach presented in this paper looks like a try and error method, trying all the parameters values in order to find the best trade-off between cost and property verification.

It is true that the GraphC has to be constructed for all the $\alpha$-cuts. But it is done in a very structured way:

- The same $\alpha$-cut is used for all the temporal constraints. The combinatorial explosion of this exploration is strongly reduced in relation to considering all the possible values for the constraints independently.

The other important point is that the fuzzy GraphC provides a synthetic vision of all these computations. In consequence, the designer exactly knows when the property violation occurs and how it occurs.