

---

# Un nouveau graphe de classes pour la préservation des contraintes temporelles quantitatives

Janette Cardoso<sup>1</sup> — Robert Valette<sup>2</sup> — Xiaoyu Mao<sup>3</sup>

<sup>1</sup> IRIT, jcardoso@univ-tlse1.fr

<sup>2</sup> LAAS-CNRS, 31077 Toulouse Cedex 4, robert@laas.fr

<sup>3</sup> IRIT / LAAS, xmao@laas.fr

---

*RÉSUMÉ.* Le but de cet article est de présenter un graphe des classes pour les réseaux de Petri  $t$ -temporels avec sémantique forte qui permette d'associer à tout chemin de ce graphe l'ensemble des séquences de franchissements effectivement franchissables. Cela veut dire que l'on définit exactement (de façon quantitative) les contraintes temporelles que doivent vérifier les dates de franchissement. Après avoir fait quelques rappels sur les réseaux de contraintes temporelles simples, nous définissons les classes d'états puis nous donnons un algorithme de génération du graphe des classes. Nous montrons que ce graphe ne coïncide pas avec les graphes de classes déjà définis ( $W$  et  $A$ ) dans l'outil TINA.

*ABSTRACT.* The objective of this paper is to present a new abstract state space for  $t$ -time Petri nets which associates with each path in this space a sequence effectively fireable in the net. This means that this state space has to exactly (in a quantitative way) define the set of constraints which have to be verified by the firings. After some definitions about the Simple Temporal Networks, the abstract states are defined and an algorithm for the generation of the abstract space are given. It is shown that this space does not coincide with the two previously defined spaces ( $W$  and  $A$ ) in TINA.

*MOTS-CLÉS :* Réseaux de Petri  $t$ -temporels, graphe des classes, réseaux de contraintes temporelles simples, vérification de propriétés temporelles.

*KEYWORDS:*  $T$ -time Petri nets, abstract state space, simple temporal networks, temporal properties verification.

---

## 1. Introduction

Pour la vérification de certaines propriétés des systèmes embarqués critiques, on est parfois amené à considérer des scénarii de fonctionnement spécifiques et à analyser très exactement les contraintes temporelles entre les événements de ces scénarii. C'est par exemple le cas lorsque l'on cherche la durée du pire cas pour atteindre un état donné après un événement donné (une commande par exemple) [Ri 05]. D'autres propriétés concernent par exemple le fait qu'un état est inaccessible et elles impliquent la recherche exhaustive de tous les états d'un système. Quand des contraintes temporelles existent, les états, en nombre infini, sont recouverts par un ensemble fini de classes d'états et l'on construit alors un graphe des classes pour étudier un système. Plusieurs types de graphes de classes ont été élaborés suivant les propriétés étudiées (en temps linéaire ou en temps arborescent) [Me 82, Me 85, Be 04, Ca 05].

Des approches permettent d'obtenir les contraintes temporelles associées à un scénario donné, en particulier dans le cas des réseaux de Petri p-temporels [Ri 03]. Toutefois elles ne sont pas applicables aux réseaux de Petri t-temporels avec sémantique forte. En effet, dans ce cas, la connaissance du franchissement de transition ayant provoqué la sensibilisation d'une transition donnée (celle ayant produit le dernier jeton du n-uplet sensibilisant la transition) est nécessaire, ce qui dans la pratique nécessite de raisonner sur des séquences et non sur des ordres partiels de franchissements de transitions.

Puisqu'il est nécessaire de se fonder sur des séquences et donc de faire apparaître des états, il est alors intéressant de se poser la question suivante : n'est-il pas possible de construire un graphe des classes tel qu'à tout chemin de ce graphe il soit possible d'associer un ensemble de contraintes temporelles délimitant exactement les domaines des dates des franchissements de transition de la séquence correspondante dans le réseau de Petri t-temporel avec sémantique forte ?

Des travaux précédents [Sc 04] ont exploré cette piste avec une approche différente, il s'agissait de chercher à reconstruire les ensembles de contraintes à partir du graphe des classes classique. Ici nous définissons un graphe des classes préservant exactement les contraintes temporelles et nous montrons sur un exemple qu'il ne coïncide pas avec les graphes de classes déjà définis.

## 2. Rappels

### 2.1. Réseau de contraintes temporelles

**Définition 1 (Réseau de contraintes simple)** *Un réseau de contraintes temporelles simple est formé d'un ensemble fini  $V$  de variables  $v_i$  et d'un ensemble fini  $C$  de contraintes binaires  $C_{ij}(v_i, v_j)$  définies sous la forme d'intervalles  $[c_{mij}, c_{Mij}]$  (sans trou, c'est-à-dire convexes) délimitant la distance possible entre les deux variables  $v_i$  et  $v_j$  de  $V$ . Nous avons donc :  $c_{mij} \leq v_j - v_i \leq c_{Mij}$ .*

**Définition 2 (Réseau complet)** *Un réseau de contraintes temporelles simple est dit complet si une distance (délimitée par un intervalle) est associée à chaque couple de variables.*

**Définition 3 (Réseau minimal)** *Un réseau de contraintes temporelles  $(V, C)$  est minimal, ssi  $\forall v_i, v_j \in V$  et  $\forall c_{ij} \in C_{ij}$ , il existe une affectation de valeurs à toutes les variables de  $V$  vérifiant toutes les contraintes et telle que  $v_j - v_i = c_{ij}$ .*

L'algorithme de Floyd-Warshall, calcule le plus court chemin entre deux sommets d'un graphe. Le réseau de contraintes obtenu est complet et minimale si le réseau initial est cohérent [De 91, Gh 04]. Après application de cet algorithme, la distance entre deux variables  $v_i$  et  $v_j$ , délimitée par l'intervalle  $C_{ij} = [d_{mij}, d_{Mij}]$ , est minimale.

## 2.2. Les réseaux de Petri t-temporels

**Définition 4** *Un réseau de Petri t-temporel [Me 82, Be 91, Be 04] est un triplet  $\langle N, M_0, I \rangle$  où :*

- $N = \langle P, T, Pre, Post \rangle$  est un Réseau de Petri,
- $M_0$  : est le marquage initial
- $I : T \rightarrow (Q^+ \cup 0) * (Q^+ \cup 0)$  est la fonction intervalle statique.

La fonction intervalle temporel statique  $I$  associe à chaque transition  $t$  un intervalle de temps appelé intervalle temporel  $[a_i, b_i]$  qui représente l'ensemble des dates de tir possibles de la transition  $t_i$  à partir de sa date de sensibilisation (tous les jetons nécessaires au tir sont présent dans les places d'entrée).

Dans un réseau de Petri t-temporel, les événements à considérer (et donc les variables temporelles associées aux dates de ces événements) sont la *sensibilisation* d'une transition, le *début de l'intervalle* de tir et la *fin de l'intervalle* de tir et enfin le *franchissement* effectif de la transition. Entre ces événements, les contraintes suivantes doivent être vérifiées :

- la date de franchissement doit appartenir à l'intervalle de sensibilisation,
- la date de sensibilisation d'une transition est égale à la date du dernier franchissement ayant contribué à la sensibilisation (et plus grande que la date de franchissement de autres transitions ayant contribué à cette sensibilisation).

Par exemple, considérons la fig. 1 avec le marquage  $p_2p_3p_4$ . La transition  $t_4$  est sensibilisée lors du tir de  $t_2$ . La date de sensibilisation de  $t_4$  est donc égale à la date de tir de  $t_2$  et bien sur plus grande que la date de tir de  $t_1$ .

Ces relations peuvent s'exprimer par des contraintes binaires simples à condition de connaître le franchissement ayant sensibilisé la transition considérée et donc de raisonner vis-à-vis d'une séquence de franchissements dans une sémantique d'entrelacement et non dans une sémantique d'ordre partiel [Ri 03].

La prise en compte de la *sémantique forte* impose, lorsque plusieurs transitions sont franchissables, de franchir l'une de ces transitions avant la fin du domaine de tir des autres transitions. Cette contrainte peut également être représentée par une conjonction de contraintes binaires (entre la date de franchissement de la transition considérée et la fin du domaine de tir des autres transitions).

Donc les réseaux de contraintes temporelles simples sont un cadre adéquat pour analyser les contraintes temporelles générées par les réseaux de Petri t-temporels *dans une sémantique d'entrelacement*. Dans la suite de ce travail, nous ne considérerons que les variables de type  $x_i^k$  (variable associée à la date du  $k^{ième}$  franchissement de la transition  $t_i$  et les variables de type  $y_i$  (associée à la borne supérieure de l'intervalle de tir de  $t_i$  lorsqu'elle est sensibilisée).

### 3. Définition des états et des classes d'états

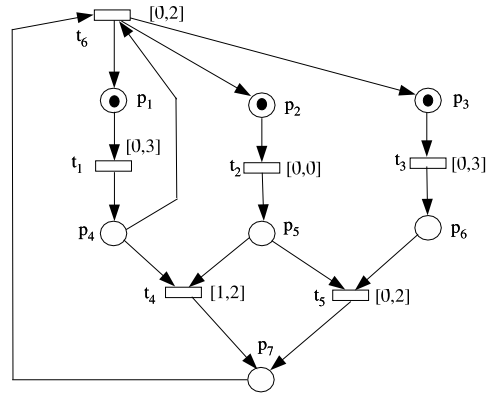
#### 3.1. Etat d'un réseau de Petri t-temporel

Considérons l'exécution d'une séquence de franchissement de transitions  $\sigma = t_i ; \dots ; t_i ; t_j ; \dots ; t_n$  pour un réseau de Petri t-temporel. Une transition peut être franchie plusieurs fois dans une séquence. Supposons que le franchissement de  $t_i$  que nous considérons soit le  $o_i^{ème}$  et celui de  $t_j$  le  $o_j^{ème}$ . Les variables correspondant à ces tirs sont  $x_i^{o_i}$  pour  $t_i$  et  $x_j^{o_j}$  pour  $t_j$ . L'état atteint après le tir de  $t_i$  dans une séquence  $\sigma$  est donné par le marquage associé à la valeur de l'horloge, et les dates de tir de toutes les transitions qui précèdent  $t_i$  dans  $\sigma$  de façon à calculer les nouveaux intervalles de tir pour chaque transition sensibilisée.

#### 3.2. Classe d'états

Nous allons regrouper dans une même classe toutes les exécutions de  $\sigma$  cohérentes avec les contraintes temporelles définies par le réseau de Petri t-temporel (avec sémantique forte, sans mémoire des sensibilisations passées et sans hypothèse du serveur unique pour le franchissement des transitions). La classe, après le franchissement de  $t_i$ , doit donner suffisamment d'information pour permettre le choix des dates de franchissement pour toutes les transitions suivantes, et d'abord pour  $t_j$ . Elle doit donc donner exactement les contraintes temporelles que doivent vérifier la variable  $x_j^{o_j}$  vis-à-vis des variables associées aux franchissements passés dans la séquence.

Pour pouvoir avoir un nombre fini de classes, il faut *oublier* une partie du passé. Au lieu de conserver les variables associées à tous les franchissements passés et les contraintes temporelles qui les lient, nous n'allons garder qu'un fragment de ce réseau de contraintes temporelles. Ce sera l'élément clé de la définition de nos classes. Après avoir défini ce fragment, nous allons donner la procédure de construction des contraintes temporelles que doit vérifier la variable  $x_j^{o_j}$ , puis nous montrerons que les contraintes obtenues ne seraient pas modifiées si l'on prenait en compte toutes les va-



**Figure 1.** Exemple de réseau de Petri t-temporel

riables temporelles et toutes les contraintes passées. Soit un réseau de Petri t-temporel et une séquence de franchissement  $\sigma = t_1; \dots; t_i; t_j; \dots; t_n$  qui définit une infinité de comportements possibles lorsque le temps est pris en compte.

**Définition 5** La classe d'état  $C_i$ , obtenue après le franchissement de la transition  $t_i$  est définie par les éléments suivants :

1) Le marquage atteint  $M_i$ , nous supposons que pour ce marquage  $n$  transitions sont sensibilisées (y compris  $t_j$ ),

2) Le réseau de contraintes temporelles simple  $Rc_i$  comprenant les variables et les contraintes suivantes :

- la variable associée à la date du dernier franchissement de transition  $x_i^{oi}$ ,
- pour chaque transition sensibilisée, la variable associée au franchissement de transition ayant provoqué cette sensibilisation, soit  $x_{si}^{osi}$  ( $i = 1 \dots n$ ).
- toutes les contraintes temporelles liant ces variables sous la forme d'un réseau complet et minimal

Dans le reste de ce travail afin de considérer le cas le plus général, nous mettrons une variable pour chaque transition sensibilisée. Si deux transitions  $t_1$  et  $t_2$  sont sensibilisées par le franchissement d'une même transition  $t_a$ , les variables correspondantes sont les mêmes  $x_{s(1)}^{os1} = x_{s(2)}^{os2} = x_a^{oa}$ . Si le dernier franchissement de la transition  $t_i$ , représenté par  $x_i^{oi}$ , est également un événement sensibilisant une transition  $t_j$ , représenté par  $x_{s(j)}^{osj}$ , alors  $x_i^{oi} = x_{s(j)}^{osj}$ .

### 3.3. Construction du réseau de contraintes délimitant le franchissement de $t_j$

Le réseau de contraintes de la classe reflète la mémoire du passé nécessaire à la caractérisation des dates des événements futurs. Dans les graphes de classes, un arc entre deux classes correspond au franchissement d'une transition. Dans notre cas nous voulons caractériser exactement les contraintes temporelles que doivent vérifier la date de ce franchissement. Il ne suffit donc pas d'associer à cet arc le nom de la transition franchie, ni même le seul domaine de franchissement vis-à-vis de la dernière transition franchie. Nous allons lui associer un réseau de contraintes *Rt complet et minimal* comprenant les variables de la classe origine de l'arc. Pour une séquence donnée, c'est-à-dire pour un chemin dans le graphe des classes, il suffira alors de concaténer les réseaux de contraintes associés à chaque arc en fusionnant les nœuds associés aux variables de même nom, pour avoir le réseau de contraintes de la séquence.

La procédure de construction du réseau de contraintes *Rt* est la suivante :

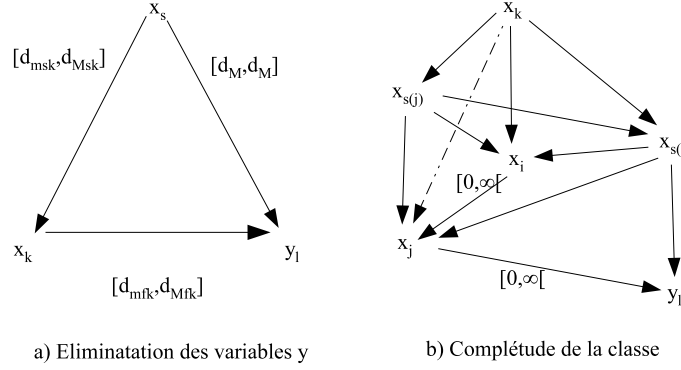
- 1) Au départ,  $Rt = Rc_i$  le réseau de contraintes de la classe origine  $C_i$ ,
- 2) Ajouter à *Rt* la variable  $x_j^{oj}$  associée au prochain franchissement de transition (celui de  $t_j$ ) dans la séquence, ajouter comme contrainte entre cette variable et la variable  $x_{s(j)}^{os(j)}$  associée au franchissement de transition ayant sensibilisé  $t_j$  l'intervalle statique de  $t_j$ ,
- 3) Ajouter à *Rt* les variables  $y_l^{ol}$  ( $l \neq j$ , cardinalité( $l$ ) =  $n - 1$ ) correspondant aux dates maximales de franchissement des autres transitions sensibilisées dans la classe origine, ajouter à chaque couple  $(x_{s(l)}^{osl}, y_l^{ol})$  l'intervalle (singleton)  $[d_{Mfi}, d_{Mfi}]$  correspondant à la borne maximale de l'intervalle statique  $I(t_l)$ ,
- 4) Ajouter la contrainte  $[0, \infty[$  entre les variables  $x_i^{oi}$  et  $x_j^{oj}$  pour exprimer le fait que  $t_j$  doit être franchie *après*  $t_i$ ,
- 5) Ajouter les contraintes  $[0, \infty[$  entre les couples  $(x_j^{oj}, y_l^{ol})$ ,  $l \neq j$ , pour exprimer que  $t_j$  doit être franchie *avant* la borne maximale de chaque transition  $t_l$ ,
- 6) Appliquer l'algorithme de Floyd-Warshall. Si le réseau est incohérent, la transition  $t_j$  n'est pas franchissable,
- 7) Effacer les variables  $y_l^{ol}$  et les contraintes auxquelles elles sont directement reliées.

L'étape 4 est imposée par la sémantique d'entrelacement et l'étape 5 est imposée par la sémantique forte. L'étape 7 est prouvé dans la section 3.4.1.

## 3.4. Preuves

### 3.4.1. Preuve que $y_l^{ol}$ peut être effacé

Une fois l'algorithme de Floyd-Warshall appliqué, les variables  $x_{s(l)}^{osl}$  et  $y_l^{ol}$  sont redondantes car les contraintes les liant sont des singletons. En effet, si l'on considère le triangle de la figure 2.a, il pourra être reconstitué si l'on ne connaît que la contrainte



**Figure 2.** Réseau de contraintes temporelles  $R_t$  illustrant les preuves

liant  $x_{s(l)}$  (événement ayant sensibilisé la transition  $t_l$ ) à  $x_k$ . En effet nous avons nécessairement :

$$d_{mfk} = d_{msk} - d_M \text{ et } d_{Mfk} = d_{Msk} - d_M \quad [1]$$

### 3.4.2. Preuve que le passé peut être oublié

Considérons le réseau de contraintes  $R_t$  de la figure 2.b. Il nous faut montrer que si l'on considère une variable faisant partie du passé oublié comme  $x_k$ , cela ne peut pas permettre de restreindre les contraintes reliant  $x_j$  aux autres nœuds.

Initialement le nœud  $x_j$  n'est relié par des contraintes qu'aux nœuds  $x_{s(j)}$  (événement ayant provoqué sa sensibilisation),  $x_i$  (sémantique d'entrelacement) et  $y_l$  (sémantique forte). Sur la figure nous n'avons représenté que le nœud  $y_l$ , les autres  $y_l$  sont identiques. La contrainte  $C_{kj}$  est initialement égale à  $] -\infty, \infty[$ . Le plus long chemin de  $x_k$  vers  $x_j$  passe donc nécessairement par  $x_{s(j)}$ ,  $x_i$  ou l'un des  $y_l$ . Supposons que ce soit celui passant par  $x_{s(j)}$  (donc  $d_{mkj} = d_{mks(j)} + d_{ms(j)j}$ ). Cette contrainte peut-elle renforcer, par exemple,  $d_{mij}$ ? Dans ce cas nous aurions (un arc  $(x_i, x_j) = [d_{mij}, d_{Mij}]$  correspond à deux arcs  $(x_i, x_j) = d_{Mij}$  et  $(x_j, x_i) = -d_{mij} = d_{mji}$ ) :

$$d_{mij} = d_{mik} + d_{mkj} = d_{mik} + d_{mks(j)} + d_{ms(j)j} \quad [2]$$

$$d_{Mij} = d_{Mik} + d_{Mkj} = d_{Mik} + d_{Mks(j)} + d_{Ms(j)j} \quad [3]$$

Or comme le réseau de contraintes caractérisant la classe est complet et minimal,  $d_{mis(j)} \geq d_{mik} + d_{mks(j)}$  et  $d_{Mis(j)} \leq d_{Mik} + d_{Mks(j)}$ , donc le chemin passant directement par  $x_{s(j)}$  est égal ou plus long que celui passant par  $x_k$  (eq. 2). Les autres cas sont analogues, prenons par exemple le cas où le chemin le plus long entre  $x_j$  et  $x_k$  passe par  $x_{s(j)}$  et celui entre  $x_k$  et  $y_l$  passe par  $x_{s(l)}$  et étudions si la prise en compte de  $x_k$  peut renforcer la contrainte entre  $x_j$  et  $y_l$ . Nous aurions :

$$d_{mjl} = d_{mjk} + d_{mkl} = d_{mjs(j)} + d_{ms(j)k} + d_{mks(l)} + d_{ms(l)l} \quad [4]$$

or  $d_{ms(j)s(l)} \geq d_{ms(j)k} + d_{mks(l)}$ , et donc le chemin  $(x_j, x_{s(j)}, x_{s(l)}, y_l)$  est plus long ou égal que celui passant par  $x_k$ .

### 3.5. Classe restreinte

Après application de l'algorithme de Floyd-Warshall pour  $Rt_j$ , certaines des contraintes du réseau  $Rc_i$  de la classe  $C_i$  peuvent être modifiées. Cela implique que la transition n'est pas franchissable à partir de tous les états de la classe, mais seulement à partir de certains d'entre eux. Cela définit une sous-classe d'une classe, restreinte au franchissement d'une transition déterminée.

Les contraintes entre les événements d'une classe ont nécessairement été obtenues lors de la définition de l'ensemble des contraintes devant être vérifiées par le dernier franchissement (le réseau  $Rc_i$  de la classe  $C_i$  est un fragment du réseau  $Rt_i$  caractérisant la date du franchissement de  $t_i$ ). Cela veut dire que par exemple la restriction  $Cr_i$ , d'une classe  $C_i$ , caractérisant les états à partir desquels la transition  $t_j$  est franchissable correspond aux états atteints par une restriction du domaine de franchissement de la transition  $t_i$  précédant  $t_j$  dans la séquence. Cette restriction du domaine de franchissement de  $t_i$  peut elle-même induire une restriction de la classe  $C_{i-1}$  à partir de laquelle  $t_i$  est franchie et ainsi de suite.

**Définition 6** Une classe restreinte  $Cr_{ij}$  d'une classe  $C_i$  est une caractérisation de l'ensemble des états de  $C_i$  à partir desquels une séquence donnée  $s_j$  est franchissable.

### 3.6. Équivalence de deux classes

**Définition 7** Deux classes sont équivalentes si elles correspondent au même marquage et s'il est possible de définir une bijection  $\mathcal{E}$  entre les variables des deux classes telle que  $\mathcal{E}$  respecte les indices des événements (deux variables vérifiant  $\mathcal{E}$  correspondent à deux franchissements différents de la même transition) et que les contraintes de distance entre deux paires de variables vérifiant la bijection sont les égales.

Bien évidemment, pour que deux classes restreintes soient équivalentes, il faut, de plus, que les séquences franchissables définissant les restrictions soient les mêmes.

## 4. Algorithme

La construction du graphe des classes est donnée par l'algorithme 1, implémenté en Java et disponible à [www.irit.fr/~Janette.Cardoso/feria](http://www.irit.fr/~Janette.Cardoso/feria). La structure de données de cet algorithme utilise quatre classes d'objets : Classe=[identifiant, marquage, Rc, transitions franchissables, antécédents]; Rt=[classe origine, transition, Est\_cohérent,

**Algorithm 1** Construction du graphe des classes

---

```

AlgoGraphC( $M_0, \text{tpn}$ )
N = null { The List of nodes (class, enabled_transition), FIFO}
l=0 {Class Identifier}; arcs = null {List of arcs in the graph}
c = new.Class(l,  $M_0, x_0, \text{Enabled}(M_0, \text{tpn}), \text{null}$ ) {Creation of class  $C_0$ }
for each  $t \in c.\text{trans\_enabled}$  do
  add( $N, (c, t)$ ) {Add the nodes for the initial class  $C_0$ }
end for
while nonempty(N) do
  ( $k, t$ ) = extract(N); s=Floyd_Warshall(Nt_initial( $k, t$ )) {Calculate  $Nt$ }
  if s.IsCoherent then
    FutureCreation( $k, t, Nt(s)$ ) {Create a new class  $c$ , arc ( $k, c$ ) and nodes for  $c$ }
  end if
end while
for each  $ai \in \text{arcs}$  do
  t=ai.transition; cp=ai.source {Is there a restriction in the Nt attached to the arc?}
   $I_R = \text{Restic}(ai.\text{origine}, Rc, ai.Rt)$  {Calculate the restriction, otherwise  $I_R = \text{null}$ }
  if  $I_R \neq \text{null}$  then
    Cr=new.Class(l++, ai.source.marking,  $I_R, t, ai.source.ancestor$ )
    FutureCreation(Cr, t, ai.Nt)
    ai.target.ancestor = ai.target.ancestor - ai {update list of antecedents of ai.target}
    arcs=takeoff(ai); ai.delete
    PastResearch(cp,t, Ir, Cr)
  end if
end for
FutureCreation( $k, t, s$ )
m = Next_m(k.marking, t)
c = new.Class(l++, m, Nc(s), Enabled(m,tpn),.)
 $c_e = \text{EquivClass}(c)$  {Verify whether there are the equivalent classes or not}
if  $c_e \neq \text{null}$  then
  c.delete {There was an equivalent class  $C_e$ }
  a = new.Arc(k,  $c_e, s, t, t.\text{index}$ ) {arc creation};  $c_e.\text{ancestor} = c_e.\text{ancestor} \cup a$ 
else
  a = new.Arc(k, c, s, t, t.index) {arc creation}; c.ancestor = a
  for each  $t \in c_s.\text{enabled\_transition}$  do
    add( $N, (c, t)$ ) {add the nodes for this class}
  end for
end if
PastResearch( $k, t, I_R, c'$ ) {Verify the existence of other restrictions in the past}
for each  $a \in k.\text{ancestor}$  do
   $N'_t = \text{Floyd\_Warshall}(a.N_t \cap I_R)$ ;  $I'_R = N'_t.\text{restriction}$ ;  $c_p = a.\text{source}$ 
  if  $I'_R = \text{null}$  then
     $a' = \text{new.Arc}(c_p, c', R'_t, t, t.\text{index})$  {recursion finished}; arcs = add( $a'$ )
  else
    t = a.transition {restricted class created in the past based on  $I'_R$ }
     $c' := \text{new.Classe}(l++, cp.marking, I'_R, t, cp.ancestor)$ 
     $a'' = \text{new.Arc}(c'', c', N'_t, t, t.\text{index})$  {arc creation}; arcs = add( $a''$ )
    PastResearch( $c_p, t, I'_R, c''$ ) {The arc ( $c_p, k$ ) always exist and so do the ( $c_p, cx$ )}
  end if
end for

```

---

$t.\text{indice}$ ]; Arc=[origine, cible,  $Rt$ , transition]. L'algorithme utilise les fonctions suivantes :

Enabled( $m$ ) : calcule l'ensemble des transitions franchissables pour le marquage  $m$  ;

Nt\_inicial( $k, t$ ) : génère le réseau de contraintes initial correspondant au tir de  $t$  (section 3.3) à partir d'une classe  $k$  ;

Floyd\_Warshall( $r$ ) : calcule, à partir du réseau  $r$ , le réseau de contraintes complet et minimal utilisant l'algorithme de Floyd-Warshall ;

Classe	Marquage	Contraintes ( $Rc$ )	Classe	Marquage	Contraintes ( $Rc$ )
$C_0$	$p_1 \otimes p_2 \otimes p_3$	$x_0$	$C_{13}$	$p_4 \otimes p_5 \otimes p_6$	$0 \leq x_3^3 - x_1^3 \leq 2$
$C_1$	$p_1 \otimes p_3 \otimes p_5$	$0 \leq x_2^1 - x_0 \leq 0$	$C_{16}$	$p_2 \otimes p_3 \otimes p_4$	$0 \leq x_1^4 - x_0 \leq 0$
$C_2$	$p_1 \otimes p_5 \otimes p_6$	$0 \leq x_3^1 - x_0 \leq 3$	$C_{17}$	$p_3 \otimes p_4 \otimes p_5$	$0 \leq x_2^2 - x_0 \leq 0$
$C_3$	$p_4 \otimes p_5 \otimes p_6$	$0 \leq x_1^1 - x_3^1 \leq 2$	$C_{18}$	$p_4 \otimes p_5 \otimes p_6$	$0 \leq x_3^4 - x_2^2 \leq 2$
$C_4$	$p_4 \otimes p_5 \otimes p_6$	$0 \leq x_1^1 - x_3^1 \leq 1$	$C_{20}$	$p_3 \otimes p_7$	$1 \leq x_4^5 - x_0 \leq 2$
$C_5$	$p_6 \otimes p_7$	$x_4^1$	$C_{24}$	$p_2 \otimes p_4 \otimes p_6$	$0 \leq x_3^6 - x_0 \leq 0$
$C_6$	$p_4 \otimes p_7$	$x_5^1$	$C_{25}$	$p_4 \otimes p_5 \otimes p_6$	$x_2^3$
$C_8$	$p_1 \otimes p_7$	$0 \leq x_5^2 - x_0 \leq 3$	$C_{28}$	$p_1 \otimes p_2 \otimes p_6$	$0 \leq x_3^7 - x_0 \leq 0$
$C_9$	$p_4 \otimes p_7$	$x_1^2$	$C_{29}$	$p_2 \otimes p_4 \otimes p_6$	$0 \leq x_1^5 - x_0 \leq 0$
$C_{10}$	$p_3 \otimes p_4 \otimes p_5$	$0 \leq x_1^3 - x_0 \leq 3$	$C_{33}$	$p_1 \otimes p_5 \otimes p_6$	$0 \leq x_2^5 - x_0 \leq 0$
$C_{14}$	$p_3 \otimes p_4 \otimes p_5$	$0 \leq x_1^3 - x_0 \leq 2$	$C_{34}$	$p_4 \otimes p_5 \otimes p_6$	$0 \leq x_1^6 - x_2^5 \leq 2$
$C_{11}$	$p_3 \otimes p_7$	$1 \leq x_4^2 - x_0 \leq 3$	$C_{35}$	$p_4 \otimes p_5 \otimes p_6$	$0 \leq x_1^6 - x_2^5 \leq 1$
$C_{12}$	$p_6 \otimes p_7$	$x_3^2$	$C_{37}$	$p_1 \otimes p_7$	$0 \leq x_5^5 - x_0 \leq 2$

**Tableau 1.** Les classes préservant les contraintes

$\text{Next}_m(m, t)$  : calcule le marquage suivant atteint par le tir de  $t$  à partir du marquage  $m$  ;

$\text{EquivClass}(c)$  : retourne la une classe équivalente à  $c$  le cas échéant, sinon retourne *null* ;

$\text{Nc}(r)$  : retourne le réseau de contraintes de la classe à partir du réseau  $r$  (def. 5).

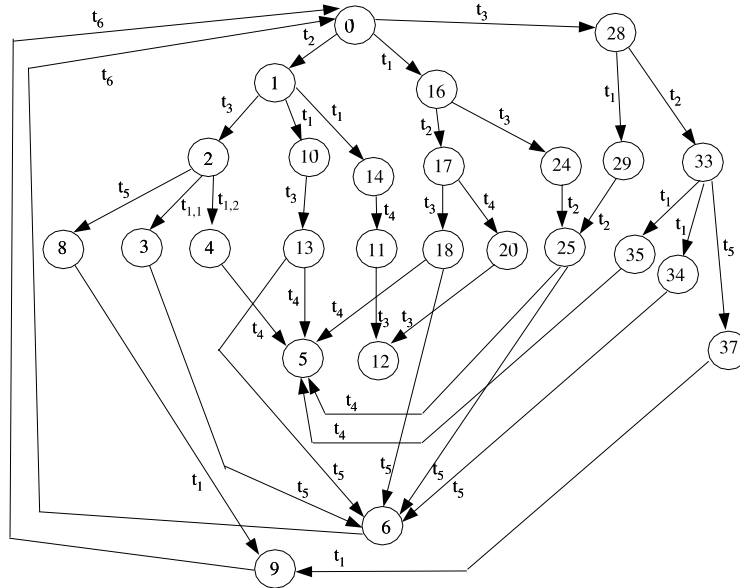
## 5. Exemple

### 5.1. Construction du graphe des classes

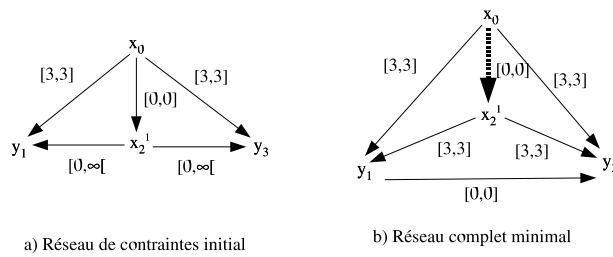
Considérons l'exemple de la figure 1. Il comporte un exemple de blocage mortel mais aussi une possibilité de fonctionnement cyclique et donc de séquences de longueur infinie. Les classes avec leur marquage et leur réseau de contraintes sous forme textuelle sont représentées dans le tableau 1. La représentation du graphe des classes est donnée par la figure 3.

Considérons la séquence  $\sigma_1 = t_2; t_3; t_1; t_4$  dans le graphe de la figure 3. Elle aboutit au blocage mortel  $M_4 = p_6 \otimes p_7$ . La classe  $C_0$  (voir tableau 1), correspondant à l'état initial, est définie par un réseau de contraintes  $R_{C_0}$  formé d'un seul nœud  $x_0$  correspondant à un événement initial de création des jetons du marquage initial. Il représente l'origine des temps.

A partir de cette classe on va appliquer l'algorithme 1. On considère le franchissement de la transition  $t_2$  dans la séquence  $\sigma_1$  à la date  $x_2^1$ . Les contraintes liées à ce franchissement sont définies par le réseau de contrainte  $R_{t_2,1}$  qui initialement est donné par la figure 4.a puis après application de l'algorithme de Floyd-Warshall (permettant d'obtenir un réseau complet minimal) par la figure 4.b. À partir de  $R_{t_2,1}$  on trouve alors la classe  $C_1$ . Son réseau de contraintes  $R_{C_1}$  est l'arc en pointillés gras de la figure 4.b.

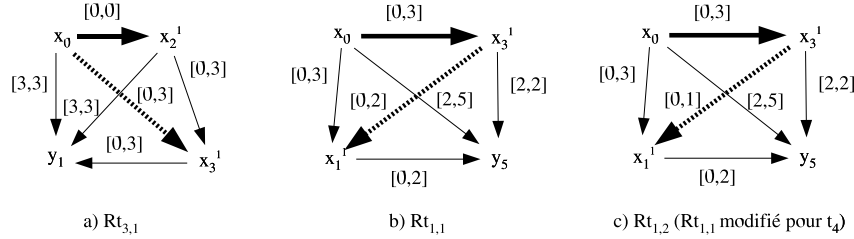


**Figure 3.** Graphe des classes conservant les contraintes des franchisements



**Figure 4.** Réseau de contraintes  $Rt_{2,1}$  pour franchissement de  $t_2$

Maintenant la poursuite de la séquence  $\sigma_1$  implique le franchissement de la transition  $t_3$ . Les contraintes devant être vérifiées par sa date de franchissement sont données dans la figure 5.a. Elles font passer de la classe  $C_1$  (contrainte en gras) à la classe  $C_2$  (contrainte en pointillés gras). Ensuite, on considère le franchissement de la transition  $t_1$  à partir de  $C_2$ . Le réseau de contraintes complet et minimal  $Rt_{1,1}$  délimitant la date  $x_1^1$  de ce franchissement est donné par la figure 5.b. Nous aboutissons à la classe  $C_3$  (pointillés gras dans la figure 5.b).



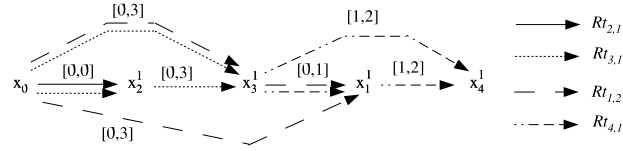
**Figure 5.** Réseaux de contraintes  $Rt_{3,1}$ ,  $Rt_{1,1}$  et  $Rt_{1,2}$

Le franchissement de la transition  $t_4$  à partir de  $C_3$ , est défini par le réseau de contraintes  $Rt_{4,1}$  délimitant la date  $x_4^1$  de ce franchissement. Le  $Rt_{4,1}$  initial est formé par les noeuds  $x_1^1$ ,  $x_3^1$ ,  $x_4^1$  et  $y_5$  et les contraintes  $C_{31} = [0\ 2]$ ,  $C_{35} = [2\ 2]$ ,  $C_{14} = [1\ 2]$  et  $C_{45} = [0\ \infty]$ . Le  $Rt_{4,1}$  complet et minimal présente, pour l'arc  $(x_3^1, x_1^1)$ ,  $C_{31} = [0\ 1]$  au lieu de  $[0\ 2]$ , ce qui représente une restriction temporelle dans la classe  $C_3$ . En effet,  $Rc_3$  est donnée par  $C_{31} = [0\ 2]$  ou  $0 \leq x_1^1 - x_3^1 \leq 2$ . Cela veut dire que si l'on veut pouvoir franchir la transition  $t_4$  après la transition  $t_1$ , il faut restreindre le domaine de tir de  $t_1$ . L'arc  $(x_3^1, x_1^1)$  définit donc une *classe restreinte*, la classe  $C_4$ , avec  $C_4.\text{marquage} = C_3.\text{marquage}$ , et  $C_4.Rc$  donné par  $C_{31} = [0\ 1]$ . Cette classe regroupe tous les états obtenus à partir de l'état initial par le franchissement de la séquence  $t_2$ ;  $t_3$ ;  $t_1$  sachant que la transition suivante  $t_4$  doit pouvoir être franchie.

La classe  $C_3$  avait été définie par le réseau de contraintes  $Rt_{1,1}$  délimitant la date du franchissement de la transition  $t_1$  (arc pointillé gras de la figure 5.b). Il faut donc reconstruire un réseau complet minimal prenant en compte  $C_{31}$  ( $Rt_{4,1}$ .restriction). Cela donne le réseau  $Rt_{1,2} = Rt_{1,1} \cap C_{31}$  de la figure 5.c qui donne les contraintes que doit vérifier la date de franchissement de  $t_1$  pour que la transition  $t_4$  soit ensuite effectivement franchissable. Comme la contrainte de la classe précédente  $C_{03}$  n'est pas modifiée, cette propagation arrière s'arrête ici.

Il faut rajouter  $(C_4, t_4)$  à la liste de noeuds pendants (on ne rajoute pas  $(C_4, t_5)$  parce que l'on s'intéresse seulement au tir de  $t_4$ ) et créer l'arc entre  $C_2$  (classe précédente de  $C_3$ ) et  $C_4$  étiqueté par  $Rt_{1,2}$  de la figure 5.c. Comme la classe  $C_3$  ne change pas, la date du franchissement de  $t_1$  associé à l'arc  $(C_2, C_3)$  est toujours délimitée par  $Rt_{1,1}$ .

Pour retrouver la classe  $C_0$  par le franchissement de la transition  $t_6$  à partir des classes  $C_6$  et  $C_9$ , nous avons confondu l'événement initial  $x_0$  avec un franchissement de la transition  $t_6$ . Sinon il aurait fallu attendre que les classes ne contiennent plus  $x_0$  (comme par exemple  $C_3$  et  $C_4$ ) pour retrouver des classes équivalentes. Cela aurait inutilement augmenté le nombre de classes.



**Figure 6.** Réseau de contraintes pour la séquence  $\sigma 1$

### 5.2. Réseau de contraintes pour une séquence

Les contraintes dans la classe (tableau 1) ne concernent que les contraintes d'écart entre les événements ayant une influence sur le passé. Si l'on veut globalement les contraintes concernant les dates des franchissements de toutes les transitions d'une séquence donnée, il faut concaténer les réseaux  $Rt$  correspondants. Ainsi, pour obtenir le graphe des contraintes temporelles de la séquence  $\sigma 1$  (figure 6) il faut concaténer  $Rt_{2,1}$ ,  $Rt_{3,1}$ ,  $Rt_{1,2}$  et  $Rt_{4,1}$ . Les deux arcs entre  $x_3^1$  et  $x_1^1$  illustrent le fait que le réseau  $Rc_4$  de la classe  $C_4$  est égal à l'intersection de  $Rt_{1,2}$  et  $Rt_{4,1}$ . On peut souligner qu'en plus des fenêtres de tir des transitions, il existe des contraintes supplémentaires qui réduisent l'espace des solutions. Par exemple c'est le cas de la contrainte entre  $x_3^1$  et  $x_4^1$ . Par contre, il se trouve que la contrainte entre  $x_0^1$  et  $x_3^1$  est redondante.

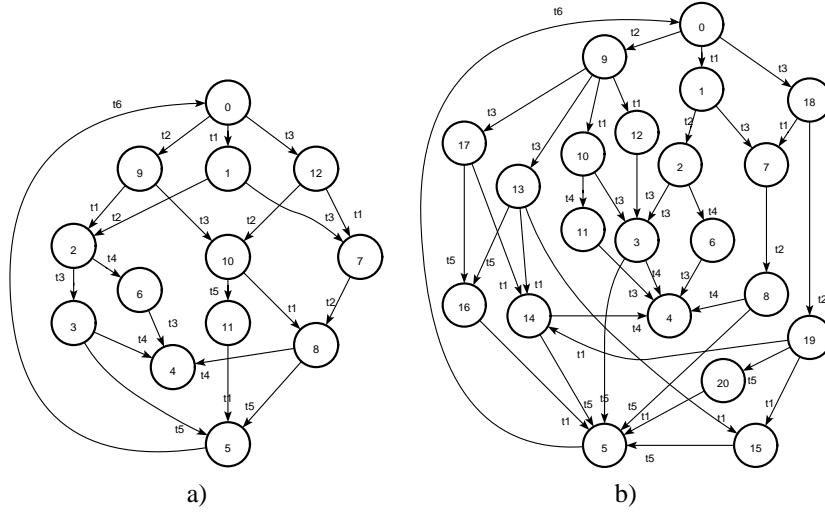
### 5.3. Comparaison avec les classes en mode W et A (Tina)

Nous allons maintenant comparer notre graphe des classes aux deux types de graphe de classes déjà présents dans TINA en nous appuyant sur le réseau de Petri de la figure 1. Les graphes de classes W et A sont donnés par la figure 7.

On notera respectivement  $A_i$ ,  $W_i$  et  $C_i$  les classes  $i$  en mode A, mode W et mode C, le mode C étant le mode que nous venons de présenter. Le mode W donne pour chaque classe, le marquage et le domaine temporel (intervalle de tir pour chaque transition franchissable et certaines contraintes entre les tirs). Le mode A est un raffinement du mode W. Certaines classes du mode A correspondent<sup>1</sup> à des partitions des classes du mode W.

Le mode C est aussi un raffinement du mode W, mais les classes  $C_i$  ne correspondent pas nécessairement à une partition des états à cause des classes restreintes. Nous avons les correspondances suivantes :  $W2=(C_{10}, C_{14}, C_{17})$ ;  $W3=(C_{13}, C_{18})$ ;  $W4=(C_5, C_{12})$ ;  $W5=(C_6, C_9)$ ;  $W6=(C_{11}, C_{20})$ ;  $W7=(C_{24}, C_{29})$ ;  $W8=(C_3, C_4, C_{25}, C_{34}, C_{35})$ ;  $W10=(C_2, C_{33})$ ;  $W11=(C_8, C_{37})$ ;  $W0=C_0$ ;  $W1=C_{16}$ ;  $W9=C_1$ ;  $W12=C_{28}$ .

1. Correspondances :  $W2 = (A_2, A_{10}, A_{12})$ ,  $W6 = (A_6, A_{11})$ ,  $W8 = (A_8, A_{14}, A_{15})$ ,  $W_{10} = (A_{13}, A_{17}, A_{19})$ ,  $W_{11} = (A_{16}, A_{20})$ ,  $W_{12} = A_{18}$ . For  $i = 0, 1, 3, 4, 5, 7, 9$ ,  $W_i=A_i$ .



**Figure 7.** *Graphe de classes avec Tina : a) linéaire (W), b) atomique (A)*

### 5.3.1. Différences dans la prise en compte des évolutions futures

Prenons le cas des classes équivalentes à la classe W8 c'est-à-dire les classes  $\mathcal{C}_3$ ,  $\mathcal{C}_4$ ,  $\mathcal{C}_{25}$ ,  $\mathcal{C}_{34}$  et  $\mathcal{C}_{35}$  dans le mode C et A8, A14 et A15 dans le mode A. La différence essentielle entre le mode A et le mode C provient du fait que la décomposition des classes du mode W en des classes plus petites n'a pas le même objectif lorsque l'on se focalise sur le futur.

Prenons le cas des classes  $\mathcal{C}_3$  et  $\mathcal{C}_4$ . Les états de la classe  $\mathcal{C}_3$  peuvent tous se prolonger par le franchissement de la transition  $t_5$ , indépendamment du fait que  $t_4$  est franchissable ou non. Par contre, la classe  $\mathcal{C}_4$  est une classe restreinte qui a été définie pour délimiter les états qui peuvent être prolongés par le franchissement de  $t_4$ . Bien que  $t_5$  soit également franchissable, il n'y a pas d'arc en sortie de  $\mathcal{C}_4$  étiqueté par  $t_5$  car si on se trouve dans cette classe, c'est précisément parce que l'on veut caractériser les contraintes temporelles d'une séquence qui se poursuit par  $t_4$  et non par  $t_5$ .

Dans le graphe des classes atomiques, ce que l'on doit conserver, ce sont les propriétés en logique CTL. Ce qui compte c'est la différenciation entre les états pour lesquels il y a un conflit entre deux transitions de ceux pour lesquels seule l'une des deux transitions est franchissable. Ainsi par exemple, A14 regroupe les états qui sont tels qu'il y a un conflit entre  $t_4$  et  $t_5$  (les deux sont simultanément franchissables) alors que A15 regroupe tous les états qui ne peuvent se prolonger que par  $t_5$  ( $t_4$  n'est pas franchissable).

Dans le mode A les classes réalisent une partition des états. Un état ne peut appartenir à la fois à la classe A14 et à la classe A15 car les deux situations sont en

exclusion mutuelle. Ce n'est pas du tout le cas dans le mode  $\mathcal{C}$  puisque, au contraire, tous les états de la classe  $\mathcal{C}_4$  sont également des états de la classe  $\mathcal{C}_3$ .

Les états caractérisés par la classe  $\mathcal{C}_3$  (et donc également ceux de la classe  $\mathcal{C}_4$ ) font partie des états regroupés par les deux classes A14 et A15 car ils résultent du franchissement de la séquence  $t_2; t_3; t_1$ . Mais les classes A14 et A15 contiennent également des états résultant de la séquence  $t_3; t_2; t_1$  ce qui n'est pas le cas de la classe  $\mathcal{C}_3$ . Ces états sont regroupés dans les classes  $\mathcal{C}_{34}$  et  $\mathcal{C}_{35}$ . Nous voyons donc que la prise en compte du passé est également différente dans les modes A et  $\mathcal{C}$ .

### 5.3.2. Différences dans la mémorisation du passé

Au vu des considérations concernant la prise en compte du futur, on pourrait penser que notre graphe des classes devrait comporter moins de nœuds que le graphe en mode A, or c'est le contraire qui se produit. Cela vient du fait que notre notion d'équivalence entre classes est plus restrictive que celle du mode A. En effet, pour permettre la reconstruction exacte d'un réseau de contraintes temporelles associé à une séquence de franchissements de transitions, il faut conserver une mémoire du passé suffisante pour assembler correctement les sous-réseaux de contraintes temporelles associés aux franchissements des transitions. Pour les propriétés CTL, seul le futur compte.

Considérons par exemple l'ensemble des états précédant le franchissement de la transition  $t_6$ . En mode W ils sont tous dans une seule classe, la classe W5, que le franchissement précédent soit celui de  $t_1$  ou de  $t_5$ . Dans notre cas, nous avons deux classes,  $\mathcal{C}_6$  et  $\mathcal{C}_9$ . En effet, il est important pour nous de savoir si l'intervalle  $[0, 2]$  délimitant les contraintes temporelles concernant le franchissement de  $t_6$ , a pour origine la date du dernier franchissement de  $t_5$  (une variable  $x_5^i$ , dans  $\mathcal{C}_6$ ) ou la date du dernier franchissement de  $t_1$  (une variable  $x_1^i$  dans  $\mathcal{C}_9$ ).

La mémoire du passé peut remonter au delà du dernier événement. Par exemple, les classes  $\mathcal{C}_{13}$  et  $\mathcal{C}_{18}$  sont différentes bien que l'événement entraînant l'arrivée dans ces classes soit le même : le franchissement de  $t_3$ . En effet, dans le cas de  $\mathcal{C}_{13}$  c'est le franchissement préalable de  $t_1$  qui a provoqué la sensibilisation de  $t_4$ , et sa mémoire doit être conservée. Dans le cas de  $\mathcal{C}_{18}$ , c'est celui de  $t_2$ . En conséquence, les classes  $\mathcal{C}_{13}$  et  $\mathcal{C}_{18}$  ne peuvent pas être confondues pour nous alors qu'elles le sont pour le mode A sous la forme de la classe A3=W3. La différenciation entre les classes  $\mathcal{C}_3$  et  $\mathcal{C}_{34}$  s'explique de la même manière.

## 6. Conclusion

Nous avons montré que si l'on cherchait à caractériser sous la forme d'un graphe des classes les contraintes temporelles devant être vérifiées par les dates de franchissement des transitions d'une séquence de franchissement donnée, nous n'obtenions pas le même graphe des classes que les deux graphes de classes déjà définis, l'un pour les propriétés TTL et l'autre pour les propriétés CTL. La preuve de sa finitude ne devrait pas être différente de celle du graphe des classes pour les propriétés CTL.

Cela ne veut pas dire, que les contraintes ne peuvent pas être obtenues à partir des autres graphes de classes. Mais elles ne peuvent pas être obtenues par une simple concaténation des contraintes rencontrées sur les arcs si l'on veut qu'elles soient complètes et minimales.

Nous allons poursuivre ce travail de différentes manières. Il nous semble que ce travail peut très aisément être transposé aux réseaux p-temporels et également qu'il constitue une base pour la construction d'un graphe des classes dans le cas des réseaux de Petri temporels flous pour l'évaluation des degrés de possibilité de certaines séquences. Nous avons également déjà entamé la définition d'un graphe de classes pour les réseaux de Petri t-temporels qui conserve à la fois les contraintes et les propriétés CTL.

## Remerciements

Nous remercions Christophe Sibertin-Blanc pour les discussions enrichissantes.

## 7. Bibliographie

- [Be 91] B. Berthomieu, M. Diaz : Modeling and verification of time dependent systems using Time Petri nets *IEEE Trans. on Software Engineering*, Vol 17, No 3 p.259-273 1991.
- [Be 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° 14, pp.2741-2756, 15 Juillet 2004.
- [De 91] R. Dechter, I.Meiri, J. Pearl : Temporal constraint networks, *Artificial Intelligence*, vol 49, p.61-91, 1991.
- [Ca 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, Juillet 2005, Prague.
- [Gh 04] M. Ghallab, D. Nau, P. Traverso : Automated Planning Theory and practice, Morgan Kaufman, 2004 ISBN 1-55860-856-7.
- [Me 82] Miguel Menasche : Analyse des réseaux de Petri temporisés et application aux systèmes distribués", Thèse de Doctorat UPS, 1982, Toulouse, France, Rapport LAAS n. 824.
- [Me 85] M. Menasche : PAREDE : an automated tool for the analysis of time Petri nets, International workshop on timed Petri nets Torino July 1985, p. 162-169
- [Ri 03] Nicolas Rivière, Modélisation et analyse temporelle par réseaux de Petri et logique linéaire, Thèse de l'INSA de Toulouse, le 26 novembre 2003 au LAAS, Toulouse
- [Ri 05] N. Rivière, H. Demmou, R. Valette, M. Medjoudj, Symbolic temporal constraint analysis, an approach for verifying hybrid systems, 16<sup>th</sup> IFAC, Prague, July 2005.
- [Sc 04] V. Schastai, E.A. Lima, L.A. Künzle : Sequence analysis for time Petri nets, IFAC 7<sup>th</sup> International Workshop on Discrete Event Systems (WODES'04), France, Sept. 2004.