N° d'ordre : 01/2013-M/IN

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministére de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE DES SCIENCE TECHNOLOGIE
"HOUARI BOUMEDIENE"
FACULTE D'ELECTRONIQUE ET INFORMATIQUE

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER

EN : INFORMATIQUE
Spécialité : Informatique

Par : Abdia HAMDANI

Sujet :

Analyse des Systèmes Temps Réel Préemptifs

soutenu publiquement le 16/09/2013, devant le jury composé de :

M. M. Boukala	Maître de conférences/ A, à USTHB	*Président*
M. A. Abdelli	Maître de conférences/ A, à USTHB	*Directeur de Mémoire*
M. M. Feredj	Maître de conférences/ A, à USTHB	*Examinateur*
Mme. K. Akli	Maître de conférences/ A, à USTHB	*Examinateur*

A mes parents...
En hommage à la personne qui m'a beaucoup appris...
Mr Robert Valette (ancien maître de stage)
A la personne qui m'a beaucoup aidé pour avancer...
Mr A.AbdelKrim (mon encadreur)

Remerciement

En premier lieu, je souhaite remercier Mr. Boukala Mohamed Cherif qui me fait l'honneur d'être le président du jury de ce mémoire, ainsi que Mr. Feredj Mohamed et Mme. Akli Karima pour leur participation au jury. Je tiens à remercier tout particulièrement mon encadreur Mr Abdelkrim Abdelli qui a su me guider sur le chemin adéquat. Je lui suis grée de l'enthousiasme qu'il a manifesté pour mon travail (même lorsque le mien faillissait), et des réponses qu'il a apporté à mes nombreuses interrogations.

je tiens à remercier pleinement Mr Robert Valette. C'est avec lui que j'ai découvert la recherche lors de mon master. Je peux maintenant apprécier le chemin que nous avons parcouru. Je lui dois également une grande partie de ma réussite tant son aide scientifique a comptée pour moi. Ainsi qu'a son intéressement à ce jour de ce que je deviens.

Merci à Bernard Berthomieu et Marie José Huguet du LAAS, qui m'ont beaucoup aidé avec la documentation difficile à obtenir et aussi mon ami du master Oumar Koné.

Grand merci à une personne qui m'a apporté beaucoup d'aide morale pendant des périodes très difficiles : Mokhtar.

Pour terminer je remercie ma famille et mes amis. A chaque fois que j'ai pu les retrouver cela m'a procuré des instants de repos salutaires. Merci en particulier à mes parents mes frères et belle soeur qui m'ont continuellement encouragé.

Résumé

Le but de ce mémoire est de présenter un panorama des méthodes d'analyse disponibles pour les réseaux de Petri T-temporels et discuter leur mise en œuvre avec une étude comparative, d'un autre côté on s'intéresse aux systèmes préemptifs et l'abstraction de l'espace d'états des réseaux de Petri étendus à chronomètres. Nous nous intéressons plus particulièrement au modèle des réseaux de Petri à arcs inhibiteurs $ITPN$ [60](Inhibitors arcs Time Petri Nets). Nous proposons dans la suite, une adaptation simple du graphe de classes en modeC, avec sémantique forte préservant, non seulement les propriétés CTL^* (Computation Tree Logic) mais également les contraintes temporelles quantitatives du modèle pour prendre en considération la préemption.

Cette technique permet d'associer à tout chemin de ce graphe l'ensemble des séquences de franchissements effectivement franchissables. L'ensemble des séquences est alors délimité par un réseau de contraintes temporelles simple STN (Simple Temporal Network), qui définit exactement les contraintes devant être vérifiées par les tirs des transitions.

MOTS CLES : systèmes préemptifs, Réseaux de Petri temporels *RdpT*, Réseaux de Petri à arcs inhibiteurs $ITPN$, classes d'états, propriétés (LTL ; CTL ; CTL^* ; quantitatives), surapproximation DBM.

Abstract

The objective of this report is to present a panorama of the existing methods analyzing the T-time petri nets and discuss their putting works with a comparative study, on one side, on the other side we are interested in preemptive systems and the abstraction of the states space of time Petri nets extended with stopwatches. we are interested by Inhibitors Time Petri Net model [60]. We propose then, a simple adaptation of the graph of classes modeC, with strong semantics preserving CTL^* proprieties (Computation Tree Logic) as well as the quantitative temporal constraints of the model to taking into account the preemption.

This graph allows to associate with each path, a sequence of transitions effectively firables in the net. The set of séquences is then delimited by a simple time network STN (Simple Temporal Network), witch defines exactly the constraints that have to be verified.

KEYWORDS : preemptive system, Time petri nets TPN, Inhibitors arcs Time Petri Nets $ITPN$, state classes, (LTL ; CTL ; CTL^* ; quantitatives) proprieties, DBM ouverapproximation.

Table des matières

Table des figures

Chapitre 1

Introduction Générale

1.1 Le fil conducteur...

Le temps (*du latin tempus*), cette notion fondamentale de la Nature dans son sens de *temps qui passe*. « Les physiciens, écrit Étienne Klein, n'essaient pas de résoudre directement la délicate question de la nature du temps (...). Ils cherchent plutôt la meilleure façon de représenter le temps». Cela vise précisément à mieux appréhender le problème de la mesure physique du temps et des prédictions qui s'y rapportent.

Dans les systèmes complexes, souvent, on parle du *temps réel*. Ce paradigme qui décrit les systèmes qui pilotent un procédé physique à une vitesse adaptée à l'évolution du procédé contrôlé.

Les systèmes informatiques temps réel se différencient des autres systèmes informatiques par la prise en compte de contraintes temporelles dont le respect est aussi important que l'exactitude du résultat, autrement dit le système ne doit pas simplement délivrer des résultats exacts, il doit les délivrer dans des délais imposés.

De tels systèmes sont de plus en plus présents en milieu industriel. On peut citer par exemple : les salles de marché au travers du traitement des données boursières en *temps réel*, dans l'aéronautique au travers des systèmes de pilotage embarqués (avions, satellites), ou encore dans le secteur de la nouvelle économie au travers du besoin, toujours croissant, du traitement et de l'acheminement de l'information (vidéo, données, pilotage à distance, réalité virtuelle, etc.).

Cependant, certains domaines critiques tel que la *médecine*, *l'aéronautique* et *le nucléaire* exigent de pouvoir garantir de façon absolue l'absence de toute défaillance dans certains logiciels, d'où le besoin de méthodes formelles pour la vérification et l'analyse de tels systèmes et ceci constitue le cadre très général de notre travail.

En informatique, un système d'exploitation multi-tâches est un systéme qui permet d'exécuter, de façon apparemment simultanée, plusieurs programmes informatiques. On parle également de multiprogrammation. La simultanéité apparente est le résultat de l'alternance rapide d'exécution des processus présents en mémoire. Alors on peut avoir deux cas de figures :

Le *multitâche coopératif* [66], est une forme simple de multitâche où chaque processus doit explicitement permettre à une autre tâche de s'exécuter. Il a été utilisé, par exemple, dans les produits Microsoft Windows jusqu'à Windows 3.11 ou dans Mac OS jusqu'à Mac OS 9. Ce type de processus présente certains inconvénients par exemple si le processus est bugué, le système entier peut s'arrêter. En pratique, la coopération se passait de façon visible : quand on cliquait sur une fenêtre l'application correspondante prenait la main et les autres étaient figées, en quelque sorte.

Dans un *Multitâche préemptif* [66], le processeur signale au système d'exploitation que le processus en cours d'exécution doit être mis en pause pour permettre l'exécution d'un autre processus. Le système doit alors sauver l'état du processus en cours (le compteur ordinal, les valeurs des registres). Le processus est placé dans une file d'attente ; lorsqu'il est en bout de file, son contexte d'exécution est restauré. Les premiers systémes d'exploitation préemptifs sont : OS2 et OS warp, certains systèmes d'exploitation le sont devenus comme Windows (avec Windows NT 3.1 en 1993 pour les professionnels et Windows 95 en 1995 pour le grand public) et MacOS X en 2001. En pratique, avec ces systèmes récents on peut parfaitement afficher plusieurs vidéos en même temps, même la fenêtre dont la barre apparaît en gris (inactive) continue de fonctionner.

On retient que : *La préemption* est la capacité à *exécuter ou stopper* une tâche planifiée en cours, en respectant les délais et évitant de perdre des données. L'ordonnaceur doit se charger de l'ordonnancement des processus (scheduling) tout en définissant quel processus doit être stoppé et lequel doit continuer à s'executer sur le CPU ?

1.2 Analyse des systèmes temps réel préemptifs

Tout comme les applications *temps réel*, les applications *temps réel préemptifs* étant en général critiques, il est important de détecter d'éventuelles erreurs le plus en amont possible dans la phase de conception afin de minimiser les coûts engendrés par leur correction.

La première méthode utilisée pour traquer les défaillances est le *test*. Le développeur ou une autre personne utilise le programme et prend en compte les erreurs ou les incohérences qu'il rencontre, en essayant de prévoir tous les

comportements que pourra avoir un utilisateur. Cette méthode porte ses fruits dans un premier temps, mais s'avère vite limitée, puisqu'il est difficile d'anticiper toutes les actions que peut effectuer un utilisateur, en particulier si on veut que l'application soit robuste aux actions d'une personne qui ne sait pas comment l'utiliser.

Une deuxième méthode qui apparaît alors naturellement est l'utilisation de *tests aléatoires* intensifs. Un test aléatoire prend en compte toutes les actions qui sont à la disposition d'un utilisateur, et les enchaîne de façon désordonnée tout en s'assurant que le programme se comporte comme il se doit. Mais impossible de tester tous les scénarios, et la plupart du temps le programme comporte encore des bugs à l'issue de cette étape d'où la nécessité de méthodes dites *formelles*. La nécessité de telles méthodes s'est faite sentir depuis longtemps (1960), par *John Backus* qui présentait une notation formelle pour décrire la syntaxe des langages de programmation (notation appelée Backus-Naur form, BNF).

Les méthodes formelles s'appuient sur deux données : l'application informatique, qui est *modélisée* afin de pouvoir être manipulée efficacement, et une *spécification* qui exprime une ou plusieurs propriétés que l'application doit réaliser (voir les sections qui suivent).

Les méthodes formelles les plus courantes sont les suivantes [31] :

Les simulations : Dans ce cas, plutôt que d'executer des tests sur un programme ou sur un matériel informatique, on peut les effectuer sur un modéle formel qui respecte à la lettre son comportement. Ceci permet, en particulier dans le cas d'un systéme matériel, d'éliminer les erreurs de conception avant la production du circuit, plutôt que de les déceler a posteriori et d'assumer le coût de production d'une maquette ou, pire, d'une série défectueuse. Il n'existe à l'issue de cette vérification aucune garantie de l'absence de bugs.

Les preuves automatiques [43] : il s'agit de prouver pas à pas que le programme satisfait la spécification, en prouvant des résultats intermédiaires au fur et à mesure comme lorsqu'on prouve un résultat en mathématiques. Lorsqu'un outil de preuve automatique assure que la spécification est vérifiée, alors le modéle ne peut pas contenir de défaillances. Cependant, il est impossible de construire un outil pouvant trouver une preuve pour tous les programmes corrects.

Cette méthode reste utilisable, mais nécessite en général que le développeur écrive lui même la preuve, qui sera alors simplement vérifiée par l'outil de preuve automatique, ou au moins qu'il fournisse les différents axiomes dont l'outil aura besoin, et guide le processus de vérification. Une telle approche est utilisée dans des outils comme **COQ** ou **PVS**.

Le model-checking [57, 27, 63] : il s'agit d'utiliser un algorithme pour vérifier

directement que le modèle satisfait la spécification souhaitée, quelques soient les comportements de l'utilisateur. Dans ce cas aucune intervention humaine n'est nécessaire. Un outil, le model-checker, s'occupe d'effectuer tous les calculs et de retourner une réponse au problème posé.

En pratique, les méthodes formelles sont utilisées conjointement avec les méthodes classiques de test. La combinaison des deux stratégies permet de multiplier l'éfficacité de la détection d'erreurs.

Nous nous restreindrons par la suite uniquement au cadre du model-checking.

1.2.1 Les enjeux du Model-cheking...

D'une part, il s'agit d'un outil qui s'occupe d'effectuer tous les calculs et de retourner une réponse au problème posé (automatiquement), aucune intervention humaine n'est nécessaire. D'autre part, en cas de réponse négative (c'est-à-dire si le modèle ne satisfait pas la spécification), le model-checker retourne un contre-exemple : une exécution précise du modèle qui ne satisfait pas la spécification. Ce contre-exemple est particulièrement utile, parce qu'il permet de situer la source de l'erreur dans un modèle souvent complexe.

Il est à noter que le model-checking ne s'applique pas directement sur le système mais plutôt sur le modèle représentant ce dernier.

Le modèle, un objet symbolique dont le comportement est aussi proche que possible de celui du programme, et qui est particulièrement fidèle dès que des données ayant un lien avec la spécification sont en cause. Un modèle peut être vu comme une abstraction du système telle que la réponse à la question est la même pour le système et son modèle. Cela implique qu'un modèle n'est valable que pour un ensemble de propriétés à vérifier et ne peut refléter complètement le comportement réel du système. Un modèle peut aussi être construit à partir d'un simple cahier des charges qui décrit de façon informelle un algorithme.

l'intérêt se porte sur les modèles dit *temporisés* pour lesquels le temps est manipulé de manière explicite. Plus précisément, on parle des trois principales familles de modèles temporisés que sont : les extensions au temps des algèbres de processus, des automates finis notés (TA) et des réseaux de Petri (nous nous intéressons aux systèmes modélisés au moyen des réseaux de Petri et leurs extensions au temps *RdPT* [52] représentés dans le Chapitre.3).

Une propriété peut être vérifiée en explorant l'espace d'état du modèle, totalement ou en partie. Cette exploration est cependant soumise au problème bien connu de l'explosion combinatoire : la taille de l'espace d'état augmente trés rapidement avec la taille du système. D'un point de vue théorique, les algorithmes

de vérification ont donc en général une complexité assez élevée et, pour un certain nombre de modéles, leur terminaison est indécidable en particulier pour les systèmes avec préemption.

Il a été prouvé qu'en temps dense, l'accessibilité d'etat et de marquage sont des problémes indécidables sur ces réseaux, même lorsqu'ils sont bornés [12, 13]. De fait, le calcul de l'espace d'etats pour ces modèles passe par des semi-algorithmes surapproximant (voir le chapitre 4).

La spécification, exprime, dans un contexte formel imposé par le model-checker, une ou plusieurs propriétés que le modèle doit vérifier. Une propriété à vérifier est exprimée en *logique temporelle* (une extension de la logique conventionnelle).

Selon le type du système à vérifier, et les besoins attendus, on peut classer les propriétés en deux grandes catégories :

Les propriétés qualitatives qui concernent le fonctionnement du système à vérifier :

La sûreté : exprime le fait que : *"quelque chose de mauvais ne doit pas se produire"*, ou *"ne se produit jamais"*.

La vivacité : exprime le fait que : *"quelque chose finira par avoir lieu"*.

L'atteignabilité : exprime le fait que : *"une certaine situation peut être atteinte"*.

Le blocage : exprime le fait que : *"Le système ne peut plus évoluer"*. Cette propriété décrit un comportement qui met le système dans un état ou il ne peut rien faire.

L'équité : exprime le fait que : *"quelque chose aura lieu"* ou *"n'aura pas lieu"*, un nombre infini de fois.

Il est nécessaire de distinguer entre les états bloquants (blocage non souhaité), et les états de terminaison (états finaux).

Les propriétés quantitatives recouvrent des questions de performances comme la durée (minimale, maximale) d'attente d'un service ou le temps de la réponse bornée.

1.2.2 Expression d'une propriété

Une fois le système décrit sous la forme d'un modéle (réseaux de Petri temporels ou réseaux de Petri à chronométres), il importe de formaliser les spécifications

de sa correction. Nous présentons ici les deux principales méthodes permettant de s'acquitter de cette tâche : l'expression de propriétés sous la forme d'observateurs ou d'une logique dédiée.

1.2.2.1 Observateurs

Un observateur est un réseau de Petri temporel, qui est ajoutée au réseau initial de maniére non intrusive (c'est-à-dire que l'observateur ne doit pas modifier le comportement du système initial) et qui permet de déterminer la valeur de vérité d'une propriété [62]. Celle-ci est alors donnée par l'occurrence ou non d'un marquage ou du tir d'une transition dans l'espace d'états du système « Réseau de Petri temporel (à chronométres) observé + observateur ».

Cette démarche présente l'avantage de transformer la propriété à vérifier en un probléme d'accessibilité de marquage ou d'exécution de trace. Elle souffre néanmoins de plusieurs limitations. D'une part, il n'existe pas de techniques automatiques de génération d'observateurs. Or il est parfois délicat de ramener le probléme de vérification d'une propriété à un probléme d'accessibilité. D'autre part, chaque propriété requiert un observateur spécifique et donc un nouveau calcul de l'espace d'états. Enfin, l'observateur est souvent de taille aussi importante que le modèle initial, accroissant ainsi de façon non négligeable le coût de la vérification de la propriété, des méthodes altérnatives plus efficaces sont présentées dans le chapitre 3, pour la vérification de propriétées quantitatives.

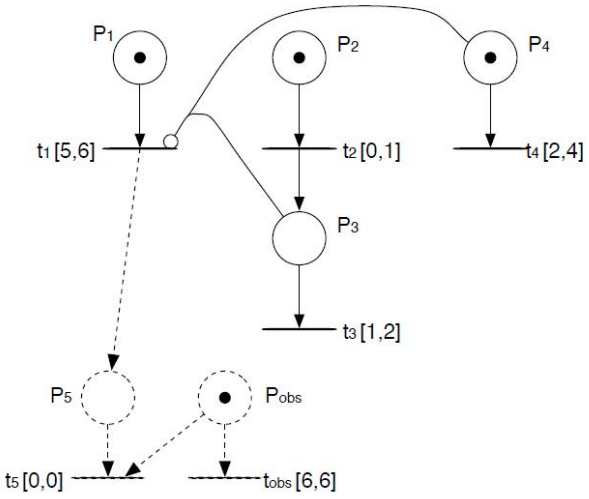


FIGURE 1.1 – Exemple d'observateur pour un réseau de Petri temporel à hyperarcs inhibiteurs

Le réseau de Petri temporel à hyperarcs inhibiteurs de la figure Fig. 1.1

représente un réseau de Petri temporel (traits plein) et un observateur (tirets). L'observateur modélise la propriété suivante : « le temps de séjour dans la place P_1 est toujours inférieur ou égal à 6 unités de temps ». Cette propriété est trivialement fausse, ce qui se traduit par l'occurrence du tir de t_{obs} dans le graphe des classes d'états du réseau avec l'observateur.

1.2.2.2 La logique temporelle

Un peu d'histoire...

Amir Pnueli, est le premier fondateur introduisant la logique temporelle en informatique pour la vérification des programmes et systèmes. Ainsi, Penueli a défini en 1977 [56], la logique temporelle linéaire LTL qui permet l'expression des propriétés sur les comportements linéaires d'un système.

Par la suite, la logique temporelle arborescente CTL a été introduite par Clark et Emerson [27]. Cette logique permet d'exprimer des propriétés sur des arbres d'exécution. Tardivement CTL* introduite en 1986 constituant une classe plus générale des deux derniéres et permettant d'exprimer des propriétés sur les chemins aussi bien que sur les arbres d'execution [26]. On peut écrire : LTL + CTL $\subseteq$ CTL*.

Une autre logique temporelle avec un temps explicite a été proposée [6] (TCTL : Temps + CTL).

Tous comme les propriétés, les principales classes de logiques sont :

La Logique temporelle non Quantitative :

- La Logique du temps linéaire comme LTL (Linear Temporal Logic).
- La Logique du temps arborescent comme CTL (Computing Temporal Logic) et CTL* . Plusieurs sous-classes de CTL ont été proposées :
  - ACTL (Universal Computation Tree Logic) ;
  - ECTL (Existential Computation Tree Logic) ;

Cette logique permet d'exprimer des propriétés plus complexes que de simples propriétés d'accessibilité. Cependant elle ne permet pas d'exprimer des propriétés temporelles quantitatives pour des systèmes temporisés. C'est pour pallier cette limitation que des logiques temporelles quantitatives ont été introduites.

La Logique temporelle Quantitative :

TCTL (Time Computation Tree Logic) est une extension de CTL obtenue en ajoutant des intervalles de temps aux modalités. Ces intervalles indiquent des

contraintes de temps sur les formules.

Il est à présent possible d'exprimer la propriété suivante : (quel que soit l'état du système, si une alarme se déclenche alors un signal est inévitablement émis en moins de 5 unités de temps).

1.2.2.3 Le pouvoir d'expression d'une logique

Du point de vue expressif, les logiques LTL et CTL sont incomparables, chacune d'entre elles permettant de décrire des propriétés qui ne sont pas exprimables dans l'autre. Par exemple, du fait de l'absence des quantificateurs sur chemins, LTL ne permet pas de discerner l'atteignabilité potentielle de l'atteignabilité inévitable, qui s'expriment facilement en CTL.

Inversement, du fait que les quantificateurs sur chemins sont attachés aux modalités linéaires, CTL ne permet pas d'exprimer les propriétés nécessitant l'imbrication des modalités de chemins (notamment, certaines propriétés d'équité) : Par exemple, le fait qu'une propriété est vérifiée continuellement sur chaque chemin du modèle (sauf éventuellement sur un préfixe fini de ce chemin), exprimé en LTL et qui n'a pas de formulation équivalente en CTL. Ces propriétés sont implémentées dans certains l'outils comme : *TINA* [11] *ROMEO* [36] et *GraphC* [21, 20].

Nous n'irons pas plus loin en ce qui concerne la logique (cette derniére est trés importante pour comprendre le chapitre.3, les constructions des graphes déterminant l'éspace d'état des *RdPT* sont classées selon les propriétées conservées par le graphe obtenu), sémantique et syntaxe sont en ANNEXES.

1.3 Les modèles temporisés

Les modèles formels temporisés tels que les automates temporisés (TA) [5] et les réseaux de Petri T-temporels (RdPT) [52] permettent de modéliser l'évolution d'un système suite à des actions discrétes ou à l'écoulement continu du temps. Le temps est représenté de façon dense ce qui implique que ces modèles ont en général un espace d'état infini.

L'espace d'état peut être représenté en un nombre fini de groupes d'états partageant une ou plusieurs propriétés *intéressantes*, par exemple, un même marquage pour un RdPT ou une accessibilité par la même séquence de transitions discrétes.

1.3.1 Les automates temporisés

Les automates temporisés étendent les automates finis classiques avec des horloges explicites [5]. Dans ce modèle, nous pouvons soit rester dans la même

localité et laisser écouler le temps, soit prendre une transition discréte et effectuer l'action associée et éventuellement remettre certaines horloges à (0). Pour chaque transition est définie une garde qui contraint les horloges de l'automate. Cette garde doit être vérifiée pour que la transition puisse être franchie. Henzingeret et al [42] étendent ce modèle en 1994 à l'aide de contraintes sur les horloges, appelées *invariants* associés aux localités. Le séjour dans une localité n'est possible que si l'invariant associé est vérifié ce qui permet de modéliser l'urgence.

1.3.2 Les réseaux de Petri t-Temporels étendus

Les deux extensions temporelles principales des réseaux de Petri sont les réseaux de Petri temporisés [58] et les réseaux de Petri temporels [52]. Pour les premiers, le temps est représenté par des durées minimales (ou exactes dans le cas d'un fonctionnement *au plus tôt* du réseau) de franchissement des transitions. Pour les seconds, le temps prend la forme d'un intervalle contraignant les instants de tir des transitions. Les réseaux temporisés constituent une classe de réseaux incluse dans les réseaux de Petri temporels. Les réseaux de Petri temporels à flux (*Time Stream Petri Nets*) [32] sont également une extension temporelle des réseaux de Petri où un intervalle de temps est associé aux arcs introduits pour modéliser les applications multimédia.

Différents types de réseaux temporels ont été introduits. Ils différent par l'élément auquel est associée la notion de temps : il peut tout aussi bien s'agir des arcs (réseaux de Petri A-temporels [29]), des places (réseaux de Petri P-temporels [45]) ou des transitions (réseaux de Petri T-temporels [52]). Pour les réseaux de Petri A-temporels et P-temporels, la sémantique forte conduit à des jetons qui ne sont plus utilisables et qu'il faut faire *mourir* ce qui est parfois difficile à interpréter. Les réseaux de Petri T-temporels sont ainsi les plus utilisés pour la modélisation des systèmes temps réel mais sont insuffisants pour modéliser et vérifier les applications temps réel avec préemption.

1.3.3 Les modèles temporisés étendus à chronomètres

Dans les modèles temporisés sus-cités, le temps s'ecoule de façon identique pour toutes les composantes du système. Ce qui ne permet pas de modéliser la politique de préemption où l'exécution d'une tâche est interrompue et reprise au même endroit un peu plus tard. C'est pourquoi il est nécessaire d'étendre ces formalismes, sous la forme de « *modèles temporels à chronométres* ».

Plusieurs modèles intégrant la notion de chronométre ont été définis. Cassez et Larsen [23] ont proposé une extension des automates temporisés : les automates à chronométres ; ils sont définis comme une sous-classe des automates

hybrides linéaires [61] pour laquelle les dérivées par rapport au temps des variables ne peuvent prendre que deux valeurs exprimant la progression (1) ou la suspension (0) du temps. Le probléme de l'accessibilité a été prouvé indécidable sur ce modèle.

Dans le cas des réseaux de Petri temporels, plusieurs extensions ont vu le jour : les Scheduling-TPNs [59], les Preemptive-TPNs [19] (ces deux modèles ajoutent ressources et priorités au formalisme des réseaux de Petri temporels respectivement associées aux places et transitions ; mais ont l'inconvénient de ne pas permettre la modélisation de relations de priorités circulaires), et les réseaux de Petri temporels à hyperarcs inhibiteurs [60]. Les $ITPN$ introduisent des arcs inhibiteurs qui contrôlent la progression temporelle des transitions. Ces trois modèles appartiennent à la classe des réseaux de Petri à chronométres [13]. Nous considérons dans ce mémoire, les $ITPN$ comme modèle de travail car bien souvent utilisés en raison de leur puissance d'expression comparativement aux autres modèles. Toutefois, il est sous-entendu que les techniques présentées et developées ici peuvent aussi s'appliquer aux autres modèles.

1.4 La Détermination de l'espace d'état

La méthode classique d'analyse des réseaux de Petri T-temporels est *le graphe des classes d'états* [9] développée en [10]. Dans cette méthode, l'espace d'états est partitionné selon la relation d'équivalence suivante : deux états sont en relation si et seulement s'ils sont accessibles par une même séquence de transitions discrétes. En particulier, ils partagent le même marquage. Le graphe des classes d'états préserve les marquages et le langage discret du graphe des états accessibles. Il est donc adapté à la vérification de l'accessibilité de marquage et de propriétés LTL. Par contre, il ne préserve pas la structure de branchement du graphe des états et ne permet pas la détermination des contraintes temporelles quantitative, ce qui rend impossible la vérification de propriétés CTL* ou TCTL du modèle.

Pour remédier au premier probléme (propriétés CTL*), *Yoneda et Ryuba* proposent dans [67] un autre partitionnement de l'espace d'états en *classes atomiques*. Dans une classe atomique, tout état a un successeur dans chacune des classes atomiques obtenues par tir d'une transition. Les classes atomiques de Yoneda et Ryuba sont obtenues en découpant les classes d'états (définies de façon différente de [10]), par l'ajout de contraintes linéaires (voir chapitre.3). Les auteurs prouvent que le graphe des classes atomiques permet la vérification de propriétées CTL*. Cependant cette technique n'est pas applicable aux réseaux pour lesquels les intervalles de temps ne sont pas bornés.

Par ailleurs, *Lilius* a proposé une technique plus améliorée [46] permettant l'utilisation de techniques d'ordre partiel développées pour les systèmes non tem-

porisés.

Dans [8], *Berthomieu et Vernadat* proposent une construction alternative des classes d'états atomiques, qui fournit un graphe plus compact. Le graphe obtenu se calcule plus rapidement et est applicable aux réseaux dont les intervalles de temps ne sont pas forcément bornés.

Hanifa Boucheneb et *Rachid Hadjidj* [14, 15, 16] ont proposé une autre méthode plus efficace en matière de rapidité et de performance, ainsi le calcul du graphe des classes préservant les propriétées CTL^* se fait en complexité $O(n^2)$ au lieu de $O(n^3)$ (n étant le nombre de transitions sensibilisées).

Pour remédier au second probléme (propriétés $TCTL$), *Rachid Hadjidj* et *Hanifa Boucheneb* [39] ont proposé une sous-classe de cette logique notée : ($TCTL_{TPN}$), ajoutant un $RdPT$ au $RdPT$ initial pour aider à capturer (relever) le temps d'un événement. une exploration à la volée est alors appliquée sur le graphe construit afin de vérifier les propriétées temporelles.

Cependant, il n'est pas toujours possible d'associer à un chemin dans le graphe une séquence *effectivement franchissable* dans le système. Dans le cas où cela est possible, les contraintes que doivent vérifier les dates de franchissement des transitions ne sont pas directement données.

R. Valette et J.Cardoso [21, 20] se sont posé la question de construire un graphe de classe tel qu'a tout chemin de ce graphe il soit possible d'associer un ensemble de contraintes temporelles délimitant exactement les domaines de dates de franchissement de transition de la séquence correspondante dans le réseau de Petri. Cette information est nécessaire pour connaître le degré de possibilité de franchissement de la transition (il faut le min de la possibilité de l'état de départ avec celui de l'intervalle possible (réel) de tir à partir de cet état). La seule information donnée pour un intervalle de tir potentiel ne suffit pas.

Ces traveaux ont été introduit pour le cas d'un $RdPT$ avant de s'attaquer aux réseaux de Petri temporels Flous (fuzzy petri net : FTPN), où l'information est imprécise et floue. Une construction nécessitant l'introduction du formalisme des réseaux de contraintes temporelles (*STN-Simple Temporal Network*) ; ces travaux sont introduit dans l'outil *GraphC* [21, 20]. Ces derniére méthodes constituent en dernière partie le chapitre.3.

Par ailleurs, *E.Vicario* a proposé dans [64] un algorithme pour le calcul de la réponse du temps borné pour n'importe quelle séquence du graphe. Une fois la séquence de franchissement de transitions identifiée, un algorithme récursif permet de calculer le temps minimum et maximum de la réalisation de cette séquence.

D'autres abstractions de l'éspace d'état d'un $RdPT$ ont vu le jour, basées essentiellent sur la notion de *régions géométrique ou zones*, ces méthodes proposées initialement pour les TA et adaptées pour les $RdPT$ constituent une autre partie

du Chapitre.3 avec les travaux de [37, 38, 18].

Toujours pour ce même objectif, avec le principe des (zones), une autre approche basée sur la vérification des propriétées $TCTL$ en établissant une traduction des réseaux de Petri vers les automates temporisés a été proposée. Les traductions se divisent en deux catégories : d'une part, les *traductions structurelles* d'un réseau de Petri temporel non nécessairement borné à dates de tir au plus tard finies ou infinies vers un automate temporisé [24, 25] ; d'autre part les traductions avec calcul de l'espace d'état qui consiste à obtenir l'éspace d'états d'un réseau de Petri temporel sous la forme d'un automate temporisé [33, 48].

D'autres sont allés un peu plus loin en définissant une nouvelle logique $TCTL$ (*native* : sans passer par les TA intermédiaire), pour les réseaux de Petri temporels en temps dense, appelée $TPN - TCTL$ [18], dans laquelle les opérateurs sont étendus avec un intervalle spécifiant une contrainte de temps sur une séquence de tirs. Les auteurs ont prouvé la décidabilité de $TPN-TCTL$ sur les TPN bornés et ont encadré sa complexité. Il ont aussi proposé une méthode en avant et à la volée de vérification d'une sous-classe de $TPN-TCTL$ sur les $RdPT$ consistant à calculer et explorer une abstraction de l'espace d'états. Cette abstraction est basée sur la notion de *zone* [37, 38] et la méthode est inspirée de celle mise au point pour les automates temporisés. Comme pour les automates temporisés, cette abstraction peut nécessiter l'emploi d'un opérateur de surapproximation des zones (connue sous le nom d'opérateur de *normalisation* ou de *k-approximation*) [18], pour assurer que son calcul finisse.

Comme pour les $RdPT$, la technique du graphe de classes [9, 10] est appliquée pour le calcul de l'espace d'état des modèles avec chronomètres. Cette technique exprime chaque classe, notée E, accessible dans le graphe, par une paire constituée d'un marquage M et d'un système de contraintes D [12, 13].

Cependant, au contraire d'un $RdPT$ où le système de contraintes D peut toujours être encodé sous forme DBM (Difference Bound Matrix) [34], dans les modèles avec chronomètres, cette forme ne peut être préservée au fur et à mesure que le processus d'énumération se prolonge. En fait, D prend une forme générale, dite *polyédrique*, dont la représentation minimale [7] est donnée par la conjonction de deux sous-systèmes : l'un peut être encodé en DBM l'autre NON. Ceci nécessite des structures de données plus complexes pour la representation d'une classe en mémoire et requiert un temps de complexité important pour son calcul (voir Chapitre 4). Comme conséquences à cette complexité, les approches exactes basées sur la représentation en polyèdres [47] induisent des temps de calcul prohibitifs, voire des débordements de mémoire, lorsque les systèmes D prennent des formes élaborées.

Contrairement aux $RdPT$ où le graphe des classes est prouvé fini lorsque le marquage du réseau est borné, ceci devient faux en présence de systèmes polyèdriques. En d'autres termes, le nombre de systèmes polyédriques que l'on

peut générer peut être infini alors que celui des systèmes DBM est toujours fini.

Pour contourner ces problèmes, des techniques alternatives de sur-approximation [19, 60, 3] ont proposé d'ignorer systématiquement les contraintes polyédriques, pour représenter une surapproximation DBM de l'espace d'état couvert par la classe E. Ces approches permettent de construire efficacement un graphe en un temps réduit, pouvant cependant dériver des séquences de tirs supplémentaires, non accessibles dans le graphe exact.

Une approche hybride a été proposée par [51]. Cette dernière exploite une condition suffisante permettant de détecter les cas où le sous-système non encodé en DBM devient redondant dans D. A partir de là, la combinaison des deux représentations permet de construire le graphe de classes exact, en réduisant sensiblement les temps de calcul et l'espace mémoire alloué comparativement à l'approche basée sur une représentation exclusivement polyédrique [47].

Plus récemment, une nouvelle approche exploitant l'approximation des grilles a été proposée [13]. Cette dernière calcule une approximation des coefficients du polyèdre produisant un graphe exact dans la majorité des cas, avec un coût inférieur à celui généré par l'approche hybride [51] . Néanmoins, cette technique reste encore coûteuse en temps de calcul et en occupation mémoire comparée à la sur-approximation par DBM, même si elle produit des graphes moins grossiers.

Dans un travail plus récent [2], l'auteur a pu montré qu'en représentant toutes les contraintes DBM du système même celles redondantes, il arrive à définir un algorithme moins complexe pour le calcul d'une classe. Il devient alors possible d'inférer une relation d'équivalence moins restrictive que l'égalité, produisant ainsi des graphes plus compacts mais pas forcément plus précis. Par ailleurs, l'auteur propose dans un autre travail [1] comment calculer une surapproximation des propriétés quantitatives de n'importe quelle séquence de franchissement grâce à l'application d'un algorithme récursif de complexité linéaire.

Dans ce mémoire, et plus précisement dans le chapitre.4, nous nous intéressons en particulier à la méthode proposée dans [3] et l'algorithme permettant de construire la plus petite sur-approximation DBM de l'espace d'état d'un système préemptif modélisé au moyen d'un $ITPN$. Une méthode qui peut être appliquée à n'importe quel autre modèle temporel étendu aux chronomètres.

1.5 Problématique et Contribution

Dans ce mémoire, nous nous intéressons aux réseaux de Petri T-temporels (TPN) comme formalisme de modélisation de base pour les systèmes temps réel. Plus précisément on s'interesse aux $ITPN$ qui nous permettent une représentation à la fois élégante et appropriée pour modéliser les systèmes temps réel avec

préemption.

Cependant, les horloges d'un TPN prenant leurs valeurs dans l'ensemble des réels, l'ensemble des états accessibles d'un TPN est en général infini, même lorsque le réseau est borné. Il est donc nécessaire de recourir à des méthodes permettant de calculer une abstraction fini de l'espace d'états.

La méthode classique de vérification sur ce modèle étant *la méthode du graphe des classes d'états [10].* Une telle méthode permet de vérifier l'accessibilité de marquage et les propriétés linéaires (LTL). Elle a été adaptée également pour la préservation des propriétés arborescentes CTL^* [11, 14, 15] aussi bien qu'a la préservation des contraintes temporelles quantitatives [39, 21, 20].

Ainsi, comme pour les TPN, la technique du graphe de classes [10] est appliquée pour le calcul de l'espace d'état des modèles avec chronomètres $ITPN$ [12, 13].

Aprés une présentation détaillée des différentes méthodes pour l'analyse des systèmes temps réels et temps réel préemptifs, déterminant ainsi l'éspace d'etat des TPN respectivement $ITPN$, notre contribution se résume dans l'adaptation de la méthode dite : *modeC* pour la construction d'un graphe de classes préservant les propriétées qualitatives CTL^* aussi bien que les contraintes temporelles quantitatives du modèle étendu à chronométres.

1.6 Plan du manuscrit

En plus de cette introduction qui décrit le cadre très général de notre travail, la suite du rapport comporte :

- Le chapitre 2 présente les modèles temporels pour l'analyse des systèmes temps réels et temps réel préemptifs. On s'intéresse plus particulièrement au réseaux de Petri à arc inhibiteurs qui ont l'avantage de contrôler la progression du temps [60]. On y présente également une brève classifications des différents modèles des réseaux de Petri étendus à chronomètres. Enfin, nous situons, en paralléle de cette classification, le modèle utilisé dans ce manuscrit ($ITPN$) par rapport aux autres.

- Le chapitre 3 présente, un panorama des méthodes classiques pour le calcul de l'espace d'états des réseaux de Petri temporels ($RdPT$) ; ces méthodes basées essentiellement sur la notion de classes d'états (on présente en particulier et en détail une méthode toute jeune dite en modeC) ; mais également des abstractions basées sur les zones. La suite du chapitre présente une comparaison des différents graphes obtenus. Il est à noter que la présentation des

méthodes a été organisée selon le type de propriétés préservées.

- Le chapitre 4, revient sur les méthodes pour le calcul de l'espace d'états appliquées aux réseaux de Petri avec chronomètres, nous illustrons plus particuliérement la problématique des structures de données qui ne sont pas forcément encodées en DBM. Dans ce cas on s'interesse au modèle des réseaux de Petri à arcs inhibiteurs ($ITPN$).

- Le chapitre 5 présente, notre contribution pour l'adaptation de la méthode en modeC au cas d'un $ITPN$, la deuxième partie de ce chapitre compare les performances des algorithmes exposés précedement dans le mémoire en s'appuyant sur les résultats des expérimentations réalisés.

- Enfin dans le chapitre 6, nous concluons sur nos perspectives qui s'inscrivent en partie dans la continuité de ce mémoire.

La fin de ce rapport comprend des annexes complémentaires.

Chapitre 2

Les Réseaux de Petri et leurs extensions temporisées

2.1 Introduction

Nous introduisons, dans ce chapitre les réseaux de Petri T-temporels et leurs extensions à chronomètres . Nous faisons d'abord un état de l'art sur le modèle classique (RdP) [55] avant de donner la syntaxe et la sémantique du modèle introduisant la notion du temps (RdPT) [52]. Dans ce modèle le temps est associé aux transitions. Enfin nous présentons le modèle des réseaux de Petri étendu à chronomètres, un outil facilitant modélisation et analyse des systèmes temps réels préemptifs. Nous nous intéressons par la suite du manuscrit au modèle (ITPN) pour Inhibitor arc Time Petri Net [60].

2.2 Les réseaux de Petri

Un peu d'histoire...

Les réseaux de Petri sont un outil à la fois graphique et mathématique qui s'applique aux systèmes concurrents, une théorie toute jeune née de la thèse, intitulée : *communication avec les automates*, présentée par Carl Adam Petri en 1962 [55]. Ils ont notamment l'avantage de pouvoir représenter des situations dynamiques et d'interpréter le parallélisme de manière intuitive.

2.2.1 Présentation et définition d'un Réseau de Petri

Un réseau de Petri est un graphe orienté biparti ayant deux types de noeuds : des *places* représentées par des cercles et des *transitions* représentées par des traits. Les arcs du graphe ne peuvent relier que des places vers des transitions,

ou des transitions vers des places.

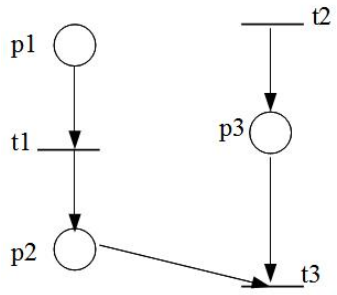


FIGURE 2.1 – Exemple d'un RdP

La Fig 2.1 représente un exemple de réseaux de Petri avec p_1, p_2, p_3 (resp. t_1, t_2, t_3) les places (resp. les transitions) du réseau. p_1 est la seule place d'entrée de la transition t_1 ; p_2 et p_3 sont les places d'entrée de la transition t_3 ; p_2 est aussi place de sortie de la transition t_1 ; p_3 est une place de sortie de la transition t_2. La transition t_2 est dite source car n'ayant pas de places en entrée et la transition t_3 est dite une transition puits car n'ayant pas de places de sortie.

Un réseau de Petri décrit un système dynamique à événements discrets. Les places permettent la description des états (aspects discrets), et les transitions permettent la description des événements (changements d'état ou évolution). Pour décrire la dynamique du système représenté, un troisième élément est ajouté : *jeton*. Une répartition des jetons dans les places à un instant donné est appelée : *marquage* du réseau de Petri.

Le marquage d'un *RdP* est précisé par la présence à l'intérieur des places d'un nombre fini (positif ou nul), de marques ou de jetons. Une place est donc vide ou marquée. Lorsque la place représente une condition logique (e.g. : machine au repos, ...), la présence d'un jeton indique que cette condition est vraie ; fausse dans le cas contraire. Lorsque la place représente une ressource (au sens le plus large, e.g. un stock, ...), elle peut contenir plusieurs jetons (e.g. le nombre de pièces en stock). Ainsi, le marquage initial d'un *RdP* correspond à la distribution initiale des jetons dans chacune des places du *RdP*, précisant l'état initial du système retenu pour l'analyse.

Définition 1 ***(Réseau de Petri marqué)***. *Un RdP marqué est le tuple* $R = (P, T, Pre, Post, M^0)$, *où :*

- P, *est un ensemble fini non vide de places ;*

- T, *est un ensemble fini non vide de transitions ;*
- $Pre : P \times T \longrightarrow \mathbb{N}$, *est l'application d'incidence avant (precondition), correspondant aux arcs directs reliant les places aux transitions ;*
- $Post : P \times T \longrightarrow \mathbb{N}$, *est l'application d'incidence arrière (post condition), correspondant aux arcs directs reliant les transitions aux places ;*
- $M^0 : P \longrightarrow \mathbb{N}$, *est une application, dite marquage initial, qui associe à chaque place p de P un nombre de jetons.*

Graphiquement, un arc relie une place p à une transition t (resp. une transition t à une place p), si $Pre(p,t) \neq 0$ (resp. $Post(p,t) \neq 0$). Cette partie statique est complétée par le marquage initial noté M^0. La partie dynamique du réseau consiste alors à faire évoluer ce marquage, c'est ce que nous allons voir dans le paragraphe qui suit.

2.2.2 Règles de fonctionnement

Une transition est dite *sensibilisée* pour un marquage donné, si chacune de ses places d'entrée contient un nombre de jetons satisfaisant la précondition de l'arc reliant la place à la transition. Quand une transition est sensibilisée, elle peut être franchie. Le *franchissement* d'une transition se fait en enlevant le nombre de jetons (dénoté par la précondition), à chacune des places d'entrée de la transition et en ajoutant un nombre de jetons à chacune des places de sortie (dénoté par la postcondition), de cette même transition. Plus formellement :

- Une transition t est dite sensibilisée par le marquage M, si : $\forall p \in P$, $M(p) \geq Pre(p,t)$. Par la suite, nous notons par $Te(M)$ l'ensemble des transitions sensibilisées pour M.
- Le tir (le franchissement) d'une transition t a pour conséquences :

- De retirer $Pre(p,t)$ jetons de chaque places d'entrée p de la transition t.
- De rajouter $Post(p,t)$ jetons à chaque place de sortie p de la même transition t.

- Si le marquage avant le tir de la transition t est M son tir conduira donc au marquage M' vérifiant : $\forall p \in P$, $M'(p) = M(p) - Pre(p,t) + Post(p,t)$.

L'ensemble des marquages accessibles du réseau R en partant du marquage M^0 en franchissant une séquence de transitions S sera noté $Acc(R)$. Si $M^n \in Acc(R)$, alors il existe au moins une séquence de transitions $S = (t_1, ..t_n)$ franchissable qui permet de passer du marquage M^0 au marquage M^n.

Lorsqu'une transition est validée, cela n'implique pas qu'elle sera immédiatement franchie ; cela ne représente qu'une possibilité de franchissement ou d'évolution

du *RdP*. Pour les *RdP* il y a un seul franchissement à la fois. Ces remarques impliquent que lorsque plusieurs transitions sont sensibilisées dans un même marquage (les transitions sont franchissables en parallèle), l'ensemble des marquages suivants sera obtenu en considérant toutes les possibilités de franchissement des transitions les unes après les autres. Ainsi, n transitions sensibilisées conduiront à n marquages permettant ainsi d'étudier l'ensemble des comportements possibles. Cependant, ce mode de fonctionnement conduit à une explosion combinatoire des marquages pour les *RdP* complexes.

2.2.3 Transitions conflictuelles, parallèles et nouvellement sensibilisées

A un instant donné, plusieurs transitions peuvent être sensibilisées. Plusieurs cas peuvent se produire :

- *Conflit* : Les transitions ont au moins une place d'entrée commune (conflit structurel), et sont sensibilisées par le marquage (conflit effectif). Après le tir d'une des transitions, les autres, en général, ne sont plus sensibilisées (à moins qu'elles redeviennent sensibilisées par le nouveau marquage). Formellement, deux transitions t_1 et t_2 sensibilisées pour M sont dites en conflit pour le marquage M, si la condition suivante est vérifiée : $\exists p \in P, \quad Pre(p,t_1) + Pre(p,t_2) > M(P)$. Un exemple de conflit est représenté en Fig. 2.2.a.
- *Parallélisme* : Les transitions n'ont pas de place d'entrée commune (parallélisme structurel), et sont sensibilisées par le marquage (parallélisme effectif). Après le tir de la transition t_2, la transition t_1 demeure sensibilisée (Fig. 2.2.b).
- Soit le tir de la transition t_f à partir du marquage M pour aboutir au marquage M'. Une transition t' sensibilisée pour M' est dite *nouvellement sensibilisée* pour M' si elle est sensibilisée seulement dans M', ou bien elle est sensibilisée pour M et M' et en conflit avec la transition franchie t_f pour le marquage M. Nous notons $New(M')$ l'ensemble des transitions nouvellement sensibilisées pour M'. Formellement, $t' \in New(M')$ ssi,

$$\begin{cases} (t\prime \in Te(M')) \wedge (t' \notin Te(M)) & \vee \\ (t' \in Te(M) \cap Te(M')) \wedge (\exists p \in P, \quad Pre(p,t_f) + Pre(p,t') > M(p)) \end{cases}$$

Autrement, la transition t' est dite *anciennement sensibilisée* ou *persistante*.

Dans le cas du parallélisme, si le modèle du *RdP* est implementé dans une machine parallèle, toutes les transitions parallèles peuvent être tirées au même instant (dit *vrai* parallélisme, définissant un ordre partiel). Si la machine est séquentielle, nous implémentons le parallélisme par *entrelacement* : une seule transition est tirée à la fois. Dans ce travail nous utiliserons le parallélisme par

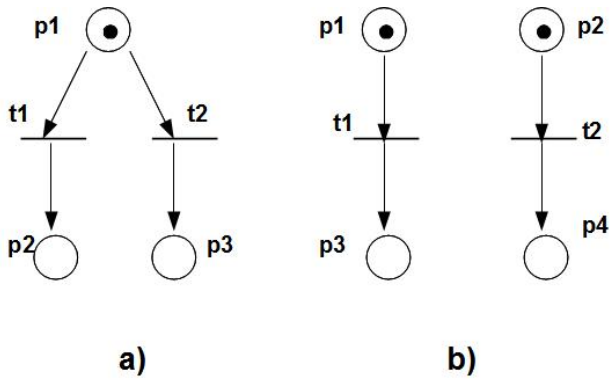


FIGURE 2.2 – transitions sensibilisées : a) conflit, b) parallélisme

entrelacement : il faut donc toujours choisir une transition à franchir parmi les transitions sensibilisées dans un marquage m donné.

Le choix de la transition à tirer (conflit ou parallélisme), est fait en respectant une priorité (statique ou dynamique) pré-établie ou au hasard. Dans le cas d'un *conflit*, ce choix correspond vraiment au choix d'une transition (ce tir empêche tous les autres), mais dans le cas du *parallélisme*, ce choix correspond seulement à un choix dans *l'ordre* du tir des transitions.

2.2.4 Propriétés des *RdP*

Les principales propriétés des *RdP* peuvent être classées en deux groupes :

- Les propriétés structurelles.
- Les propriétés dynamiques.

Les propriétés structurelles sont indépendantes du marquage initial et sont liées à la topologie du réseau. Leur analyse repose essentiellement sur les techniques d'algèbre linéaire et leur but est de bâtir une passerelle entre la structure du réseau étudié et son comportement.

Les propriétés dynamiques, quant à elles dépendent du marquage initial et sont liées à l'évolution de ce dernier. Leur vérification nécessite bien souvent la construction du graphe de marquages accessibles. Elles permettent d'apporter des réponses aux questions concernant l'accessibilité d'un marquage particulier, la bornitude, la vivacité, ...

2.2.4.1 Propriété de bornitude :

Cette propriété permet de caractériser la possibilité pour une place d'accumuler une quantité bornée ou non de marques au cours de l'évolution du réseau.

- Une place p est bornée ou *k-bornée* pour un marquage initial M_0, s'il existe un entier naturel k tel pour tout marquage accessible à partir de M_0, le nombre de marques de p reste inférieur ou égal à k, i.e. : $\exists k \in \mathbb{N} \ / \quad \forall\, M \in Acc(R)$, $M(p) \leq k$.
- Un RdP est dit *Sauf*, s'il est *1-borné*.

2.2.4.2 Réseau vivant :

Un RdP marqué est vivant si et seulement si pour tout marquage accessible M ($M \in Acc(R)$), et pour toute transition t, il existe une séquence de franchissement partant de M et contenant t. La vivacité d'un réseau garantit le franchissement de toute transition quelque soit le marquage atteint. La propriété de vivacité est une propriété forte, souvent difficile à vérifier.

2.2.4.3 Réseau bloqué :

Un RdP est bloqué si aucune transition n'est validée pour un marquage donné M ; M est appelé *marquage puits*.

2.2.4.4 Réseau ré-initialisable :

Un RdP est réinitialisable pour un marquage initial M^0, si pour tout marquage accessible M, il existe une séquence de franchissement de transitions telle que l'on puisse accéder à M^0 (M^0 est alors appelé *état d'accueil*).

2.2.5 Graphe de couverture

La vérification des propriétés données ci-dessus se fait généralement en établissant le *graphe de couverture* qui est composé de nœuds qui correspondent aux marquages accessibles, et d'arcs correspondant aux franchissements de la transition qui fait passer d'un marquage à un autre. Le nombre de nœuds dans ce graphe est fini.

La construction du graphe de couverture permet de décider si un RdP est borné : on dit que la propriété de réseau borné est décidable. Dans ce cas, le graphe sera appelé le graphe des marquages accessibles et noté $GAcc(R, M^0)$. A partir de ce graphe, on peut vérifier toutes les autres propriétés (vivacité, accessibilité d'un marquage, ...). Cependant, pour un RdP non borné, établir l'arbre de couverture

ne suffit pas pour résoudre le problème d'accessibilité et le problème de vivacité. Ces deux problèmes sont également décidables, mais par des algorithmes plus compliqués. Malgré ceci, le graphe de couverture reste le seul outil pour vérifier les propriétés dynamiques d'un *RdP* pour un marquage initial donné. Cette approche d'analyse est générale, elle est applicable à toute classe de réseaux. Mais, ses limites sont liées à l'explosion combinatoire du nombre d'états. De plus, à partir de ce graphe on peut savoir s'il existe des fonctionnements spécifiques comme par exemple l'existence d'un fonctionnement répétitif stationnaire. L'approche qui est basée sur le graphe de couverture pour étudier les *RdP* constitue l'analyse énumérative.

Les graphes de marquages des réseaux de petri de la Fig 2.2.a) et la Fig 2.2.b) sont représentés respectivement sur la Fig 2.3.a) et Fig 2.3.b).

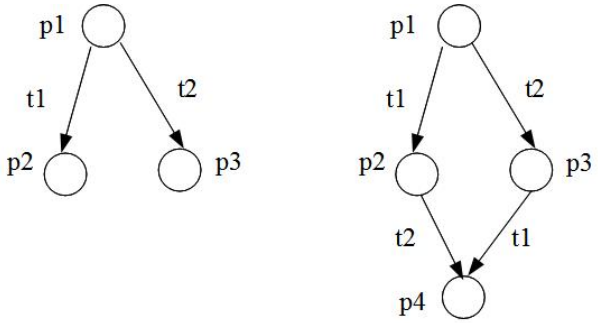


FIGURE 2.3 – graphe de marquage : a) conflit, b) parallélisme

2.3 Les réseaux de Petri Temporels

Plusieurs extensions ont été apportées aux réseaux de Petri pour pouvoir représenter implicitement le temps. En associant le temps aux places et/ou transitions et/ou arcs. Nous considérons ici le cas où le temps est associé à chaque transition [52] sous la forme d'un intervalle de temps compris entre une valeur *tmin* notée :$x(t)$ et une valeur *tmax* qui représentent respectivement la borne minimum et la borne maximum du domaine du tir où la transition sensibilisée peut être tirée ; le modèle est appelé réseau de petri temporel (*RdPT*) [52].

2.3.1 Présentation et définition des RdPT

Les réseaux de Petri T-temporels, sont une extension des réseaux de Petri introduite par Merlin en 1974 [52]. Ils consistent à ajouter du temps au modèle

classique sous la forme d'un intervalle (statique) : $I(t) = [tmin(t), tmax(t)]$ associé à chaque transition t du réseau. Pour être tirée, une transition t doit non seulement être sensibilisée mais également l'avoir été continûment pendant une durée comprise entre $tmin(t)$ et $tmax(t)$. Dans la suite de ce manuscrit, ces réseaux sont appelés simplement réseaux de Petri temporels ; nous les désignons par l'acronyme $RdPT$.

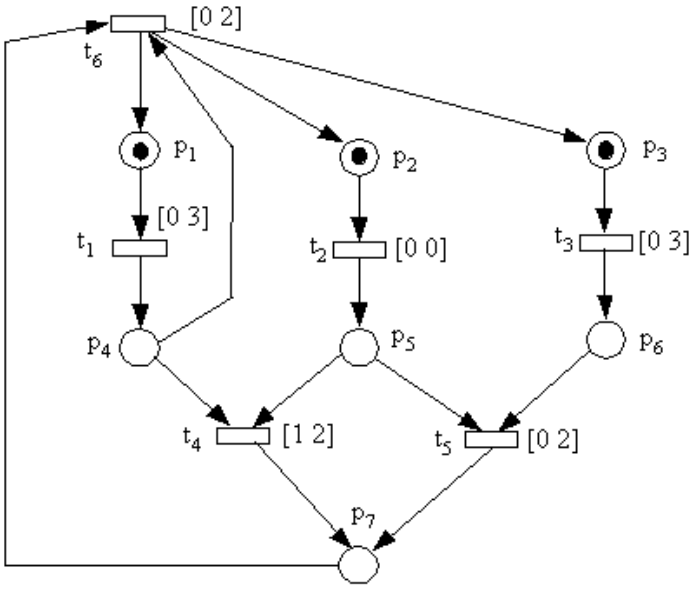


FIGURE 2.4 – Un réseau de Petri T-temporel

Considérons le réseau de Petri temporel de la Fig 2.4 présenté dans [4]. Les transitions t_1, t_2 et t_3 sont sensibilisées au sens classique des réseaux de Petri.

Au départ, les horloges associées respectivement à t_1, t_2 et t_3 ont la même valuation $\underline{t_1} = \underline{t_2} = \underline{t_3} = 0$. Lors du franchissement de t_2, le jeton de la place p_2 est consommé et un jeton est généré dans la place p_5. Les transitions t_4 et t_5 ne sont pas encore sensibilisées car les places p_4 et p_6 ne contiennent toujours pas de jetons.

Nous allons maintenant définir plus formellement les réseaux de Petri T-temporels.

Définition 2 *Un TPN (Time Petri Net) ou Réseau de Petri T-temporel [52] est défini par le couple (R, I), où :*

– *R est un réseau de Petri marqué ;*

– *I est l'application de délai* $I : T \longrightarrow \mathbb{Q}^+ \times \mathbb{Q}^+ \cup \{\infty\}$, *où* $\mathbb{Q}^+$ *est l'ensemble des rationnels non négatifs, écrite* $I(t) = [tmin(t), tmax(t)]$ *avec* $0 \leq tmin(t) \leq tmax(t)$

Soit $RT := (R, I)$ un *RdPT La smantique forte* impose dans un réseau de Petri temporel, lorsque plusieurs transitions sont franchissables, de franchir l'une de ces transitions avant la fin du domaine de tir des autres transitions. Pour plus de détail sur le fonctionnement du modèle, la sémantique formelle d'un *RdPT* est représentée dans la section suivante conjointement avec celle d'un $ITPN$.

Pour un *RdPT*, l'accessibilité de marquages est un probléme indécidable [44], ce qui implique que la bornitude, l'accessibilité d'états et la vivacité sont également des problémes indécidables. En revanche, la k-bornitude est décidable et peut être décidée par le calcul d'une abstraction finie de l'espace d'états telle que le graphe des classes d'états de [9, 10] (voir chapitre 3).

2.4 Les réseaux de Petri étendus à Chronométres

Les *RdPT* [52] sont vus comme un modèle large et très répandu pour l'analyse et la spécification formelle alliant les avantages d'une description graphique puissante et d'une sémantique formelle. Cependant, un tel modèle ne permet pas de spécifier les mécanismes de préemption temporelle : mécanisme où une action peut être stoppée pour être reprise plus tard. Pour ce faire, un certain nombre d'auteurs ont proposé d'étendre les *RdPT* afin de prendre en compte ces aspects : Les *Scheduling-TPN* [47, 49], les *Preemptive-TPN* [19], *Inhibitor-TPN* [60] et les *Stopwatch-TPN* [12]. Les deux premiers ajoutent des ressources et des priorités aux *RdPT*, alors que les $ITPN$ introduisent des arcs inhibiteurs qui contrôlent la progression des horloges des transitions. Dans cette section on s'interesse au cas des $ITPN$ [60] (*Inhibitor Arc Time Petri Nets*).

2.4.1 Présentation et Définition d'un ITPN

Considérons le modèle *ITPN* décrit dans la figure Fig. 2.5 présenté dans [60]. Il y'a un arc inhibiteur (représenté par un arc terminé par un petit disque), connectant la place p_7 à la transition t_3. Initialement, la place p_3 est marquée et p_7 ne l'est pas. Par conséquent, t_3 est sensibilisée mais non inhibée, et son chronomètre est activé. Il en est de même pour la transition t_4 sensibilisée pour le marquage initial. Cependant, le tir de la transition t_4 va consommer le jeton de la place p_4 et produire un jeton dans p_2 et un autre dans p_7. Il en résulte l'activation

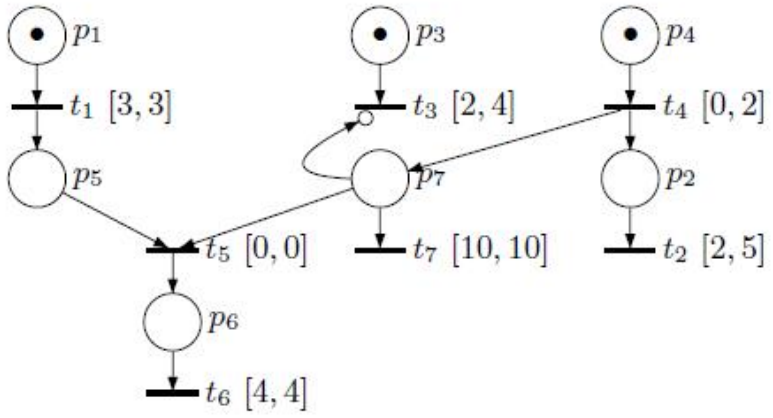


FIGURE 2.5 – Un exemple d' $ITPN$

de l'arc inhibiteur et la suspension du chronomètre de la transition t_3 (t_3 est ainsi inhibée), qui persistera aussi longtemps que la place p_7 restera marquée.

Nous allons maintenant définir plus formellement les réseaux de Petri à arcs inhibiteurs ($ITPN$)

Définition 3 *Un ITPN (Inhibitor arc Time Petri Net) ou réseau de Petri à arcs inhibiteurs [60] est défini par le couple* (RT, IH) *où :*

- RT *est un réseau de Petri temporel* ;
- $IH : P \times T \longrightarrow \mathbb{N}$ *est la fonction arc inhibiteur ; il y'a un arc inhibiteur connectant la place* p *à la transition* t, *si* $IH(p,t) \neq 0$.

En plus des notions définies plus haut pour la sémantique des réseaux de Petri temporels classiques, pour un $ITPN$ nous avons :

- Une transition t est dite *inhibée* pour le marquage M, si elle est *sensibilisée* et qu'il existe une place connectée à t par un arc inhibiteur dont le marquage satisfait la valuation de cet arc $(t \in Te(M)) \wedge \exists p \in P, 0 < IH(p,t) \leq M(p)$. Nous notons $Ti(M)$ l'ensemble des transitions *inhibées* pour M.
- Une transition t est dites *activée* pour le marquage M, si elle est sensibilisée et non inhibée, $(t \in Te(M)) \wedge \ (t \notin Ti(M))$; nous notons par $Ta(M)$ l'ensemble des transitions *activées* pour le marquage M.

Nous définissons la sémantique d'un $ITPN$ par un système de transitions étiquetées [61].

Définition 4 *La sémantique d'un ITPN est définie sous la forme d'un système de transitions* $ST = (\Gamma, e^0, \rightarrow)$, *tel que :*

- Γ *est l'ensemble des configurations, chaque configuration, notée* e, *apartenant à* Γ *un couple* (M, V) *où* M *est un marquage et* V *est une fonction de valuation qui associe à chaque transition sensibilisée* t *de* $Te(M)$ *la valeur du chronométre associé à* t. *Formellement on a :* $\forall t \in Te(M), \quad V(t) := [x(t), y(t)]$
- $e^0 = (M^0, V^0)$ *est l'état initial, tel que :* $\forall t \in Te(M^0), \quad V^0(t) := I(t) := [tmin(t), tmax(t)]$.
- $\rightarrow \in \Gamma \times (T \times \mathbb{Q}^+) \times \Gamma$ *est la relation de transition, tel que :* $((M, V), (t_f, \underline{t_f}), (M^{\uparrow}, V^{\uparrow})) \in \rightarrow$, *ssi :*

  (i) $t_f \in Ta(M)$.

  (ii) $x(t_f) \leq \underline{t_f} \leq \underset{\forall t \in Ta(M)}{MIN} \{y(t)\}$.

  et on a :
  $\forall p \in P,\ M^{\uparrow}(p) := M(p) - Pre(p, t_f) + Post(p, t_f)$.
  $\forall t \in Te(M^{\uparrow})$
  si $t \notin New(M^{\uparrow})$:

$$\begin{cases} [x^{\uparrow}(t),\ y^{\uparrow}(t)] := [MAX(0,\ x(t) - \underline{t_f}),\ \ y(t) - \underline{t_f}] & t \in Ta(M) \\ [x^{\uparrow}(t),\ y^{\uparrow}(t)] := [x(t),\ \ y(t)] & t \in Ti(M) \end{cases}$$

  si $t \in New(M^{\uparrow})$
  $[x^{\uparrow}(t),\ y^{\uparrow}(t)] := I(t) = [tmin(t),\ tmax(t)]$

Seuls les chronométres associés aux transitions activées progressent, alors que ceux relatives aux transitions inhibées sont suspendues. Par conséquent, une transition t_f peut être tirée à l'instant relatif $\underline{t_f}$, à partir de l'état accessible e, si *(i)* t_f est activée pour le marquage M, et si *(ii)* le temps peut progresser durant l'intervalle de tir de t_f en satisfaisant les contraintes temporelles des autres transitions activées. Aprés le tir de t_f l'état accessible, noté $e^{\uparrow}$, est obtenu :

- en consommant un nombre de jetons présents dans chaque place p en entrée de t_f (donné par la valeur $Pre(p, t_f)$), et en produisant un certain nombre de jetons dans chaque place p en entrée de t_f (donné par la valeur $Post(p, t_f)$) ;
- en initialisant la valeur du chronométre de chaque transition nouvellement sensibilisée à zéro. Cependant, la valeur de chaque chronométre relatif à un transition persistante reste inchangée.

Remarque 1 *En l'absence d'arcs inhibiteurs dans le réseau, la sémantique définie précédemment est celle d'un RdPT.*

2.5 Exemples et classification des réseaux de Petri à Chronométres

Propriété 1 *Le modèle $ITPN$ étend les RdPT en contrôlant la progression des transitions. Le franchissement d'une transition peut être interrompu, s'il y a une place non vide connectée à cette transition par un arc inhibiteur (voir Fig. 2.6).*

La Fig 2.6-b- présente le modèle $ITPN$ modélisant la tâche dans la Fig 2.6-a-. Dans cette figure, les transitions t_2 et t_4 seront sensibilisées par le franchissement de t_1. Le chronomètre associé à t_2 compte la durée effective de l'exécution. Ce chronomètre devient inactif lorsque p_4 est marquée. La sensibilisation de t_4 signifie que l'interruption peut arriver à n'importe quel instant pendant l'exécution. Lorsqu'une interruption se produit par le franchissement de t_4 alors t_6 sera validée et son chronomètre décomptera la durée de l'interruption. L'interruption et la reprise peuvent se produire plusieurs fois pendant l'exécution. Ceci est présenté par le franchissement successivement de t_5 et t_6.

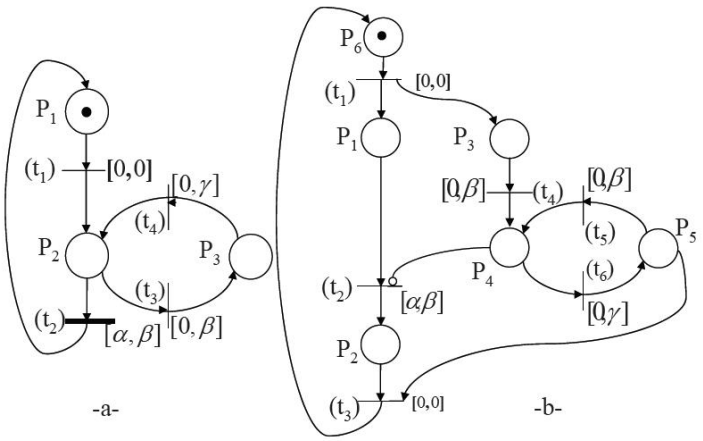


FIGURE 2.6 – a) $SwTPN$ d'une tâche interruptible b) Modélisation de la même tâche interruptible par $ITPN$

Propriété 2 ***(Relation entre** $SwPN$ **et Sheduling-TPN ou Preemptive-TPN)*** *Les Scheduling-TPN [47] étendent les RdP en associant aux places deux nouveaux attributs, les ressources et les priorités. Les preemptive-TPN [19] associent les mêmes attributs aux transitions au lieu des places.*

Les Scheduling-TPN et Preemptive-TPN sont particulièrement bien adaptés à la modélisation des systèmes préemptifs pour des objectifs d'ordonnancement. La façon d'ordonnancer des tâches réparties sur différents processeurs est prise en compte pour une priorité fixe, tandis que le problème de la suspension et la reprise d'une tâche est plus général. A titre d'exemple, ces modèles ne sont pas capables de modéliser une relation circulaire de la priorité pouvant arrêter définitivement l'écoulement de temps des transitions, tandis que les ressources nécessaires associées aux transitions ou aux places sont disponibles. Donc, un Scheduling-TPN n'est pas capable de modéliser cette relation de priorité tandis que le modèle SwPN le peut.

Propriété 3 *(**Relation entre** $ITPN$ **et Sheduling-TPN ou Preemptive-TPN**) On peut facilement conclure que les Sheduling-TPN et Preemptive-TPN sont une stricte sous-classe des $ITPN$.*

On peut aussi conclure qu'un Scheduling-TPN ou un Preemptive-TPN peut être modélisé par un $SwPN$ mais le contraire n'est pas vrai ; nous n'irons pas plus loin en ce qui concerne ces derniers car on s'interesse exclusivement aux $ITPN$.

2.6 Conclusion

Dans ce chapitre, nous avons présenté des modèles utilisés pour la modélisation et la vérification formelles des systèmes temps réel et temps réel préemptifs : les réseaux de Petri temporels et leurs extentions à chronométres. Nous nous sommes surtout intéressés aux réseaux de Petri à arcs inhibiteurs($ITPN$). Après un rappel sur la syntaxe et la sémantique de ce dernier, nous avons présenté une brève classification entre les differents modèles à chronométres et en exposant certaines relations intéressantes. La question est à présent : comment déterminer l'éspace d'état d'un réseau de Petri temporel et d'un réseau de Petri à arc inhibiteur pour les systèmes temps réel respectivement temps réel préemptifs. Ceci fait l'objet des deux chapitres qui suivent.

Chapitre 3

Le calcul de l'espace d'états d'un *RdPT*

3.1 Introduction

Pour les modèles formels temporisés tels que les réseaux de Petri T-temporels, le temps est représenté de façon dense ce qui implique qu'en général, on se trouve face à un espace d'états infini dont on peut trouver, sous certaines conditions, des abstractions finies. Dans ce chapitre, nous exposons un panorama des méthodes d'analyse disponibles pour les (RdPT). Ces méthodes, sont basées essentiellement sur la technique des «classes d'états» initiées par [9] mais d'autres aussi, basées sur des graphes des «régions et des zones» [67] [37] respectivement. Nous présentons aussi une méthode alternative basée sur le principe de classes d'états atomiques avec l'aspect quantitatif du modèle : «le graphe modeC» [21, 20]. Des méthodes récemment introduites sont présentées visant à traduire un RdPT vers un AT [24, 48]. Enfin, nous présentons une étude comparative entre les différents graphes générés afin d'expliquer les raffinements apportés par chaque méthode. L'exemple discuté dans ce chapitre a été traité avec les outils de l'environnement Tina [11] et graphC [21, 20] [4] et l'outil Roméo [36].

3.2 L'Espace d'états : Un tour d'Horizon

Dans la littérature, les abstractions de l'espace d'états des *réseaux de Petri temporels* sont principalement basées sur la notion de *classe d'états*, alors que celles des *automates temporisés* sont basées sur les *régions et les zones*.

Dans le cas des *RdPT*, plusieurs abstractions de l'espace d'états ont vu le jour, d'un point de vue chronologique, on peut citer : le graphe des classes d'états (State Class Graph : SCG) [9, 10], le graphe des régions géométriques (Geometric Region Graph : GRG) [67], le graphe des classes d'états linéaires fortes (Strong State Class Graph : SSCG) [8], le graphe des zones (Zone Based Graph : ZBG) [37, 38, 18], le graphe des classes atomiques (Atomic State Class Graphs : ASCGs) [67, 8, 14, 15] et le graphe des classes en modeC [21, 20]. Par ailleurs, une extension de la construction du graphe des classes d'états a été proposée dans le but de construire, pour un *RdPT* borné, son automate temporisé équivalent avec un nombre d'horloges minimal [37, 38].

Ces abstractions différent sur les classes de propriétés qu'elles préservent. Les graphes des classes fournissent une représentation finie de l'espace d'états d'un *RdPT* borné préservant les propriétés LTL [9, 10] ou CTL^* [67, 8, 14]. Le graphe des régions géométriques préserve les propriétés CTL et le graphe des zones préserve l'accessibilité des marquages ou les propriétés LTL en fonction du critère de convergence utilisé. Pour la vérification de propriétés exprimées avec la logique temporelle $TCTL$, les auteurs [39, 18, 40] proposent une méthode à la volée pour la vérification d'une sous-classe de propriétés $TCTL$ bien définie. D'autres travaux se basent sur la traduction du *RdPT* vers un AT vérifiant ces dernières propriétés [24, 48].

Nous présentons, dans ce chapitre les principales méthodes appliquées sur les *RdPT* (y compris celles proposées initialement pour les automates).

3.3 Les méthodes basées sur le graphe des classes d'états

3.3.1 Etats et classe d'états :

L'ensemble des états d'un réseau temporel peut être infini pour deux raisons :

1. Parce qu'un état peut admettre une infinité de successeurs,
2. Parce qu'un réseau peut admettre des échéanciers de longueurs infinies passant par des états dont les marquages sont tous différents.

Pour gérer le premier cas, on regroupera certains ensembles d'états en *classes d'états*. Plusieurs regroupements sont possibles (voir la suite de cette section).

Une classe d'états regroupe donc tous les états atteints par une même séquence de tir réalisable. En particulier, tous les états d'une même classe ont alors le même marquage.

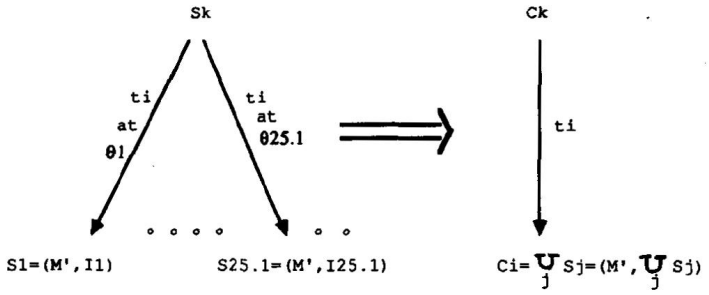


FIGURE 3.1 – Etat et classe d'états [10]

3.3.2 Graphe de classes préservant marquages et propriétés LTL

Le graphe des classes d'états de [9, 10], ou (Stat class Graph : SCG), est l'ensemble des ensembles d'états accessibles après le tir d'une même séquence non temporisée. Par conséquent, tous les états d'une même classe ont le même marquage M ; une classe est définie par le couple (M, D) où M est le marquage commun à tous ses états ; et D est un système d'inéquations. Chaque variable présente dans D, notée $\underline{t_i}$, est associée à une transition sensibilisée t_i de même nom. La classe initiale est la classe d'équivalence de l'ensemble des états constitué du seul état initial. Plus formellement, une classe est définie comme suit :

Définition 5 *Soit $ST = (\Gamma, e^0, \rightarrow)$ le système de transition associé à un RdPT. Une classe d'état notée E, est l'ensemble des états de Γ accessibles après le tir d'une même séquence non temporisée $\sigma = (t_f^1, .., t_f^n)$ à partir de l'état initial e^0. La classe E est définie par (M, D), où M est le marquage accessible après le tir de σ, et D est le domaine de tir représenté par un système d'inéquations. Nous avons :*

$$D := \begin{cases} \bigwedge_{i \neq j} & (\underline{t_j} - \underline{t_i} \ \leq \ d_{ij}) \\ \bigwedge_{i \leq s} & (d_{i\bullet} \leq \ \underline{t_i} \ \leq d_{\bullet i}) \end{cases}$$

$$avec (t_j, t_i) \in Te(M)^2 \quad d_{ij} \in \mathbb{Q} \cup \{\infty\}, \ d_{\bullet i} \in \mathbb{Q}^+ \cup \{\infty\}, d_{i\bullet} \in \mathbb{Q}^+$$

Nous notons par l'élément $\{\bullet\}$ l'instant où la classe E est atteinte. Les coefficients, $d_{\bullet i}, d_{i\bullet}$ et d_{ij} sont respectivement, le délai résiduel maximum pour le tir de

la transition t_i, le délai résiduel minimum pour le tir de la transition t_i, et l'écart maximal entre les temps de franchissement de t_i vers t_j.

Dans le cas d'un $RdPT$: Les contraintes du système D ont une forme particulière, appelée DBM (Difference Bound Matrix) [34]. Cette forme permet d'appliquer un algorithme efficace pour le calcul d'une classe, dont la complexité est estimée à $O(m^3)$, où m est le nombre de transitions sensibilisées (nous allons voir dans le chapitre suivant que cette propriété n'est pas toujours vérifiée pour les $ITPN$).

3.3.2.1 Construction du graphe

Le graphe de classes d'un $RdPT$ est calculé par énumération des classes accessibles à partir de la classe initiale E^0 jusqu'à qu'il n y ait plus de classe à explorer. Formellement, le graphe de classes d'un $RdPT$ est défini comme suit :

Définition 6 *Le graphe de classes d'un RdPT [9, 10], noté GR, est un triplet $(CE, E^0, \longmapsto)$ où :*
- CE est l'ensemble des classes accessibles dans GR;
- $E^0 = (M^0, D^0)$ est une classe de CE appelée classe initiale telle que :
$D^0 = \{ \ \forall t_i \in Te(M^0), \quad tmin(t_i) \leq \underline{t_i} \leq tmax(t_i) \ \}$
- $\longmapsto$ est la relation de transition entre classes définie sur $CE \times T \times CE$, telle que
$((M, D), t_f, (M^{\uparrow}, D^{\uparrow})) \in \longmapsto$, *si et seulement si :*

a) *Le système D augmenté par les contraintes de tir de t_f que nous écrivons : $(\forall t \in Te(M), \quad \underline{t_f} \leq \underline{t})$ admet au moins une solution.*

b) $\forall p \in P$, $M^{\uparrow}(p) := M(p) - Pre(p, t_f) + Post(p, t_f)$.

c) *Le système $D^{\uparrow}$ est calculé à partir de D, en appliquant l'algorithme suivant :*

1. *Remplacer chaque variable $\underline{t}$ (relative à une transition sensibilisée par M), par : $\underline{t} := \underline{t_f} + \underline{t'}$, exprimant ainsi la progression du temps.*
2. *Eliminer par substitution dans le système ainsi obtenu, la variable $\underline{t_f}$, ainsi que toutes les variables relatives aux transitions désensibilisées suite au tir de t_f, en utilisant par exemple la méthode de Fourier-Motzkin.*
3. *Rajouter au système ainsi obtenu, les contraintes des transitions nouvellement sensibilisées dans $M^{\uparrow}$: $\forall t_i \in New(M^{\uparrow}), \quad tmin(t_i) \leq \underline{t_i} \leq tmax(t_i)$*

Etant donné une classe $E = (M, D)$. Le calcul de la classe $E^{\uparrow} = (M^{\uparrow}, D^{\uparrow})$ accessible à partir de E en franchissant t_f consiste en le calcul du marquage accessible $M^{\uparrow}$ de manière usuelle, et du domaine de tir induit par le nouveau système $D^{\uparrow}$.

Une classe $E^{\uparrow}$ est atteignable depuis E, en franchissant la transition t_f si : *a*) cette derniére est sensibilisée par le marquage M et franchissable avant toutes les autres transitions sensibilisées ; *b*) correspond au calcul standard du nouveau marquage $M^{\uparrow}$; *c*) Le système $D^{\uparrow}$ capturant l'espace des états de $E^{\uparrow}$ est calculé à partir du système D augmenté des contraintes de tirs de t_f : (1) Le renommage et la substitution des variables permettent de décaler l'origine du temps vers l'instant où la nouvelle classe $E^{\uparrow}$ a été atteinte. (2) Un système équivalent est calculé où les variables des transitions désensibilisées suite au tir de t_f sont éliminées, (3) finalement, les contraintes des transitions nouvellement sensibilisées sont rajoutées.

Le système de transitions obtenu est potentiellement infini car il n'y a à priori aucune raison pour que l'on finisse toujours par atteindre des classes sans successeurs. Nous allons donc définir un critére de convergence sous la forme d'une relation d'équivalence entre les classes. Nous notons : $\rceil D \lceil$ l'ensemble de solutions de D. Nous définissons l'égalité de deux classes d'états comme suit [9] :

Définition 7 *Deux classes d'états $E = (M, D)$ et $E' = (M', D')$, données dans leur forme normales sont dites égales, , ssi :*

1. *elles correspondent au même marquage, $M = M'$;*
2. *l'ensemble des solutions est tel que : $\rceil D \lceil = \rceil D' \lceil$.*

La forme canonique (normalisée), du système d'inéquation D est unique, cette propriété est importante car elle permet de détecter les classes équivalentes en comparant leur forme normale. Le graphe ainsi obtenu permet de vérifier les propriétés de logique temporelle linéaire (*Linear Temporel Logic*) [56].

3.3.2.2 Illustration

En reprenant le réseau de Petri du chapitre.2 représenté sur la figure Fig 3.2a). Son graphe de classes préservant marquage et propriétées LTL est représenté sur la figure Fig 3.2b). Ce réseau nous servira comme modèle pour comparer différents graphes de classes [4]. Le graphe complet contient 13 classes et 21 transitions.

La classe initiale (état initial), est définie par $E_0 = (M_0, D_0)$, où, $M_0 = p_1, p_2, p_3 \mapsto 1$ et :

$$D_0 := \begin{cases} 0 \leq \underline{t_1} \leq 3 \\ 0 \leq \underline{t_2} \leq 0 \\ 0 \leq \underline{t_3} \leq 3 \end{cases}$$

Dans E_0, les transitions $t_1; t_2, t_3$ sont sensibilisées et franchissables. Selon la sémantique forte (Une transition sensibilisée doit être tirée avant la fin de tir des autres : le plus petit *tmax(t))*, les transitions $t_1; t_2; t_3$ peuvent être tirées dans un

intervalle $\in [0\ \ 0]$. Si on considère le tir de la transition t_2 de la séquence $\sigma = (t_2; t_3; t_1; t_4)$ à partir de E_0 à un instant $\in [0\ \ 0]$, on obtient la classe $E_2 = (M_2, D_2)$ définie comme suit :

a) Le système D_0 est augmenté par les contraintes de tir de t_2 que nous écrivons :
$\underline{t_2} \leq \underline{t_1}$
$\underline{t_2} \leq \underline{t_3}$

b) $M_2 = p_1, p_3, p_5 \mapsto 1$

c) Le système D_2 est calculé à partir de D_0, en appliquant l'algorithme suivant :

1. Les transitions t_1 et t_3 restent sensibilisées lors du tir de t_2, donc pour exprimer la progression du temps il faut remplacer :
$\underline{t_1} = \underline{t_2} + \underline{t_1}'$
$\underline{t_3} = \underline{t_2} + \underline{t_3}'$

2. Eliminer du système obtenu, les variables désensibilisées aprés le tir de t_2 (c-à-d : $\underline{t_1}; \underline{t_2}; \underline{t_3}$), ce qui conduit à faire les calculs suivants :

   $0 \leq \underline{t_2} + \underline{t'_1} \leq 3$
   $0 \leq \underline{t_2} + \underline{t'_3} \leq 3$

   $0 - \underline{t'_1} \leq \underline{t_2} \leq 3 - \underline{t'_1}$
   $0 - \underline{t'_3} \leq \underline{t_2} \leq 3 - \underline{t'_3}$
   Avec : $0 \leq \underline{t_2} \leq 0$

   On obtient :
   $0 - \underline{t'_1} \leq 3 - \underline{t'_3}$
   $0 - \underline{t'_3} \leq 3 - \underline{t'_1}$

   $0 - \underline{t'_1} \leq 0$
   $0 \leq 3 - \underline{t'_1}$

   $0 - \underline{t'_3} \leq 0$
   $0 \leq 3 - \underline{t'_3}$

   On obtient :
   $0 \leq \underline{t'_1} \leq 3$
   $0 \leq \underline{t'_3} \leq 3$

3. Aucune transition nouvellement sensibilisée donc, aucune contrainte à rajouter.

Pour simplification, on remplace $\underline{t'_1}$ et $\underline{t'_3}$ par $\underline{t_1}$ et $\underline{t_3}$ respectivement pour la suite du calcul ; on a :

$$D_2 := \begin{cases} 0 \leq \underline{t_1} \leq 3 \\ 0 \leq \underline{t_3} \leq 3 \end{cases}$$

Le tir de t_3 à un instant $\in [0\ \ 3]$ à partir de E_2 conduit à la classe $E_6 = (M_6, D_6)$ obtenue de la même maniére :

a) Le système D_2 est augmenté par les contraintes de tir de t_3 que nous écrivons :
$\underline{t_3} \leq \underline{t_1}$

b) $M_6 = p_1, p_5, p_6 \mapsto 1$

c) Le système D_6 est calculé à partir de D_2, en appliquant l'algorithme suivant :

1. La transitions t_1 reste sensibilisée lors du tir de t_3, donc pour exprimer la progression du temps il faut remplacer :
   $\underline{t_1} = \underline{t_3} + \underline{t'_1}$

2. Eliminer du système obtenu, les variables désensibilisées aprés le tir de t_3, ce qui conduit à faire les calculs suivant :

   $0 \leq \underline{t_3} + \underline{t'_1} \leq 3$

   $0 - \underline{t'_1} \leq \underline{t_3} \leq 3 - \underline{t'_1}$
   Avec : $0 \leq \underline{t_3} \leq 3$

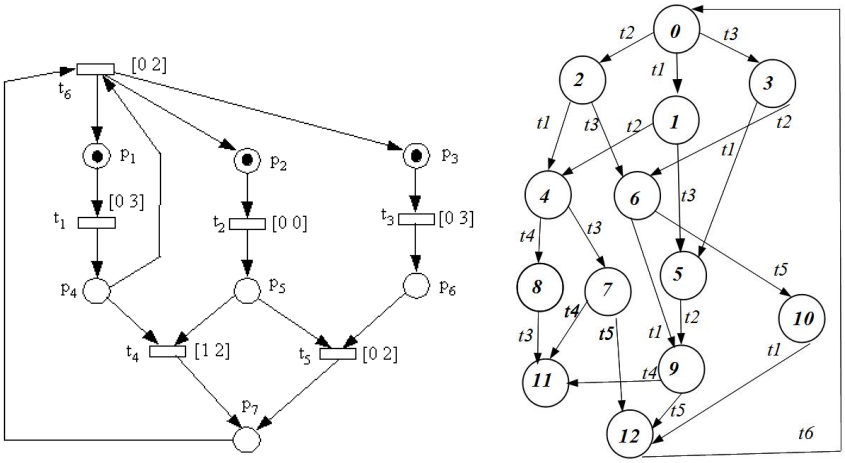
   On obtient :
   $0 \leq \underline{t'_1} \leq 3$

3. t_5 est une nouvelle transition sensibilisée, on ajoute au système obtenu la contrainte :
   $0 \leq \underline{t_5} \leq 2$

   On obtient :
   $$D_6 := \begin{cases} 0 \leq \underline{t_1} \leq 3 \\ 0 \leq \underline{t_5} \leq 2 \end{cases}$$

Les classes (marquages M et systèmes d'inéquation D), ne sont pas données explicitement dans cette section (voir ANNEXES).

La séquence de transitions $\sigma = (t_2; t_3; t_1; t_4)$ à partir de la classe initial E_0, représente un chemin de ce graphe, passant par les classes E_2, E_6, E_9 et arrivant à la classe E_{11}. On verra cependant que cette séquence n'est pas effectivement franchissable pour tous les états de la classe E_{11}.



a) Modèle du TPN b) Graphe GR [9]

FIGURE 3.2 – Le réseau de Petri et son Graphe de classes préservant marquage et LTL

Remarque 2 *Notons que dans le cas où plusieurs transitions sont concurrentes, des méthodes telle que l'élimination de Fourier-Motzkin ou Floyd-Warshall [28, 65], par exemple, est utile.*

3.3.2.3 Limites de la méthode

Lorsque la classe $E^{\uparrow}$ est le successeur de la classe E par la transition t_f, cela signifie qu'il existe au moins un état de E qui posséde un successeur par t_f dans $E^{\uparrow}$; mais cela n'implique pas forcément que tous les états de E ont un successeur par t_f dans $E^{\uparrow}$.

Considérons la figure 3.3 qui représente les intervalles pendant lesquels les transitions de la séquence précédente : $\sigma = (t_2; t_3; t_1; t_4)$, sont sensibilisées ainsi que l'instant de franchissement de chaque transition (représentée par une flèche).

Au départ, les transitions t_1, t_2, t_3 sont sensibilisées par le marquage initial M_0. Les intervalles de sensibilisation correspondent à leurs intervalles statiques respectifs ([0 0] pour t_2, [0 3] pour t_1 et t_3). Supposons que t_2 soit franchie à l'instant 0 et que t_3 le soit à l'instant 1. Le franchissement de t_3 sensibilise la transition t_5, qui peut rester sensibilisée pendant l'intervalle $[1\ 1] + [0\ 2] = [1\ 3]$.

Supposons que t_1 soit franchie à l'instant $2, 2$; elle sensibilise donc t_4, qui reste sensibilisée pendant l'intervalle $[2, 2\ \ 2, 2] + [1\ 2] = [3, 2\ \ 4, 2]$. Mais t_4 et t_5 ne sont pas en conflit effectif comme on peut le voir par les rectangles disjoints dans la figure 3.3. En fait, d'après la sémantique forte, c'est le temps qui commande et

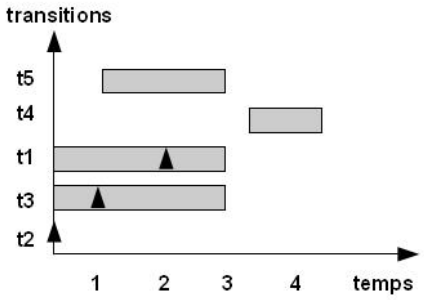


FIGURE 3.3 – Franchissement de séquence σ

pour ces dates de tir, t_5 doit être franchie avant t_4 et la séquence $\sigma = t_2; t_3; t_1; t_4$ n'est pas effectivement franchissable.

Remarque 3 *Le paragraphe précedent fait référence à des valeurs et intervalles flous [22], pour comprendre la façon de raisonner.*

Pour remédier à ce type de problématique, il est nécessaire de considérer des raffinement du graphe des classes d'états, comme celui proposé par *Berthomieu et al.* en graphe des classes d'états atomiques, présentée un peu plus loin, dans les sections suivantes.

La méthode de construction présenté dans cette section a l'avantage d'avoir été étendue au cas des réseaux de Petri à chronomètres [12, 13, 60], ce qui fait l'objet du chapitre.4

Enfin, cette construction permets de vérifier seulement les aspects *(qualitatifs)* du modèle et ne permet pas pour autant de vérifier des propriétés temporelles *quantitatives* de type $TCTL$ (Timed Computation Tree Logic) car elle ne fournit pas d'information sur les instants pour lesquels les tirs de transitions se sont produits. La sous-section suivante décrit cette extension du modèle.

3.3.2.4 Les propriétées $TCTL$

Pour la vérification de propriétés $TCTL$ sur les $RdPT$ bornés, les auteurs [39] définissent une sous-classe de cette logique notée : ($TCTL_{TPN}$), où les propositions atomiques dépendent des marquages.

Soit RT le modèle du réseau de Petri initial, on lui associe un nouveau $RdPT$ en parallèle, noté : *Alarm-clok* représenté par la figure. FIG 3.4. On obtient le modèle $RT \parallel$Alarm à partir duquel, on doit construire le graphe de classe tout en vérifiant la valeur de vérité d'une formule ($TCTL_{TPN}$), la construction s'arrête une fois la valeur de la formule est définie, cette construction permet de detecter le temps de réponse ; on peut connaitre la durée de parcours entre deux marquages

(sous forme d'un intervalle), comme ceci : une fois le premier marquage de la formule atteint pendant la construction du graphe, on met un jeton dans la place P_a pour sensibiliser t_a détectant le début de l'intevalle, plusieurs cas de figure peuvent se produire lors de la sensibilisation de t_b, le cas le plus simple détecte la fin de l'intervalle de la formule si t_b n'est pas tirée avant (avant d'atteindre le deuxiéme marquage), pour les autres cas, voir le détail de l'algorithme [39].

Dans ce cas, et pour plus de performances, les auteurs proposent une version plus relachée du graphe de classes notée, (RSCG : Relaxed Stat Class Graph), qui représente les classes accessibles par progression du temps, ce graphe est calculé de la même maniére sauf que pour chaque classe calculée, on met toutes les bornes inférieures du domaine de tir à zéro y compris pour la classe initiale.

Il est à noter que le choix est porté sur le graphe des classes car les auteurs s'intéressaient à l'aspect linéaire du modèle. Ces derniers (SCG et RSCG) préservent marquage et propriétés LTL, étants plus petits et rapides à calculer que les autres abstractions (voir la suite du chapitre). En plus, ils répondent à la propriété de finitude pour les $RdPT$ bornés.

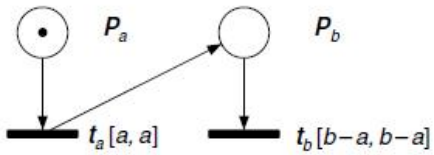


FIGURE 3.4 – Le $RdPT$ horloge d'alarme [39](Alarm-clok)

Une méthode alternative récemment introduite [40], décrit une construction du graphe des classes d'état avec inclusion, noté par (I-SCG : Inclusion contracted SCG). Ce dernier pourrait être obtenu en construisant d'abord le SCG puis regrouper les classes obtenues dans les classes englobantes par inclusion (tel que l'ensemble de solutions soit : $\rceil D' \lceil \subseteq \rceil D \lceil$), jusqu'à ce qu'aucune inclusion ne soit possible. Il est cependant plus intéressant d'effectuer cette opération lors de la construction elle-même. Par ce fait, la plupart des classes d'état ne sont pas calculées, ce qui fait gagner du temps et de l'espace ; Ce graphe préserve les propriétés d'accessibilité. Le graphe de classes par inclusion est fini ssi le $RdPT$ est borné. Il est à noter que l'égalité permet de préserver principalement le langage du modèle, alors que l'inclusion préserve les marquages accessibles. Par exemple, si l'on veut obtenir le graphe compact de [40], avec inclusion, on a : La classe E_9 incluse dans E_7. Ce qui donne un graphe de 12 classes et 21 transitions.

Enfin, rappelons aussi qu'un $RdPT$ est borné si le marquage de toute place admet une borne supérieure (voir Chapitre.2). L'ensemble des domaines de tir d'un réseau temporel étant fini [9], son graphe des classes est fini ssi le réseau est

borné. Cette propriété est indécidable, mais des conditions suffisantes peuvent être établies.

Dans [40], une condition nécessaire et suffisante basée sur (I-SCG) a été définie pour que le graphe soit un *ZENO*. Les auteurs ont aussi étendu l' approche [39], pour vérifier les propriétés ($TCTL_{TPN}$) chronométrée pour les *RdpT* pas forcement *ZENO* : le graphe de classes contient un cycle avec les bornes inférieures $tmin(t)$ de chaque transition de ce cycle est égal à zéro (a zero lower bound cycle).

3.3.3 Graphe de classes préservant états et propriétés LTL

Pour construire le graphe des classes d'états fortes ou (Strong Stat Class Grapg : SSCG) [8]il est nécessaire de représenter ces ensembles d'états S de façon canonique. Une représentation adéquate est fournie par les domaines d'horloges.

Les classes d'états fortes sont représentées par des paires $S = (M, Q)$, où M est un marquage et Q un domaine d'horloges décrit par un système d'inéquations : $G_{\underline{\gamma}} \leq \underline{g}$. Chaque variable présente dans Q, notée $\underline{\gamma_i}$, est associée à une transition sensibilisée t_i. La variable $\underline{\gamma_i}$ associe le temps écoulé depuis la dernière sensibilisation de cette transition.

3.3.3.1 Construction du graphe

Définition 8 *Le graphe de classes d'états fortes (Strong) d'un RdPT [8], noté GR_s, est un triplet $(CS, S^0, \longrightarrow)$ où :*
- CS est l'ensemble des classes accessibles dans GR_s;
- $S^0 = (M^0, Q^0)$ est une classe de CS appelée classe initiale telle que :
$Q^0 = \left\{ \ \forall t_i \in Te(M^0), \quad 0 \leq \underline{\gamma_i} \leq 0 \ \right\}$
- $\longrightarrow$ est la relation de transition entre classes définie sur $CS \times T \times CS$, telle que $((M, Q), t_f, (M^{\uparrow}, Q^{\uparrow})) \in \longrightarrow$, si et seulement si :

a) *Le système Q augmenté par les contraintes :*
$(0 \leq \theta) \wedge (tmin(t_i) \leq \theta + t_f \leq tmax(t_i), \forall t_i \in Te(M))$*, est consistent ($\theta$ est une nouvelle variable).*

b) $\forall p \in P,\ M^{\uparrow}(p) := M(p) - Pre(p, t_f) + Post(p, t_f)$.

c) *Le système $Q^{\uparrow}$ est calculé à partir de Q, en appliquant l'algorithme suivant :*

1. *Remplacer chaque variable $\underline{\gamma_i}$ (relative à une transition sensibilisée par M), par : $\underline{\gamma_i} := \underline{\gamma_i'} - \theta$,*
2. *Eliminer par substitution dans le système, les variable $\underline{\gamma_i}$ et θ.*
3. *Rajouter au système ainsi obtenu, les contraintes des transitions nouvellement sensibilisées dans $M^{\uparrow}$: $\forall t_i \in New(M^{\uparrow}), \quad 0 \leq \underline{\gamma_i} \leq 0$*

La variable θ décrit les dates de tir possibles de t depuis la classe de départ. Il est à noter que, dans le cas des réseaux de Petri avec intervalle statique borné, l'ensemble des systèmes d'horloges distincts que l'on peut construire est fini. Si cette condition n'est pas satisfaite, alors, pour assurer la terminaison de la construction, on doit ajouter à la Définition.4 une étape (d) de relaxation de la classe construite, Cette étape est bien décrite en [8]. Intuitivement, elle revient à noter l'ensemble d'états S par le plus grand ensemble d'horloges décrivant cet ensemble d'états. Ce plus grand ensemble d'horloge n'est pas en générale convexe, mais il est toujours constitué d'une réunion finie d'ensembles convexes. En pratique : la représentation des classes et la relation $\equiv$ sont inchangées, mais une classe pourra avoir plusieurs classes suivantes par la même transition.

Dans le cas où le nombre des transitions concurrentes est important, une méthode alternative récemment proposée : *Normalisation* [41] ; elle permet de préserver convexité du système d'horloge. Moins coûteuse que la relaxation, elle permet de calculer le plus grand ensemble d'horloge préservant états des classes et peut être décrite sous forme de système de différences. Cette étape est appliquée à toutes les classes générées par la Définition.4 citée ci-dessus.

Nous notons :$\langle Q \rangle$ l'ensemble de solutions de Q. Nous définissons l'égalité de deux classes d'états fortes comme suit [8] :

Définition 9 *Soient deux classes d'états fortes $S = (M, Q)$ et $S' = (M', Q')$, et une transitions t sensibilisée par M ou M' avec un intercalle statique borné, alors $S \equiv S'$, ssi :*

1. *elles correspondent au même marquage, $M = M'$;*
2. *l'ensemble des solutions est tel que : $\langle Q \rangle = \langle Q' \rangle$.*

Une construction alternative du graphe de classes d'états fortes avec inclusion est proposée dans [14, 16], notée (CSCG). Une forme plus compacte de celui proposé par Berthomieu, calculé en un temps réduit et ne préservant pas nécessairement les propriétées LTL. Il suffit pour cela de ne pas mémoriser une classe contenue dans une classe déjà construite par la définition Déf.4.

3.3.3.2 Illustration

Considérons le même réseau de Petri, la construction du graphe des classes d'états fortes est comme suit. La classe initiale est représentée par l'état initial, $S_0 = (M_0, Q_0)$ où, $M_0 = p_1, p_2, p_3 \mapsto 1$ et le système d'horloge initial :

$$Q_0 := \begin{cases} 0 \leq \underline{\gamma_1} \leq 0 \\ 0 \leq \underline{\gamma_2} \leq 0 \\ 0 \leq \underline{\gamma_3} \leq 0 \end{cases}$$

Le tir de t_2 à partir de S_0 conduit à la classe $S_2 = (M_2, Q_2)$ définie comme suit :

a) Le système Q_0 est augmenté par les contraintes suivantes :
$0 \leq \theta$
$0 \leq \underline{\gamma_2} + \theta \leq 0$
$0 \leq \underline{\gamma_1} + \theta \leq 3$
$0 \leq \underline{\gamma_3} + \theta \leq 3$

b) $M_2 = p_1, p_3, p_5 \mapsto 1$

c) Le système Q_2 est calculé à partir de Q_0, en appliquant l'algorithme suivant :

1. Les transitions t_1 et t_3 restent sensibilisées lors du tir de t_2, donc pour exprimer la progression du temps il faut remplacer :

   $\underline{\gamma_1} = \underline{\gamma'_1} - \theta$
   $\underline{\gamma_3} = \underline{\gamma'_3} - \theta$
2. Eliminer du système obtenu, les variables : $\theta, \gamma_1, \gamma_3$ et γ_2, ce qui conduit aux calculs suivants :

   $0 \leq \underline{\gamma'_1} - \theta \leq 3$
   $0 \leq \underline{\gamma'_3} - \theta \leq 3$

   Avec : $0 \leq \underline{\gamma_2} \leq 0$
   Et : $0 \leq \theta$

   On obtient :
   $0 \leq \underline{\gamma'_1} \leq 0$
   $0 \leq \underline{\gamma'_3} \leq 0$

3. Aucune transition nouvellement sensibilisée, donc, aucune contrainte à rajouter.

On a :
$Q_2 := \begin{cases} 0 \leq \underline{\gamma_1} \leq 0 \\ 0 \leq \underline{\gamma_3} \leq 0 \end{cases}$

Le tir de $t_3$partir de S_2conduit à la classe S_6, si on considére le graphe le plus compact, (S_7 sinon). La classe $S_6 = (M_6, Q_6)$est obtenue (de la même maniére), comme suit :

a) Le système Q_2 est augmenté par les contraintes suivantes :
$0 \leq \theta$
$0 \leq \underline{\gamma_3} + \theta \leq 3$
$0 \leq \underline{\gamma_1} + \theta \leq 3$

b) $M_6 = p_1, p_5, p_6 \mapsto 1$

c) Le système Q_6 est calculé à partir de Q_2, en appliquant l'algorithme suivant :

1. La transition t_1 reste sensibilisée lors du tir de t_3 :

   $\underline{\gamma_1} = \underline{\gamma'_1} - \theta$

2. Eliminer du système obtenu, les variables : $\theta, \gamma_1, \gamma_3$ avec :

   $0 \leq \underline{\gamma'_1} - \theta \leq 3$

   Avec : $0 \leq \underline{\gamma_3} \leq 3$
   Et : $0 \leq \theta$

   On obtient :

   $0 \leq \underline{\gamma'_1} \leq 3$

3. t_5 est une nouvelle transition sensibilisée, donc, on ajoute son compteur avec la contrainte :
   $0 \leq \underline{\gamma_5} \leq 0$

On va remplacer : γ'_1 par γ_1 pour la suite :

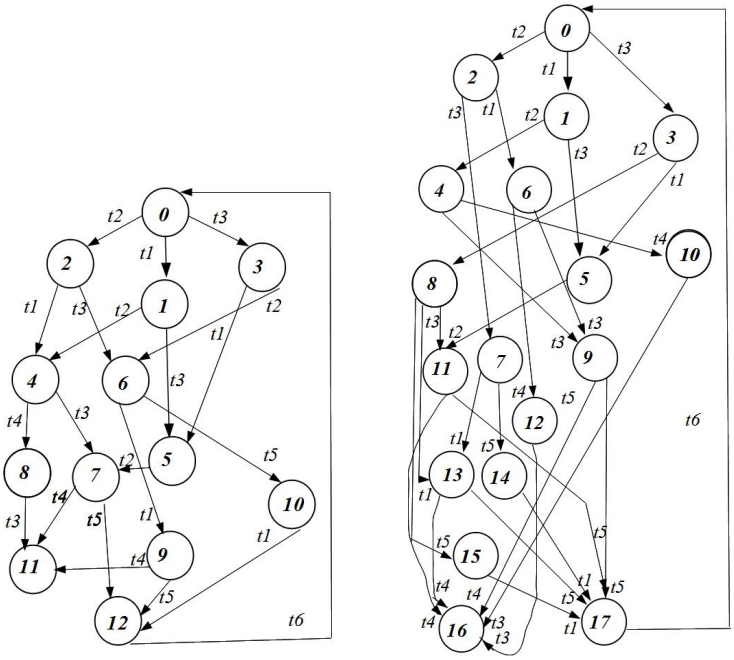
$$Q_6 := \begin{cases} 0 \leq \underline{\gamma_1} \leq 3 \\ 0 \leq \underline{\gamma_5} \leq 0 \end{cases}$$

Comme pour le cas du graphe des classes vu dans la section précedente, on peut mémoriser toutes les classes générées, notées Sl_i par la définition Déf.4 [8], ou prendre en considération l'inclusion [14] Sc_i :

Le graphe des classes d'états fortes des deux stratégies est représenté sur la figure Fig 3.5a), pour les classes avec inclusion notées Sc_i de [14, 16] et 3.5b) pour des classes Sl_i de [8], respéctivement. Le premier admet seulement 13 classes et 21 arcs, alors que le second admet 18 classes et 29 arcs (voir ANNEXE pour détail).

Les correspondances sont les suivantes :

$Sc_0 = Sl_0$; $Sc_1 = Sl_1$; $Sc_2 = Sl_2$; $Sc_3 = Sl_3$; $Sc_4 = (Sl_4, Sl_6)$; $Sc_8 = (Sl_{10}, Sl_{12})$; $Sc_6 = (Sl_7, Sl_8)$; $Sc_7 = (Sl_9, Sl_{11})$; $Sc_{10} = (Sl_{14}, Sl_{15})$. Donc les 5 classes qui n'ont pas été mémorisées dans le graphe compact sont :



a) Préservant états [14] b)Préservant états et LTL [8]

FIGURE 3.5 – Graphe de classes GR_s

$Sl_4, Sl_{10}, Sl_8, Sl_{11}, Sl_{15}$. effectivement on à 13+5 = 18 classes.

C'est important de noter que le graphe de classes d'états fortes constitué de l'ensemble Sci préserve seulement les états, alors que le graphe de classes d'états fortes constitué de l'ensemble Sli préserve états et propriétées LTL. Dans ce qui suit, on considére le graphe des classes d'états fortes du modèle large Sli, noté GR_s.

Enfin, comme le graphe de classes d'état GR, le graphe des classe d'états fortes GR_s préserve les propriétés LTL, mais il est moins compact et son calcul est plus complexe, il permet, de plus de vérifier *l'accessibilité* d'un état. La construction du graphe des classes d'états fortes ne présente donc que peu d'intérêt par elle-même. En fait, elle n'est fournie que parce qu'elle constitue le point de départ de la construction préservant les propriétés de branchement, décrite dans la section suivante.

3.3.4 Graphes de classes préservant états et propriétés CTL

La premiére construction pour un graphe de classes atomique, ou (Atomic Stat Class graph : ASCG) a été proposée par Yoneda *et al* [67] pour les réseaux de Petri avec intervalle statique borné supérieurement. Une autre méthode de construction a été proposée dans [54], mais ne concervant que les propriétés $ACTL$. Cette dernière technique a été étendue pour tout type de $RdPT$ dans [8], que nous rappelons dans cette section.

La bissimulation préserve les propriétés de branchement, le but étant de trouver une abstraction de l'éspace d'état bissimilaire avec le graphe de classe d'un $RdPT$. La technique *standard* de calcul de bissimulation est celle du *raffinement minimal de partition* introduite dans [53].

Définition 10 (Bissimulation) *Calculer une bissimulation depuis une partition initiale P d'états [53], est calculer un raffinement stable Q de P*
Initialement : $Q = P$
Tant que : il existe $A, B \in Q$ tels que $\emptyset \subset A \cap B^{-1} \subset A$
Faire : replacer A par $A_1 = A \cap B^{-1}$ et $A_2 = A - B^{-1}$ dans Q.

Dans [8], le contexte est sensiblement différent de celui supposé ci-dessus, car les ensembles d'états ne sont pas finis, mais la méthode peut être adaptée. Suivant [67], nous appelons *atomique* une classe stable par rapport à toutes ses classes suivantes, c'est-à-dire dont chaque état a un successeur dans chacune de ses classes suivantes. Les classes fortes du (GR_s) sont adéquates comme partition initiale (contrairement aux classes du (GR). Toutefois, le fait que cet ensemble soit un recouvrement plutôt qu'une partition, implique que le résultat final sera généralement non minimal en nombre de blocs, et non unique.

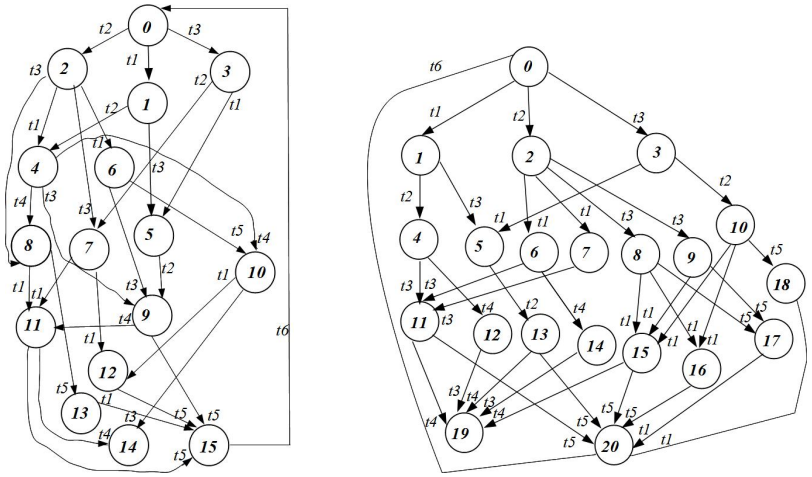
3.3.4.1 Construction du graphe

Le graphe des classes d'états atomiques, noté (GR_a) est obtenu par raffinement du graphe des classes fortes (GR_s). La technique de partition des classes est expliquée en détail dans [8]. Intuitivement, pour chaque transition du graphe courant, on calcule depuis la classe destination $S^{\uparrow}$ l'ensemble des (horloges possibles) des états ayant un successeur dans $S^{\uparrow}$. L'intersection de cet ensemble avec celui capturé par la classe source S définit une partition de S. Les classes atomiques sont considérées modulo l'équivalence $\equiv$, comme les classes fortes.

Le graphe de classes atomiques pour notre exemple est le suivant :

– Avec raffinement du graphe GR_s avec inclusion [67, 8, 14, 15] on a : 16 calsses et 28 arcs (voir Fig :3.6a)) ;

– Avec raffinement du graphe GR_s [67, 8] : 21 classes et 37 arcs, (voir Fig :3.6b)).



a) GR_a selon [67, 8, 14, 15] b) GR_a selon [67, 8]

FIGURE 3.6 – Graphe de classes GR_a préservant les propriétées CTL^*

Il est à noter que les deux graphes représentés ci-dessus sont bissimilaires, le graphe avec inclusion est le plus utilisé en pratique.

On s'interesse dans cette partie aux graphes GR_a GR_s les plus larges dans un but de comparaison. Les classes non-atomiques dans le graphe GR_s (Fig : 3.5b) sont : Sl_6, Sl_7, Sl_{13}. Ces dernières vont étre éclatées, on a les correspondances suivantes :

– $Sl_6 = (A_6; A_7)$.... (1)
– $Sl_7 = (A_8; A_9)$.... (2)
– $Sl_{13} = (A_{15}; A_{16})$....(3)

3.3.4.2 Limites de la méthode

Dans le graphe (GR_a), le problème de branchement est résolu. Dans le cas de la séquence σ=(t_2 ; t_3 ; t_1 ; t_4) à partir de M_0 dans le réseau de la figure Fig.2.4, la classe E_8 est partitionnée en deux classes A_{15} et A_{16}. A partir de A_{15}, on peut tirer soit t_4, soit t_5 ; et de A_{16}, t_4 ne peut pas être franchie. La figure 3.7.a donne la description textuelle du graphe mode GR_a ; la figure 3.7.b présente un fragment du graphe de classes en mode atomique (voir la Fig : 3.6b) pour le graphe complet). Les intervalles sur les arcs de la fig. 3.7.b ne sont pas donnés, mais ont été calculés à la main de façon assez laborieuse. Bien que le problème de branchement soit résolu, le graphe (GR_a) présente deux inconvénients :

1. L'arc correspondant au tir d'une transition t à partir d'une classe A n'a pas suffisamment d'informations temporelles entre les différents événements (tir des transitions précédentes par exemple ;
2. Il ne garde pas des informations dans le passé (une classe peut être atteinte par plusieurs tirs de transitions).

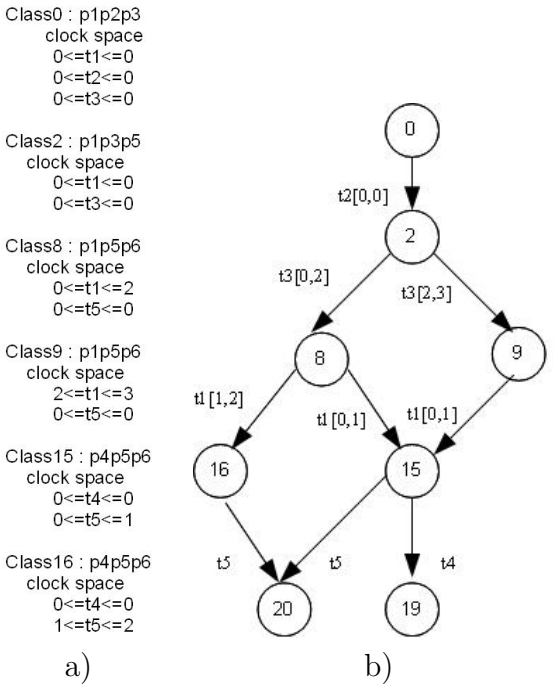


FIGURE 3.7 – Exemple de partitionnement dans GR_s : (2) et (3)

Donc, il serait intéressant d'avoir un graphe des classes pour les *RdPTs* avec sémantique forte qui permette d'associer à tout chemin de ce graphe l'ensemble des séquences de franchissements effectivement franchissables.

Il y a une classe de propriétés d'un grand intérêt pratique, pour laquelle la construction dédiée n'est pas proposée dans [67, 8, 14], celle des propriétés *quantitatives*, comme exprimées dans les logiques temporelles *temporisées*. Pour préserver ces propriétés *quantitatives*, une nouvelle proposition de construction du graphe de classes pour les réseaux de *RdPT* a été proposée par Valette et Cardoso et sera présentée dans la section suivante. Ce graphe permet : *d'associer à tout chemin l'ensemble des séquences de franchissements effectivement franchissables avec les contraintes temporelles minimales que doivent vérifier les dates de franchissement.*

La représentation des informations temporelles (contraintes et intervalles) dans les graphes présentés précedément utilise un système d'inequations *en horloges*. Dans l'approche utilisée dans GraphC, les dates de franchissements de transitions et les contraintes entre ces franchissements sont exprimées par les réseaux de contraintes temporelles (STN : Simple Temporal Network), décrits aussi dans la section qui suit.

3.3.5 Graphe de classes préservant les contraintes quantitatives

Dans cette section, nous allons présenter en détail l'ensemble des régles et définitions permettant d'obtenir à partir d'un *RdPT*, le graphe modeC préservant les contraintes quantitatives, noté GR_c [21, 20] ainsi que l'ensemble de ses classes C_i. L'exemple discuté à la fin de cette section, illustre chaque définition préalablement abordée.

Dans un *RdPT*, les événements à considérer (et donc les variables temporelles associées aux dates de ces évenements) sont :

- La sensibilisation d'une transition ;
- Le début de l'intervalle de tir ;
- La fin de l'intervalle de tir ;
- Le franchissement effectif de la transition.

Nous présentons tout d'abord le Réseau de contraintes complet et minimal.

3.3.5.1 Réseau de contraintes complet et minimal

Définition 11 (Réseau de contraintes simple) - ***Simple Temporal Network*** *(STN)[30] 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 convexes $[c_{mij}, c_{Mij}]$ (sans trou) délimitant la distance possible entre les deux variables v_i et v_j de V. Nous avons donc : $c_{mij} \leq v_j - v_i \leq c_{Mij}$. Nous notons $C\mathcal{A}$ l'ensemble de tous les réseaux de contraintes défini sur V.*

Définition 12 (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 13 (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 $v_j - v_i = c_{ij}$.*

Définition 14 (Algorithme de Floyd-Warshall) *[35, 65] L'algorithme de Floyd-Warshall permet de construire un réseau de contraintes temporelles complet et minimal à partir d'un réseau de contraintes temporelles quelconques ; il commence par construire un réseau complet en associant une contrainte* $]-\infty, \infty]$ *à chaque paire de variables pour laquelle aucune contrainte de distance n'a été spécifiée, ensuite il remplace pour chaque contrainte* C_{ij}, d_{mij} *par la longueur du plus long chemin dans le graphe allant de* v_i *vers* v_j. *Ce calcul étant fait aussi bien pour* C_{ij} *que pour* C_{ji}, *la valeur de la borne maximale de* C_{ij} *est déduite de celle de la borne minimale de* C_{ij} *puisque* $d_{Mij} = -d_{mji}$.

Par exemple, l'algorithme appliqué au réseau de contraintes de la figure(3.8.a)) donne le réseau de contraintes complet et minimal de 3.8. d) ; le b) et c) sont les graphes de coût correspondants, respectivement.

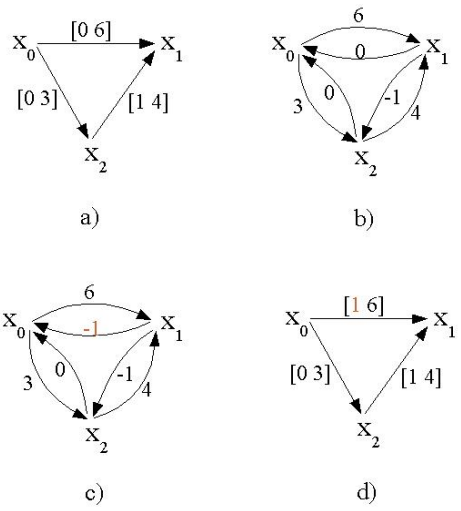


FIGURE 3.8 – Réseau de contraintes complet et minimal

Définition 15 (Intersection) *L'intersection de deux réseaux de contraintes temporelles* $N = (V, C)$ *et* $N' = (V', C')$ *est un réseau de contraintes temporelles* $Nc" = N \bigcap N' = (V", C")$, *où*

- $V" = V \bigcup V'$
- $\forall v_i, v_j \in V"$, $C"_{ij} = C_{ij} \bigcap C'_{ij}$

Les contraintes sont données par l'intersection des intervalles. Lorsqu'une des intersections est vide, le réseau de contraintes temporelles N sera évidement incohérent.

Les réseaux de contraintes temporelles simples sont un cadre adéquat pour analyser les contraintes temporelles générées par *RdPT dans une sémantique d'entrelacement*. Par la suite, nous considérons les variables suivantes :

- x_i^k : variable associée à la date du k^{ieme} franchissement de la transition t_i ;
- y_i : variable associée à la borne supérieure de l'intervalle de tir de t_i lorsqu'elle est sensibilisée.

3.3.5.2 Classe d'états

Considérons l'exécution d'une séquence de franchissement de transitions $\sigma = t_l$; $\cdots$; t_i ; t_j ; $\cdots$; t_n pour un réseau de *RdPT*. Une transition peut être franchie plusieurs fois dans une séquence. Supposons que le franchissement de t_i que nous considérons soit le oi^{eme} et celui de t_j le oj^{eme}. Comme il s'agit de l'exécution d'une séquence particulière, les variables associées aux dates de franchissement des transitions soit par exemple x_i^{oi} pour t_i et x_j^{oj} pour t_j ont une valeur précise.

Nous allons regrouper dans une même classe toutes les exécutions de σ cohérentes avec les contraintes temporelles définies par le *RdPT* (sans mémoire des sensibilisations passées et sans hypothèse du serveur unique pour le franchissement des transitions) [21]. 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^{oj} vis-à-vis des variables associées aux franchissements passés.

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, seulement un fragment de ce réseau de contraintes temporelles, appelé Nc, est gardé. Cela est l'élément clé de la définition de la classe mode C.

La procédure de construction du réseau de contraintes temporelles que doit vérifier la variable x_j^{oj} (correspondant au franchissement de la transition t_j), est présentée dans la section suivante.

3.3.5.3 Réseau de contraintes N_t 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 le graphe de classes, un arc entre deux classes correspond au franchissement d'une transition. Dans ce cas présent, 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 N_t *complet et minimal* comprenant les variables de la classe d'origine

de l'arc. L'information temporelle représentée par ce réseau dépend du réseau de contraintes N_c de la classe d'origine.

Soit t_j une transition parmi les n transitions franchissables dans la classe C (atteinte par le franchissement de t_j). Soit t_l, $l \neq j$, les autres $n-1$ transitions sensibilisées dans la classe C. Soit $Nt_{j,c}$ le réseau de contraintes temporelles délimitant le franchissement de t_j à partir de la classe C pour atteindre la classe $C^{\uparrow}$.

3.3.5.4 Construction du graphe

Définition 16 *Le graphe de classes modeC d'un RdPT [21, 20], noté GR_c, est un tripplet $(C\mathcal{C}, \mathcal{C}_0, \twoheadrightarrow)$ où :*
- $C\mathcal{C}$ est l'ensemble des classes accessibles dans GR_c;
- $C_0 = (M_0, Nc_0)$ est une classe de $C\mathcal{C}$ appelée classe initiale où :
  – M_0 est le marquage initial du RdPT ;
  – Nc_0 est le réseau de contraintes temporelles composé par la variable x_0 représentant l'origine de temps.
- $\twoheadrightarrow$ est la relation de transition entre classes définie sur $C\mathcal{C} \times (C\mathcal{A}, T) \times C\mathcal{C}$, telle que :
$((M, Nc), (Nt_{j,c}, t_j), (M^{\uparrow}, Nc^{\uparrow})) \in \twoheadrightarrow$, si et seulement si :
$\forall p \in P, M(p) \leq Pre(p, t_j)$, et le réseau $NF_{j,c}$ est consistant, tel que $NF_{j,c}$ est obtenu comme suit :

1. *Au départ, $NF_{j,c} = Nc$, le réseau de contraintes temporelles de la classe origine $\mathcal{C}$,*
2. *Ajouter à $NF_{j,c}$ la variable x_j^{oj} (associée au franchissement de t_j) ; ajouter l'intervalle statique de t_j, $I(t_j)$, comme contrainte entre x_j^{oj} et x_{sj}^{osj} (associée au franchissement de la transition ayant sensibilisé t_j),*
3. *Ajouter à $NF_{j,c}$ les variables y_l^{ol} ($l \neq j$) correspondantes aux dates maximales de franchissement des autres transitions sensibilisées dans la classe $\mathcal{C}(t_l)$. Ajouter comme contrainte entre chaque couple (x_{sl}^{osl}, y_l^{ol}) l'intervalle (singleton) $[d_{Ml}, d_{Ml}]$ 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 de variables x_j^{oj} et y_l^{ol} ($l \neq j$), pour exprimer le fait que t_j doit être franchie* avant *la borne maximale de t_l,*

La classe accessible est obtenue comme suit :

a) $\forall p \in P$, $M^{\uparrow}(p) := M(p) - Pre(p, t_j) + Post(p, t_j)$.

b) *$Nt_{j,c}$ est obtenu à partir de $NF_{j,c}$ en effaçant les variables y_l^{ol} et les contraintes auxquelles elles sont directement reliées.*

c) *$Nc^{\uparrow}$ est un fragment $Nt_{j,c}$ et est le réseau de contraintes temporelles simple comprenant les variables et les contraintes suivantes :*

1. *la variable associée à la date du dernier franchissement de la transition x_i^{oi},*
2. *pour chaque transition t_k sensibilisée par $M^{\uparrow}$, la variable associée au franchissement de la transition ayant provoqué cette sensibilisation, soit $x_{s(k)}^{osk}$ $(k = 1 \dots n)$.*
3. *toutes les contraintes temporelles liant ces variables sous la forme d'un réseau complet et minimal.*

L'origine de temps est l'événement qui a sensibilisé toutes les n_0 transitions sensibilisées dans la classe C_0, et d'une certaine façon, est considérée comme le début du monde.

La première étape correspond au calcul du réseau de contraintes délimitant le franchissement de t_j à partir de la classe origine C : (3) est imposée par la sémantique d'entrelacement et (4) est imposée par la sémantique forte. Une fois l'algorithme de Floyd-Warshall appliqué, les variables x_{si}^{osi} et y_l^{ol} sont redondantes (étape b) ;

L'étape (a) correspond au calcul usuel du marquage $M^{\uparrow}$;

L'étape (c) implique que $Nc^{\uparrow}$ est un fragment de $Nt_{j,c}$, $Nc^{\uparrow} \subseteq Nt_{j,c}$.

Pour toutes les classes $\mathcal{C} \neq \mathcal{C}_0$, il existe deux cas particuliers :

- Deux transitions t_1 and t_2 ayant été sensibilisées par la même transition t_a, les variables correspondantes x_{s1}^{os1} et x_{s2}^{os2} sont les mêmes, donc $x_{s1}^{os1} = x_{s2}^{os2} = x_a^{oa}$;
- Le dernier franchissement (représenté par la date x_i^{oi}), est aussi l'événement qui sensibilise la transition t_k dans cette classe (représentée par la variable $x_{s(k)}^{osk}$), les deux variables sont les mêmes, $x_i^{oi} = x_{s(k)}^{osk}$.

Par exemple, considérons le RdPT de la figure Fig 3.2a), avec le tir de la séquence $t_2; t_3; t_1$ à partir de la classe initiale $\mathcal{C}_0$ avec le marquage initial $M_0 : \{p_1, p_2, p_3\} \mapsto 1$. La classe $\mathcal{C}$ atteinte par la franchissement de cette séquence a comme marquage $M_0 : \{p_4, p_5, p_6\} \mapsto 1$. Les transitions sensibilisées par ce marquage sont t_4 et t_5. Les événement suivants doivent être considérés pour construire le réseau de contraintes Nc (Def. 16) :

- La derniere transition franchie dans cette séquence est $t_i = t_1$, et la variable correspondante est x_1.

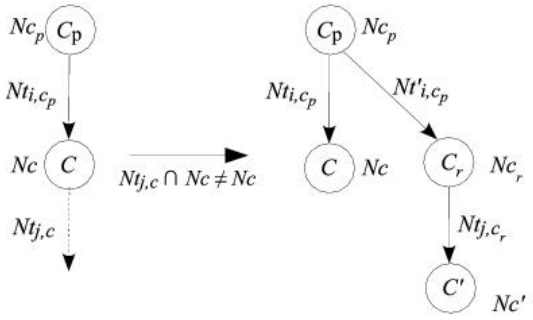


FIGURE 3.9 – Classe restreinte $\mathcal{C}r$ de la classe $\mathcal{C}$

– t_4 a été sensibilisée par le tir de t_1, donc $x_{s4}^{os4} = x_1$; t_5 a été sensibilisée par le tir de t_3, donc $x_{s5}^{os5} = x_3$.

Il manque les valeurs des contraintes entre les événements x_1 et x_3, pour avoir le réseau de contraintes temporelles Nc (complet et minimal). Nc est un fragment du réseau de contraintes qui délimite le dernier franchissement (t_1 dans cet exemple).

Remarque 4 *Il est à noter que le paragraphe ci-dessus n'est qu'un exemple illustratif déterminant les variables et contraintes qui constituent la classe courante. Le calcul des classes de la séquence σ et des classes restreintes est expliqué en détail un peu plus loin.*

3.3.5.5 Les classes restreintes

Considérons le fragment de graphe de classes de la figure 3.9. Le réseau des contraintes temporelles entre les événements d'une classe $\mathcal{C}$ est obtenu à partir de l'ensemble des contraintes délimitant le franchissement de t_i (la transition qui conduit le système vers la classe $\mathcal{C}$). En effet, le réseau de contraintes temporelles Nc de la classe $\mathcal{C}$ est un fragment de Nt_{i,c_p}, le réseau de contraintes associé à l'arc entre $\mathcal{C}_p$ et $\mathcal{C}$.

Après l'application de l'algorithme de Floyd-Warshall pour $Nt_{j,c}$, certaines des contraintes $C_{k,l}$ entre deux nœuds x_k et x_l peuvent se trouver modifiées, donc plus restreintes que leurs valeurs initiales dans le réseau Nc de la classe $\mathcal{C}$. Cela implique que la transition t_j 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 $\mathcal{C}$, notée $\mathcal{C}r$, qui est restreinte au franchissement d'une transition déterminée.

Définition 17 *La classe restreinte $\mathcal{C}' = (M', Nc')$ de la classe $\mathcal{C} = (M, Nc)$ est définie par*

– $M' = M$,
– Nc' *est le réseau* $Nt_{j,c} \cap Nc$ *après l'application de l'algorithme de Floyd-Warshall.*

La classe $\mathcal{C}'$ caractérise les états à partir desquels la transition t_j est franchissable ; ces états ont été obtenus par le franchissement de la transition t_i (précédant t_j dans la séquence $\tau = t_1; ...; t_i; t_j$). Cela veut dire que cette date est délimitée par le réseau $Nt'_{i,c_p} = Nt_{i,c_p} \cap Nc'$, qui, après une nouvelle application de Floyd-Warshall, peut être tel que $Ntr_{i,c_p} \cap Nc_p \neq Nc_p$: cela implique que la transition t_i peut elle même induire une restriction de la classe $\mathcal{C}_p$, notée $\mathcal{C}_{pr}$, à partir de laquelle t_i est franchie et ainsi de suite.

Si Nt_{i,c_p} est un STN avec des variables qui représentent les différents franchissements de la même transition, alors le Nc' correspond au derniers franchissements de de cette transition, c'est à dire les variables avec les plus grands exposants.

On ne peut pas considérer la notion de classes restreintes pour deux classes C_1 et C_2 dans le cas ou au moins une n'a pas été créée pendant l'étape 2 du calcul même si : elles ont le même marquage $M_1 = M_2$ et que le STN de l'une est inclus dans le STN de l'autre : $Nc_1 \subset Nc_2$ ou $Nc_2 \subset Nc_1$.

3.3.5.6 Les classes équivalentes

Chaque fois qu'une nouvelle classe est créée, il faut vérifier s'il n'existe pas déjà une classe équivalente dans le graphe. La définition de deux classes équivalentes est donnée par la définition 19. Le cas particulier de l'équivalence avec la classe initiale est donnée par la définition 18.

Définition 18 *Une classes* $\mathcal{C} = (M, N_c)$ *est définie équivalente avec la classe initiale* $\mathcal{C}_0 = (M_0, x_0)$*, si*

1. *elles correspondent au même marquage,* $M = M_0$ *;*
2. *l'ensemble des variables* X *de* Nc *est un singleton,* $X = \{x_k\}$ *(pas de mémoire du passé).*

La définition 18 montre que le franchissement de la transition t_k ramène le système vers le marquage initial M_0 (sensibilise donc toutes les transitions initialement sensibilisées). Cela correspond à l'état du *début du monde.*

Définition 19 *Deux classes* $\mathcal{C} = (M, N_c)$ *et* $\mathcal{C}' = (M', N'_c)$*, avec* $\mathcal{C} \neq \mathcal{C}_0$*,* $\mathcal{C}' \neq \mathcal{C}_0$*,* N_c *= (X, C) et* N'_c *= (X', C'), sont définies équivalentes si.*

1. *elles correspondent au même marquage,* $M = M'$*;*

2. *il existe une bijection τ entre les variables des deux classes X et X' telle que :*
   - $x'^{ok}_k = \tau(x^{oi}_i)$ *implique* $k = i$ *(les variables sont les dates de franchissement d'une même transition),*
   - *si* $x'_i = \tau(x_i)$ *et* $x'_j = \tau(x_j)$, C'_{ij} *(contrainte temporelle entre* x'_i *et* x'_j*) est égale (au sens des intervalles) à* C_{ij} *(contrainte temporelle entre* x_i *et* x_j*).*

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

3.3.5.7 Illustration

Considérons l'exemple du *RdPT* dans la figure Fig 3.2a). La représentation du graphe des classes mode C est donnée par la figure 3.10. Selon ce graphe, on voit qu'il comporte des exemples de blocage mortel $(\mathcal{C}_{19}, \mathcal{C}_{21})$ mais aussi des possibilités de fonctionnement cyclique et donc de séquences de longueur infinie. Les classes avec leur marquage et leur réseau de contraintes sous forme textuelle (En ANNEXE) ; sont représentées dans le tableau B.4.a) et les réseaux de contraintes temporelles associés aux arcs sont représentés dans le tableau B.4.b).

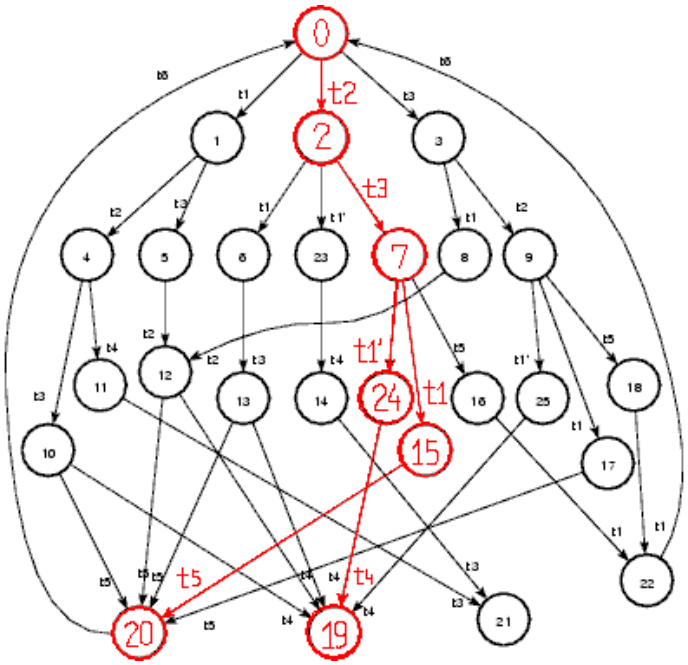


FIGURE 3.10 – Graphe de classes GR_c préservant CTL^* [21, 20]

Considérons la même séquence $\sigma = t_2; t_3; t_1; t_4$ dans le graphe de la figure 3.10. Elle aboutit au blocage mortel $M_{19} = p_6 p_7$. La classe initiale $\mathcal{C}_0$ (voir ANNEXE) correspondant à l'état initial, est définie par le marquage initial $M_0 : p_1, p_2, p_3 \mapsto 1$ et un réseau de contraintes (Nc_0) formé d'un seul nœud x_0 correspondant à un événement initial de création des jetons du marquage initial. Il représente l'origine de temps.

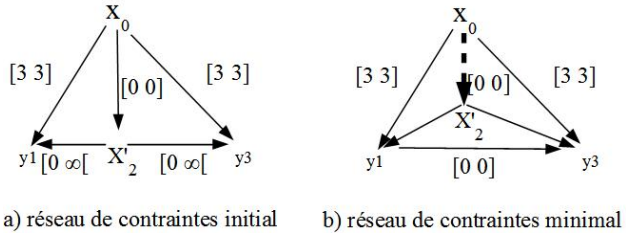


FIGURE 3.11 – Réseau de contraintes $Nt_{2,0}$ pour franchissement de t_2

A partir de cette classe on va appliquer l'algorithme de construction du graphe GR_c et on considère le franchissement de la transition t_2 dans la séquence σ à la date x_2. Les contraintes liées à ce franchissement sont définies par le réseau de contrainte $Nt_{2,0}$ (voir tableau B.4.b) qui initialement est donné par la figure 3.11.a, puis après application de l'algorithme de Floyd-Warshall (permettant d'obtenir un réseau complet minimal) par la figure 3.11.b. À partir de $Nt_{2,0}$ on trouve alors la classe $\mathcal{C}_2$. Son réseau de contraintes (Nc_2) est l'arc en pointillés gras de la figure 3.11.b.

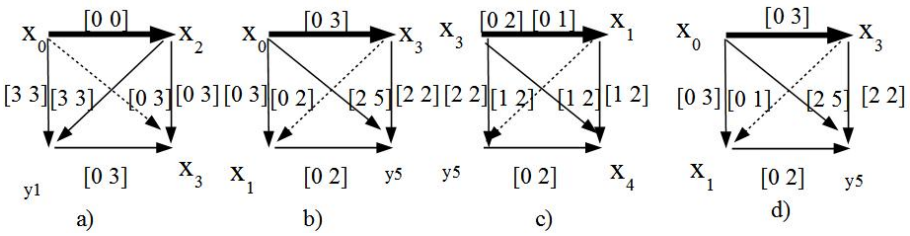


FIGURE 3.12 – Réseaux temporels $a) Nt_{3,2}$, $b) Nt_{1,7}$, $c) Nt_{4,15}$ et $d) Nt'_{1,7}$

Maintenant la poursuite de la séquence σ implique le franchissement de la transition t_3. Les contraintes devant être vérifiées par rapport à la date de franchissement sont données dans la figure 3.12.a. Elles font passer de la classe $\mathcal{C}_2$ (contrainte en gras), à la classe $\mathcal{C}_7$ (contrainte en pointillés gras).

Ensuite, on considère le franchissement de la transition t_1 à partir de $\mathcal{C}_7$. Le réseau de contraintes complet et minimal $Nt_{1,7}$ délimitant la date x_1 de ce franchissement est donné par la figure 3.12.b. Nous aboutissons à la classe $\mathcal{C}_{15}$ (pointillés gras dans la figure 3.12.b).

Jusque là aucune restriction n'est définie, la pouruite de la méthode représente la phase (1) du calcul du graphe modeC, jusqu'a arriver à la classe $\mathcal{C}_{20}$. A partir de la on va considérer les restrictions trouvées durant cette derniére phase et voir s'il y a génération de nouvelles classes notées, *restreintes*, le paragraphe suivant représente un exemple de calcul des classes restreintes : $\mathcal{C}_{24}$ classe restreinte de $\mathcal{C}_{15}$.

Le franchissement de la transition t_4 à partir de $\mathcal{C}_{15}$, est défini par le réseau de contraintes $Nt_{4,15}$(Fig. 3.12.c) délimitant la date x_4 de ce franchissement. Le $Nt_{4,15}$ initial est formé par les nœuds x_1, x_3, x_4 et y_5 et les contraintes $C_{31} = [0\ 2]$, $C_{35} = [2\ 2]$, $C_{14} = [1\ 2]$ et $C_{45} = [0\ \infty]$. Le $Nt_{4,15}$ complet et minimal présente, pour l'arc (x_3, x_1), $C_{31} = [0\ 1]$ au lieu de $[0\ 2]$, ce qui représente une restriction temporelle dans la classe $\mathcal{C}_{15}$. En effet, Nc_{15} est donné par $C_{31} = [0\ 2]$ ou $0 \leq x_1 - x_3 \leq 2$. 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, x_1) définit donc une *classe restreinte*, la classe $\mathcal{C}_{24}$ a le même marquage que $\mathcal{C}_{15}$.marquage, et $\mathcal{C}_{24}.Nt$ 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.*

Une fois la classe restreinte $\mathcal{C}_{24}$ est créee, un nouveau arc entre la classe précédente $\mathcal{C}_7$ et $\mathcal{C}_{24}$ doit être établi avec le réseau de contraintes temporelles délimitant le franchissement de t_1, ce réseau est donné par $Nt'_{1,7} = Nt_{1,7} \cap Nc_{24}$ (Fig. 3.12.d). Comme $Nt'_{1,7} \cap Nc_7 = Nc_7$, la propagation d'amont s'arrête. Le chemin en rouge est celui défini par la génération de la classe $\mathcal{C}_{24}$ nouvellement calculée.

Pour retrouver la classe $\mathcal{C}_0$ par le franchissement de la transition t_6 à partir des classes $\mathcal{C}_{20}$ et $\mathcal{C}_{22}$, 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 $\mathcal{C}_{15}$ et $\mathcal{C}_{24}$) pour retrouver des classes équivalentes. Cela aurait inutilement augmenté le nombre de classes.

On a les classes restrientes suivantes : $\mathcal{C}_{21}, \mathcal{C}_{22}, \mathcal{C}_{23}, \mathcal{C}_{24}, \mathcal{C}_{25}, \mathcal{C}_{26}$ et les correspondances suivantes :

- $\mathcal{C}_{23}$ est la classe restreinte de $\mathcal{C}_6$; avec $(x_0, x_3) = [0\ \ 2]$ au lieu de $[0\ \ 3]$;
- $\mathcal{C}_{24}$ est la classe restreinte de $\mathcal{C}_{15}$; avec $(x_3, x_1) = [0\ \ 1]$ au lieu de $[0\ \ 2]$;
- $\mathcal{C}_{25}$ est la classe restreinte de $\mathcal{C}_{17}$; avec $(x_2, x_1) = [0\ \ 1]$ au lieu de $[0\ \ 2]$;

3.3.5.8 Caractérisation des séquences

Les contraintes dans la classe (Fig. B.4.a) 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 Nt correspondants. Ainsi, pour obtenir le graphe des contraintes temporelles de la séquence σ (Fig. 3.13) il faut concatener $Nt_{2,0}, Nt_{3,2}, Nt'_{1,7}$ et $Nt_{4,24}$. Les deux arcs entre x_3 et x_1 illustrent le fait que le réseau Nc_{24} de la classe $\mathcal{C}_{24}$ est égal à l'intersection de $Nt'_{1,7}$ et $Nt_{4,24}$. 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 et x_4. Par contre, il se trouve que la contrainte entre x_0 et x_3 est redondante.

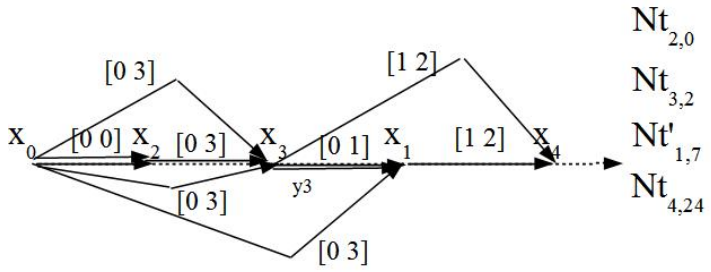


FIGURE 3.13 – Réseau de contraintes pour la séquence σ

3.3.6 Quelle correspondance entre graphes basés classe d'états ? : (GR, GR_a et GR_c)

Nous allons maintenant comparer *qualitativement* parlant, les différents graphes obtenus par application des méthodes décrites précédemment (basées graphe de classe d'états) et voir quelles sont les principales correspondances entre les classes de graphes : GR figure Fig. 3.2b), GR_a figure Fig. 3.6b) et GR_c figure Fig. 3.10, en nous appuyant sur le même réseau de Petri de la figure Fig. 3.2a).

Respectivement on va noter : Ai, Ei et $\mathcal{C}_i$ l'ensemble des classes i constituant les graphes : GR_a, GR et GR_c. Le graphe GR donne pour chaque classe, le marquage et le domaine temporel (intervalle de tir pour chaque transition franchissable et certaines contraintes entre les tirs). Le GR_a est un raffinement du GR. Certaines classes de GR_a correspondent à des partitions des classes du GR. Les correspondances sont les suivantes :
$E_0 = A_0; E_1 = A_1; E_2 = A_2; E_3 = A_3; E_4 = (A_4, A_6, A_7); E_5 = A_5; E_6 = (A_8, A_9, A_{10}); E_7 = A_{11}; E_8 = (A_{12}, A_{14}); E_9 = (A_{13}, A_{15}, A_{16}); E_{10} = (A_{17}, A_{18}); E_{11} = A_{19}; E_{12} = A_{20}$ (nous ne présentons pas ici le détail de ces classes).

Le graphe GR_c représente aussi un raffinement du GR, mais les classes $\mathcal{C}_i$ ne correspondent pas nécessairement à une partition des états à cause des classes restreintes. Nous avons les correspondances suivantes :

$E_0 = \mathcal{C}_0$; $E_1 = \mathcal{C}_1$; $E_2 = \mathcal{C}_2$; $E_3 = \mathcal{C}_3$; $E_4 = (\mathcal{C}_4, \mathcal{C}_6, \mathcal{C}_{23})$; $E_5 = (\mathcal{C}_5, \mathcal{C}_8)$; $E_6 = (\mathcal{C}_7, \mathcal{C}_9)$; $E_7 = (\mathcal{C}_{10}, \mathcal{C}_{13})$; $E_8 = (\mathcal{C}_{11} ; \mathcal{C}_{14})$;$E_9 = (\mathcal{C}_{12}, \mathcal{C}_{15}, \mathcal{C}_{17}, \mathcal{C}_{24}, \mathcal{C}_{25})$; $E_{10} = (\mathcal{C}_{16}, \mathcal{C}_{18})$; $E_{11} = (\mathcal{C}_{19}, \mathcal{C}_{21})$; $E_{12} = (\mathcal{C}_{20}, \mathcal{C}_{22})$.

3.3.6.1 Différences dans la prise en compte des évolutions futures

Prenons le cas des classes équivalentes à la classe E_9 c'est-à-dire les classes $\mathcal{C}_{12}$, $\mathcal{C}_{15}$, $\mathcal{C}_{17}$, $\mathcal{C}_{24}$ et $\mathcal{C}_{25}$ dans le GR_c et A_{13}, A_{15} et A_{16} dans le GR_a. La différence essentielle entre le GR_a et le GR_c provient du fait que la décomposition des classes du GR en des classes plus petites n'a pas le même objectif lorsque l'on se focalise sur le futur.

Prenons le cas des classes $\mathcal{C}_{15}$ et $\mathcal{C}_{24}$. Les états de la classe $\mathcal{C}_{15}$ 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 $\mathcal{C}_{24}$ 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 $\mathcal{C}_{24}$ 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 différentiation 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, A_{15} regroupe les états qui sont tels qu'il y a conflit entre t_4 et t_5 (les deux sont simultanément franchissables) alors que A_{16} regroupe tous les états qui ne peuvent se prolonger que par t_5 (t_4 n'est pas franchissable).

Dans le GR_a les classes réalisent une partition des états. Un état ne peut appartenir à la fois à la classe A_{15} et à la classe A_{16} car les deux situations sont en exclusion mutuelle. Ce n'est pas du tout le cas dans le mode GR_c puisque, au contraire, tous les états de la classe $\mathcal{C}_{24}$ sont également des états de la classe $\mathcal{C}_{15}$.

Les états caractérisés par la classe $\mathcal{C}_{15}$ (et donc également ceux de la classe $\mathcal{C}_{24}$) font partie des états regroupés par les deux classes A_{15} et A_{16} car ils résultent du franchissement de la séquence t_2 ; t_3 ; t_1. Mais les classes A15 et A16 contiennent également des états résultant de la séquence t_3 ; t_2 ; t_1 ce qui n'est pas le cas de la classe $\mathcal{C}_{15}$. Ces états sont regroupés dans les classes $\mathcal{C}_{17}$ et $\mathcal{C}_{25}$. Nous voyons donc que la prise en compte du passé est également différente dans les GR_a et GR_c.

3.3.6.2 Différences dans la mémorisation du passé

Au vu des considérations concernant la prise en compte du futur, on pourrait penser que le GR_c devrait comporter moins de nœuds que le graphe GR_a, or c'est le contraire qui se produit. Cela vient du fait que la notion d'équivalence entre

classes est plus restrictive que celle du GR_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 GR ils sont tous dans une seule classe, la classe E_{12}, que le franchissement précédent soit celui de t_1 ou de t_5. Dans le cas GR_c, nous avons deux classes, $\mathcal{C}_{20}$ et $\mathcal{C}_{22}$. 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 $\mathcal{C}_{20}$) ou la date du dernier franchissement de t_1 (une variable x_1^i dans $\mathcal{C}_{22}$).

La mémoire du passé peut remonter au dela du dernier événement. Par exemple, les classes $\mathcal{C}_{10}$ et $\mathcal{C}_{13}$ 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 $\mathcal{C}_{10}$ c'est le franchissement préalable de t_2 qui a provoqué la sensibilisation de t_4, et sa mémoire doit être conservée. Dans le cas de $\mathcal{C}_{13}$, c'est celui de t_1. En conséquence, les classes $\mathcal{C}_{10}$ et $\mathcal{C}_{13}$ ne peuvent pas être confondues pour nous alors qu'elles le sont pour le GR_a sous la forme de la classe $A_{11} = E_7$. La différentiation entre les classes $\mathcal{C}_{15}$ et $\mathcal{C}_{17}$ s'explique de la même manière.

Nous allons présenter dans la suite de ce chapitre, les méthodes basées sur la notion de régions géométriques.

3.4 Les méthodes basées sur les Régions Géométriques

3.4.1 Graphe des Régions

Le graphe de régions proposé par Yoneda *et al* [67] pour les $RdPT$ (initialement proposée pour les TA [6]), se construit en deux étapes : la première génère un graphe initiale, la deuxième applique itérativement un algorithme permettant l'éclatement (partitionnement) des classes non atomiques du graphe initiale, le résultat est un graphe de région préservant l'atomicité et les propriétés CTL. Les auteurs ont prouvé que si les propriétés linéaires sont préservées et toutes les classes sont atomiques alors le graphe finale préserve les propriétés CTL^* aussi.

3.4.1.1 Construction du graphe

Une classe est caractérisée par le couple $R = (M, d)$ associée à la séquence σ, M étant le marquage atteint aprés le tir de la séquence σ à partir du marquage initial M_0 et d est un sous-systéme [8] de D défini plus-haut Déf.1. La classe initiale est

représentée par $(M_0, \emptyset)$ associée à l'ensemble vide de séquences de tir. η étant l'origine du temps. Chaque transition t_f sensibilisée par M a un 'parent', qui est la derniére transition t dans la séquence σ. On note : $t = parent(t_f, (M, d))$;

La classe $(M^{\uparrow}, d^{\uparrow})$ obtenue aprés le tir de t_f à partir de la classe (M, d) est calculée par l'algorithme suivant :

Définition 20 *Le graphe des régions géométriques, noté GR_r d'un RdPT [67], est un triplet $(CR, R^0, \rightharpoonup)$ où :*
- CR est l'ensemble des classes accessibles dans GR_r;
- $R^0 = (M^0, \emptyset)$ est une classe de CR appelée classe initiale, associée à la séquence vide.
- $\rightharpoonup$ est la relation de transition entre classes définie sur $CR \times T \times CR$, telle que : $((M, d), t_f, (M^{\uparrow}, d^{\uparrow})) \in \rightharpoonup$, si et seulement si :

a) *Le système augmenté par les contraintes de tir de t_f que nous écrivons :*
$(\forall t \in Te(M), \quad d \cup \{parent(t_f, (M, d)) + tmin(t_f) \leq parent(t, (M, d)) + tmax(t)\})$ *est consistant.*

b) $\forall p \in P,\ M^{\uparrow}(p) := M(p) - Pre(p, t_f) + Post(p, t_f)$.

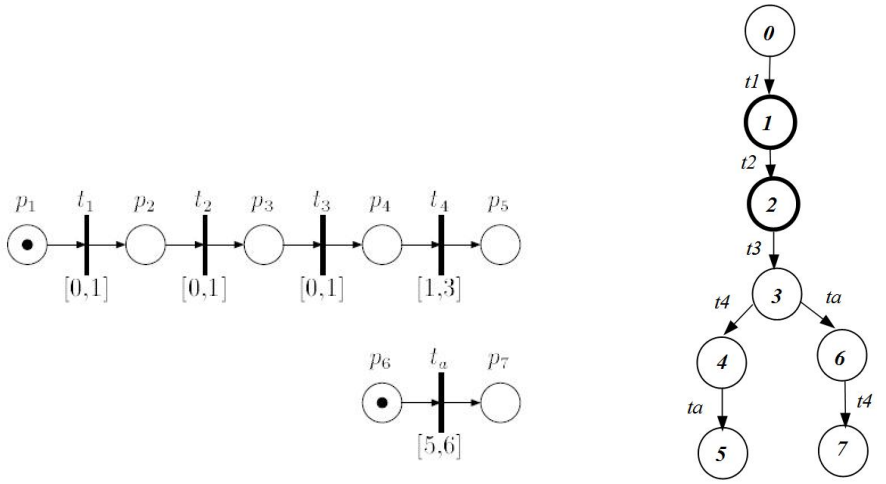
c) *Le système $d^{\uparrow}$ est calculé à partir de D_{sub}, en appliquant l'algorithme suivant :*

1. *Calculer la valeur V_1 tel que : $V_1 = \{tmin(t_f) \leq t_f - paren(t_f, (M, d)) \leq tmax(t_f)\}$. (exprimant les contraintes temporelles pour le tir de t_f : t_f doit être tirée entre $tmin(t_f)$ et $tmax(t_f)$).*
2. *Calculer la valeur V_2 tel que : $V_2 = \{t_f \leq parent(t_i, (M, d)) + tmax(t_i)\}$ ($\forall t_i \in (M), (t_f)$ doit être tirée avant toutes les autres tranbsitions sensibilisées par M).*
3. *Déterminer le nouveau système $d^{\uparrow}$:*
$\forall t_i \in New(M^{\uparrow}),\ d^{\uparrow} = d \cup V_1 \cup V_2$

a) exprime les contraintes de tirabilité de $t_f, b)$ étant le calcul usuel du marquage atteint aprés tir de t_f et les valeurs V_1 et V_2 expriment réspectivement que t_f doit être tirée au plus tôt aprés la date du début de tir de t_f et au plus tard avant la fin de l'interval de tir, que t_f doit être tirée avant toutes les autres transitions concurrentes.

La deuxiéme étape étant le calcul d'un nouveau graphe à partir du graphe initial en éclatant les classes non atomiques, le résultat est un graphe préservant les propriétées CTL (ou CTL^*) du graphe.

Intuitivement, dans la deuxiéme étape, toutes les classe non stables, sont partitionnées jusqu' à ce qu'il n'y est plus de classes non atomiques. Les auteurs montrent que si (M, d) est une classe non atomique, alors il existe forcement une



a)Le Réseau de Petri [67] b)Graphe de région : étape1

FIGURE 3.14 – Graphe de région selon [67]

contrainte linéaire ρ, non redondante dans d tel que, pour certains successuers $(M^{\uparrow}, d^{\uparrow})$ de (M, d), ρ est necessaire pour que la classe (M, D_{sub}) ait un successeur dans $(M^{\uparrow}, d^{\uparrow})$. Dans ce cas, (M, D_{sub}) est patitionnée en sous-classes : $(M, d \wedge \rho)$ et $(M, d \wedge \neg\rho)$ (l'algorithme pour le calcul de ρ est détaillé avec exemple dans [67].

3.4.1.2 Principe en deux étapes

Considérons l'exemple de la figure Fig 3.14 .a), présenté auparavant dans [67] :

Le graphe de région de la premiére étape Fig 3.14 .b), admet 8 classes et 7 arcs. Les classes en gras constituent des classes *non stables* $(1, 2)$, qui doivent être traitées durant la deuxiéme partie du processus constituant le *partitionnement* de ces derniéres.

Aprés partitionnement des classes non atomiques (1) et (2), on obtient un graphe plus grand avec 15 classes et 14 transitions Fig. 3.15 .b).

3.4.1.3 Illustration

Construisons quelques classes du réseau de Petri précédent (Fig 3.2a)). L'état initial est défini par $R_0 = (M_0, \emptyset)$ associé à la séquence vide, $M_0 = p_1, p_2, p_3 \mapsto 1$. Le tir de t_2 de la séquence σ conduit à la classe $R_2 = (M_2, d_2)$ associée à la séquence t_2, avec $M_2 = p_1, p_3, p_5 \mapsto 1$, et d_2 défini comme suit :

1. $0 \leq t_2 - \eta \leq 0$

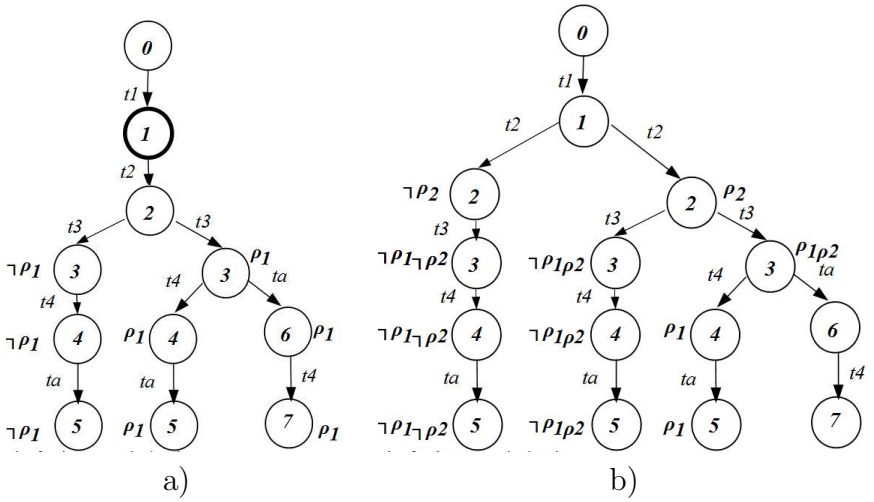


FIGURE 3.15 – Graphe de région GR_r : a) partitionnement (2), b) partitionnement (1)

2. $t_2 \leq \eta + 3$
   $t_2 \leq \eta + 3$

3. $d_2 := \begin{pmatrix} 0 \leq t_2 - \eta \leq 0 \\ t_2 \leq \eta + 3 \end{pmatrix}$

On a alors :

$$R_2 := \begin{cases} M_2 = p_1, p_3, p_5 \mapsto 1 \\ \sigma = t_2 \\ d_2 := \begin{pmatrix} 0 \leq t_2 - \eta \leq 0 \\ t_2 \leq \eta + 3 \end{pmatrix} \end{cases}$$

Le tir de t_3 à partir de cette classe, conduit à la classe $R_6 = (M_6, d_6)$ associée à la séquence t_2, t_3, avec : $M_6 = p_1, p_5, p_6 \mapsto 1$ et d_6 obtenu par :

1. $0 \leq t_3 - t_2 \leq 3$

2. $t_3 \leq \eta + 3$

3. $d_6 := \begin{pmatrix} 0 \leq t_2 - \eta \leq 0 \\ 0 \leq t_3 - t_2 \leq 3 \\ t_2 \leq \eta + 3 \\ t_3 \leq \eta + 3 \end{pmatrix}$.

On a alors :

$$R_6 := \left\{ \begin{array}{l} M_6 = p_1, p_5, p_6 \mapsto 1 \\ \sigma = t_2, t_3 \\ d_6 := \left(\begin{array}{l} 0 \leq t_2 - \eta \leq 0 \\ 0 \leq t_3 - t_2 \leq 3 \\ t_2 \leq \eta + 3 \\ t_3 \leq \eta + 3 \end{array} \right) \end{array} \right. .$$

L'inconvénient d'une telle méthode est, comme on le voit : la taille du graphe, qui est plus grand que necéssaire, ceci est du au fait que : on mémorise à chaque étape le passé, en plus, la relation d'équivalence est associée à la séquence de tir plutôt qu'au contenu des classes. Le GR_s est un graphe altérnatif ne souffrant pas de cet inconvénient. Concernant le partitionnement des classes non-atomiques, une autre méthode plus éfficace a été proposée basée sur la bissimulation Déf. 10(voir plus-haut).

En pratique, la manipulation directe des graphes de régions est côuteuse, c'est pour cela que des méthodes plus interessantes basées sur des zones *(:manipulation d'union convexe de régions)*, ont été proposées (voir ci-après). Il est important de noter qu'un tel graphe n'est pas utilisé pour la vérification des propriétées $TCTL$ à cause du probléme de l'explosion combinatoire du nombre d'états, pour plus de détails voir [18].

3.4.2 Graphe des Zones

Nous proposons dans cette section, le graphe des zones [37, 38](défini à l'origine pour les TA [5]), pour le calcul de l'espace d'etats des $RdPT$. Cette méthode est particuliérement efficace et rapide par rapport aux autres abstractions basées sur les classes d'etats (plus haut).

Nous donnons d'abord quelques définitions classiques.

Définition 21 (Zone) *Soit H un ensemble d'horloges. Une zone est un ensemble convexe de valuations d'horloges représentant un ensemble de contraintes de la forme : $\alpha_i - \alpha_j \prec h_{ij}, \alpha_i \prec h_{i0}, -\alpha_j \prec h_{0j}$; avec :$\alpha_i, \alpha_j \in H$; $h_{ij}, h_{i0}, h_{0j} \in \mathbb{Q} \cup \{-\infty \ \ \infty\}$ et $\prec \in \{<, \leq, =, >, \geq\}$.*

De même que pour les classes d'états de Berthomieu, une zone peut être encodée par une DBM (Difference Bound Matrices), en utilisant une horloge additionnelle α_0 toujours égale à 0. Nous associons à cette horloge une transition t_0 toujours sensibilisée mais jamais tirée.
Soit α_0 une horloge

Une zone Z représente l'ensemble des contraintes atomiques de la forme : $\alpha_i - \alpha_j \prec z_{ij}$, avec $\alpha_i, \alpha_j \in H \cup \{\alpha_0\}$, $z_{ij} \in \mathbb{Q} \cup \{\infty\}$ et $\prec \in \{<, \leq, =, >, \geq\}$, on peut écrire :

$Z = \bigcap_{\alpha_i,\alpha_j \in H \cup \{\alpha_0\}} (\alpha_i - \alpha_j \prec z_{ij})$

Le futur d'une zone noté $\overrightarrow{Z}$ est la zone obtenue en mettant Z sous forme canoniqu e puis de remplacer chaque contrainte de la forme $\alpha_i - \alpha_0 \prec z_{i0}$, où $\alpha_i \neq \alpha_0$, par : $\alpha_i - \alpha_0 < \infty$

Définition 22 (Etat symbolique) *Un état symbolique d'un RdPT est un couple (M, Z), où : M est un marquage et Z est une zone sur l'ensemble des horloges associées aux transitions sensibilisées par le marquage M.*

Informellement, un état symbolique représente un ensemble de valuations pour lesquelles un marquage est accessible .

Définition 23 (Successeur Discret) *[37, 38] Soit $s = (M, Z)$ un état symbolique. Les successeurs discrets de s obtenus par le tir de la transition t sont :*

$$post_t(s) = (M^{\uparrow}, Z^{\uparrow}) := \begin{cases} M^{\uparrow}(p) := M(p) - Pre(p,t) + Post(p,t). \\ Z^{\uparrow} = (Z \cap \{\alpha_t \geq tmin(t)\}) \cap \bigcap_{\forall t_j \in New(M^{\uparrow})} \{\alpha_j = 0\} \end{cases}$$

$post_t(s)$ est l'ensemble des états qui sont accessibles à partir de s par le tir de la transition discréte t. $Z \cap \{\alpha_t \geq tmin(t)\}$ représente l'ensemble de valuations pour lesquelles t est tirable (t est tirable si cet ensemble est non vide). $\{\alpha_j = 0\}$ est l'ensemble des horloges des transitions nouvellement sensibilisées.

Définition 24 (Successeur Temporel) *[37, 38] Soit $s = (M, Z)$ un état symbolique. Les successeurs temporels de s sont les états de l'état symbolique :*

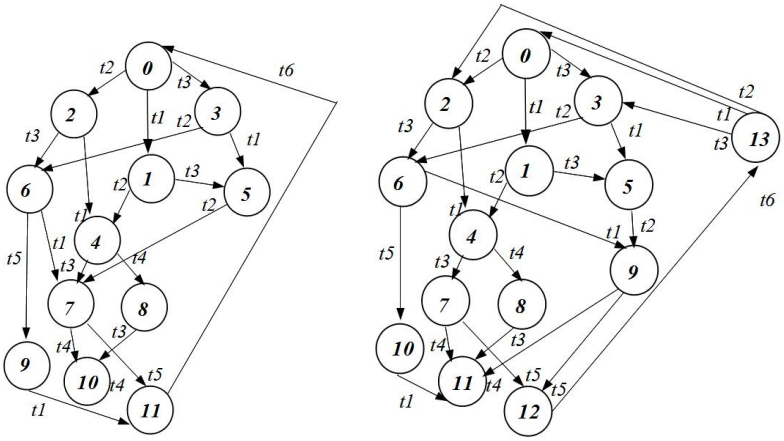
$\overrightarrow{post}(s) = (M, Z^{\uparrow})$ *avec :*
$Z^{\uparrow} = \overrightarrow{Z} \bigcap_{t \in T(M)} \{\alpha_t \leq tmax(t)\}$

$\overrightarrow{post}(M, Z)$ est l'ensemble des états qui sont accessibles à partir de s en laissant écouler du temps (jusqu'a ce qu'une horloge atteigne la borne maximale de tir.

Nous pouvons écrire : $\overrightarrow{post_t}(s) = \overrightarrow{post}(post_t(s))$.

3.4.2.1 Construction du graphe

Le principe est de calculer itérativement les états symboliques accessibles à partir de l'état symbolique initial $\overrightarrow{post}(M_0, Z_0)$, où M_0 est le marquage initial et les horloges de Z_0 sont à zéro. L'état successeur symbolique est calculé comme suit : soit $s = (M, Z)$ un état symbolique accessible, pour chaque transition t tirable dans s :



a) Avec Inclusion [38] b) Préservant : LTL [18]

FIGURE 3.16 – Graphe de zone GR_z

Définition 25 *Le graphe de zone d'un RdPT [38, 18], noté GR_z, est obtenu par :*

a) *Calcul des états accessibles par écoulement du temps :*
$(M, Z^{\uparrow}) = \overrightarrow{post}(M, Z)$ *;*

b) *Pour chaque transition t tirable dans $(M, Z^{\uparrow})$, calcul de tous les successeurs discrets $post_t(M, Z^{\uparrow})$.*

Si le réseau comporte des transitions avec une borne de tir maximale infinie, une étape supplémentaire est nécessaire consistant à appliquer une approximation assurant la convergence [38, 18]. Dans le cas contraire, aucune approximation n'est nécessaire et l'algorithme calcule uniquement et exactement les états du *RdPT* (notre cas d'étude).

3.4.2.2 illustration

Pour notre exemple (figure Fig 3.2a)), son graphe de zone avec inclusion et celui conservant les propriétées linéaires (LTL) sont représentés dans Fig. 3.16a) et 3.16b), respectivement.

L'état initial est définit par $s_0 = (M_0, Z_0)$, où, $M_0 = p_1, p_2, p_3 \mapsto 1$ et $Z_0 = \{\alpha_1 = \alpha_2 = \alpha_3 = 0\}$. Avec le marquage M_0, les transitions $t_1; t_2; t_3$ sont sensibilisées. La premiére étape étant de calculer les futurs possibles (le temps maximale) pour lequel M_0 peut exister :

En suivant la définition Déf.25, on peut construire le graphe de zone comme suit :
La zone maximale (futur possible) est $\overrightarrow{Z} = \{\alpha_1 = \alpha_2 = \alpha_3 \in [0 \ \ \infty[\}$.

a) Calcul des états accessibles par écoulement du temps :
$\overrightarrow{post}(M_0, Z_0) = (M_0, Z_0^{\uparrow})$, telque :
$Z_0^{\uparrow} = \overrightarrow{Z} \cap \{\alpha_1 \leq 3 \wedge \alpha_2 \leq 0 \wedge \alpha_3 \leq 3\}$
$Z_0^{\uparrow} = \{\alpha_1 = \alpha_2 = \alpha_3 \in [0 \ \ 0]\}$
Si on considére le tir de t_2 de la séquence précedente, alors :

b) Pour la transition t_2 tirable depuis s_0 , on calcule $s_2 = (M_2, Z_2)$ successeur discret de s_0 :

$$post_t(s_0) = (M_2, Z_2) := \left\{ \begin{array}{l} M_2 = p_1, p_3, p_5 \mapsto 1. \\ Z_2 = Z_0^{\uparrow} \cap \{\alpha_1 \geq 0 \wedge \alpha_3 \geq 0\}) = \{\alpha_1 = \alpha_3 \in [0 \ \ 0]\} \end{array} \right.$$

(pas de transitions nouvellement sensibilisées)
De la même maniére on a :

a) Calcul des états accessibles par écoulement du temps :
$\overrightarrow{post}(M_2, Z_2) = (M_2, Z_2^{\uparrow})$, telque :
$Z_2^{\uparrow} = Z_0^{\uparrow} \cap \{\alpha_1 \leq 3 \wedge \alpha_3 \leq 3\}$
$Z_2^{\uparrow} = \{\alpha_1 = \alpha_3 \in [0 \ \ \infty]\}$

b) Pour la transition t_3 tirable depuis s_2 , on calcule $s_6 = (M_6, Z_6)$ successeur discret de s_2 :

$$post_t(s_2) = (M_6, Z_6) := \left\{ \begin{array}{l} M_6 = p_1, p_5, p_6 \mapsto 1. \\ Z_6 = Z_2^{\uparrow} \cap \{\alpha_1 \geq 0 \wedge \alpha_5 = 0\} = \{\alpha_1 = \alpha_5 \in [0 \ \ 0]\} \\ \end{array} \right.$$

avec α_5 la variable qui correspond à t_5 ; nouvellement sensibilisée ;

Les états accessibles par écoulement du temps :
$Z_6^{\uparrow} = \{\alpha_1 = \alpha_5 \in [0 \ \ \infty[\}$.

On obtient à la fin, selon les critéres de convergence considérés (égalité ou inclusion des états symboliques), les graphes de zone de la figure Fig. 3.16a) préservant marquages [38] et la figure Fig. 3.16b). préservant marquages et traces (LTL) [18]. On retrouve les correspondances suivantes :

- la classe d'état initiale de [38] inclut les états : initiale et finale de [18] ;
- la classe d'état 7 de [38] inclut les états : 7 et 8 de [18].

On obtient un graphe correspondant au (SSCG) [8] précedent en prenant en considération, en plus les états futurs pour le graphe de zones (Dans la littérature certains auteurs désignent le graphe des classe fortes par le graphe des zones).

3.4.2.3 Les propriétés $TCTL$

Les auteurs [18], sont allés plus loin en proposant une méthode à la volée basée sur le graphe de zone pour la vérification de propriétés $TPN - TCTL_{\mathfrak{s}}$ (une sous-classe de propriétés $TCTL$) directement sur le $RdPT$: (natif). En effet, augmenter le $RdPT$ avec une horloge additionnelle pour capturer le temps de parcours d'une séquence de tir du $RdPt$, est équivalent à l'ajout d'une transition avec l'intervalle $[0 \ \infty[$, qui n'est jamais tirée.

Dans ce cas on se trouve avec des $RdPT$ non bornés, on a besoin de determiner une valeur (approximée (k) pour obtenir des résultats plus performants, soit C_{min}, C_{max} les bornes Min et Max de la formule à vérifier, deux cas de figures peuvent se produire (voir approximation [18]) :

– Si la formule à vérifier est bornée par une valeur définie, C_{max}, l'intervalle ci-dessus prend comme borne maximale cette dernière, $k = C_{max}$ ($[o \ \ C_{max}]$) ; dans ce cas, on peut obtenir le temps exact de parcours de la séquence.

– Si la formule à vérifier est non bornée supérieurement, mais avec une borne minimale définie, l'intervalle prend comme borne maximale cette valeur et le résultat étant un temps de parcours de la séquence plus grand que cette derniére.

Les auteurs de cette méthode prouvent la décidabilité du $TPN - TCTL_{\mathfrak{s}}$ pour les modèles bornés et encadrent sa complexité.

3.5 D'autres méthodes

Dans la littérature, alors que la méthode standard pour la vérification des propriétés temporelles quantitatives, sur un $RdPT$ consistait en l'utilisation des observateurs, en le traduisant en un probléme d'accessibilité de marquage. Plusieurs méthodes et outils efficaces permettant la vérification de propriétés exprimées avec la logique temporelle $TCTL$ ont été définis pour le cas des automates temporisés : Uppaal et Kronos.

L'une des approches pour vérifier des propriétés $TCTL$ sur des réseaux de Petri temporels consiste à établir une traduction des réseaux de Petri vers les automates temporisés. Les méthodes de traduction peuvent être classées en deux catégories :

– D'une part, les traductions structurelles (telle la traduction proposée dans [24, 25] d'un réseau de Petri temporel non nécessairement borné à dates de tir

au plus tard finies ou infinies vers un automate temporisé) ;

– Dautre part les traductions avec calcul de l'espace d'états (à l'instar de la traduction présentée dans [33, 48] qui consiste à obtenir l'espace d'états d'un réseau de Petri temporel sous la forme d'un automate temporisé).

La deuxiéme méthode est plus proche de notre objectif. En effet, l'automate correspondant à notre exemple de travail contient 16 localités différentes et 26 arcs. Par rapport à la méthode basée classe d'états préservant marquage et propriétés LTL (GR), on a 3 classes et 5 arcs en plus. Notons que l'automate est représenté ci-après en omettant les contraintes temporelles associées aux localités et aux arcs.

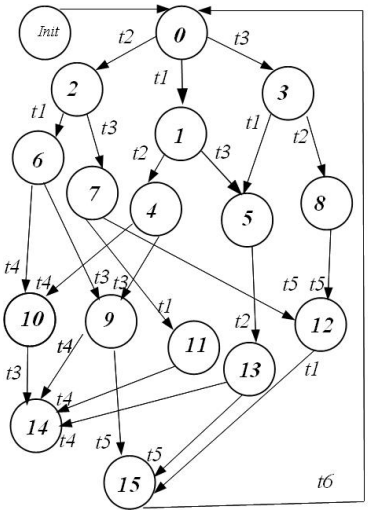


FIGURE 3.17 – Traduction du TPN précedent en un Automate [33, 48]

Nous restons exclusivement dans le cadre des $RdPT$, nous n'irons pas plus loin en ce qui concerne les traductions entre des $RdPT$ vers les AT.

3.6 Comparaison

Rappelons que les graphes notés $GR, GR_a, GR_c, GR_r, GR_z$ correspondent aux graphes : des classes d'états [9, 10], des classes atomiques [67, 8, 14], des classes en modeC [21, 20], des graphes de régions [67] et des zones [38, 18] respectivement.

Il est à noter que cette section n'est qu'une conclusion de ce qui a été présenté dans les sections précédentes, on portera certaines comparaisons jugées *intéressantes*, les autres seront intuitivement déduites par rapport à ce qui a été développé à partir des calculs précédents, nous en retenons en l'occurrence que :

Chaque graphe construit, répond à des besoins bien spécifiques préservant différentes propriétés (accessibilité, $LTL, CTL, CTL^*, TCTL$). Dans le GR, une

classe est donnée par son marquage, le domaine temporel des transitions sensibilisées et les contraintes (non-redondantes) existantes entre ces transitions dans le passé. Dans le GR_a, une classe est donnée par son marquage et une horloge pour chaque transition sensibilisee (si la transition vient d'être sensibilisée, l'horloge est à zéro, sinon elle prend la valeur précédente). Le graphe des classes atomiques résoud le problème de branchement du graphe GR, tandis que le graphe en modeC génère des chemins plus déterministes (avec notion de classes restreintes), avec en plus, les contraintes temporelles devant être vérifiées. De plus, le graphe GR_a permet de différencier les états pour lesquels il y a un conflit entre deux transitions de ceux pour lesquels une seule transitions est franchissable.

Une autre différence est dans l'utilisation des STN pour le modeC (il suffit de concatener les Nt_s du chemin étudié (voir l'exemple du chemin σ) au lieu de faire des calculs supplémentaires.

Par ailleurs, le graphe en modeC GR_c est plus proche de celui du graphe des zones Yoneda [67], avec cependant des différences. Dans le cas du modeC, quand une classe restreinte C_r est créee, la classe initiale C est conservée au lieu d'être remplacée par une classe qui est complémentaire à C_r. Le graphe GR_c ne garde pas toutes les contraintes du passé et surtout il traite les transitions non-bornées.

La notion des contraintes temporelles récemment introduite aux graphes GR et GR_z pour une vérification à la volée des propriétées $TCTL$ (une sous-classe de propriétées $TCTL$, notées : $TCTL_{TPN}$, $TPN - TCTL_s$ [39, 18], respectivement sont proposées) génére un graphe pas forcément complet, en effet, une fois la valeur de la formule déterminée, la construction s'arrête. De plus, pour ces méthodes, les $RdPT_s$ initiaux sont augmentés par l'ajout d'un $RdPT$ appelé horloge d'alarme pour GR et qui est en fait une horloge supplémentaire de $[0 \ \infty[$ pour le cas du GR_z ; elle détermine la contrainte temporelle du parcours du chemin. Ceci le différencie par rapport au GR_c, car on peut voir clairement que Le graphe de classes en modeC permet d'obtenir les contraintes temporelles qui doivent être vérifiées par chaque tir de transition dans une séquence de tir effectivement franchissable (exp : σ). Ce n'est pas le cas pour ceux basées sur une vérification à la volée.

Une autre différence apparaît dans la façon dont le passé est mémorisé. Le GR_c permet d'obtenir directement l'ensemble des contraintes temporelles d'une séquence. Bien que l'obtention des contraintes quantitatives liées à une séquence de tir ne soit pas nécessaire pour la vérification des propriétés, elle est fondamentale pour aider le concepteur à ajuster les paramétres qui permettent de vérifier ces propriétés.

Il est à noter que l'accéssibilité d'un marquage permet de répondre à la question :(Peut-on accéder à un marquage M à partir de M_0 ?), ce qui est le cas du graphe GR. Alors que l'accessibilité d'un état permet de répondre à la quastion :(Peut-on accéder au marquage M tel que t soit sensibilisée n unités de

Graphe	Accessibilité	LTL	CTL	CTL^*	TCTL
GR	[9, 10, 39, 40]	[9, 10, 39]			$(TCTL_{TPN})$ [39]
GR_s	[14]	[8]			
GR_a				[67, 8, 14]	
GR_c				[21, 20]	
GR_r			[67]		
GRz	[37, 38, 18]	[18]			$TPN - TCTL_{\mathfrak{s}}$ [18]

FIGURE 3.18 – Tableau Comparatif.

temps ?) ; le graphe GR_s est le mieu adapté pour y répondre.

Le tableau 3.18 synthétise les différences entre les graphes construits plus haut.

3.7 Conclusion

Dans ce chapitre, nous avons rappelé les principales méthodes permettant de calculer l'espace d'états des réseaux de Petri T-temporels (RdPT). Ces abstractions diffèrent sur les classes de propriétés qu'elles préservent. Les graphes des classes fournissent une représentation finie de l'espace d'états d'un RdPT borné préservant les propriétés LTL [10] ou CTL^* [67, 8, 14]. Le graphe des régions géométriques préserve les propriétés CTL et le graphe des zones préserve l'accessibilité des marquages ou les propriétés LTL en fonction du critère de convergence utilisé. Le grapheC a été proposée pour la vérification de propriétés temporelles quantitatives [21, 20]. Afin de bénéficier des outils définis sur les AT, certains auteurs [24, 25, 33, 48] ont eu l'idée de traduire un RdPT vers un AT et de verifier ses propriétées directement sur le AT obtenu. Les méthodes basées sur la notion du graphe des classes *ont l'avantage d'être exploitables aux modèles à chronométre, ceci va constituer le chapitre suivant.*

Chapitre 4

Le Calcul de l'espace d'état d'un $ITPN$

4.1 Introduction

Comme dans le cas des réseaux de Petri temporels, l'espace d'état d'un réseau de Petri à chronomètres est infini. De ce fait, son calcul nécessite de recourir à des abstractions. Cependant, étant donné que l'accessibilité et la bornitude sont indécidables pour ces modèles, même ceux bornés, il n'existe pas de garantie pour que le calcul d'abstractions exactes se termine. Le recours à des surapproximations de l'espace d'état a permis de calculer des graphes finis lorsque le calcul exact s'avère infini, préservant ainsi un sous ensemble de propriétés. Nous passons en revue dans ce chapitre, les différentes techniques pour le calcul exact et surapproximé de l'espace d'état d'un $ITPN$.

4.2 Méthodes exactes pour le calcul de l'espace d'état d'un $ITPN$

Nous passons en revue dans cette section les méthodes définies dans la littérature pour le calcul exact d'abstractions de l'espace d'états d'un $ITPN$ ou tout autre modèle étendu aux chronomètres.

4.2.1 Construction du graphe de classe d'états (SCG) d'un $ITPN$

La première méthode de calcul de l'espace d'état des $ITPN$ a été proposée par Olivier H. Roux et Didier Lime [47] sous la forme d'une adaptation de la méthode du graphe des classes d'état. Cette méthode calcule le SCG exact d'un $ITPN$ préservant principalement les propriétés linéaires du modèle (LTL).

4.2.1.1 Classe d'état

Pour un $RdPT$, la méthode du graphe de classes [10] permet de calculer un graphe symbolique préservant principalement les propriétés linéaires du modèle. De la même manière, cette construction peut être aussi appliquée à un $ITPN$. Une classe est définie par le couple (M, D) où M est le marquage commun à tout les états et D est un système d'inéquations générales (polyédriques). Plus formellement une classe d'état d'un $ITPN$ est définie comme suit [13] :

Définition 26 *Soit $ST = (\Gamma, e^0, \rightarrow)$ le système de transition associé à un $ITPN$. Une classe d'état d'un $ITPN$, notée E, est l'ensemble des états de Γ accessibles aprés le tir d'une même séquence non temporisée $S = (t_f^1, .., t_f^n)$ à partir de l'état initial e^0. La classe E est définie par (M, D), où M est le marquage accessible aprés le tir de S, et D est le domaine de tir représenté par un système d'inéquations.*
Pour $Te(M) = \{t_1, .., t_s\}$, nous avons : $D = \widehat{D} \wedge \overrightarrow{D}$

$$\overrightarrow{D} := \begin{cases} \bigwedge_{i \neq j} & (\underline{t_j} - \underline{t_i} \ \leq \ d_{ij}) \\ \bigwedge_{i \leq s} & (d_{i\bullet} \leq \ \underline{t_i} \ \leq d_{\bullet i}) \end{cases}$$

avec $(t_j, t_i) \in Te(M)^2 \quad d_{ij} \in \mathbb{Q} \cup \{\infty\}$,
$d_{\bullet i} \in \mathbb{Q}^+ \cup \{\infty\}, d_{i\bullet} \in \mathbb{Q}^+$

$$\widehat{D} := \bigwedge_{k=1..p} \quad (\alpha_{1k}\underline{t_1} + .. + \alpha_{sk}\underline{t_s} \ \leq \ d_k)$$

avec $d_k \in \mathbb{Q} \cup \{\infty\}$, $(\alpha_{1k}, .., \alpha_{sk}) \in \mathbb{Z}^s$ *et*
$\forall k, \exists (i, j), (\alpha_{ik}, \alpha_{jk}) \notin \{(0,0), (1,0), (0,1), (1,-1)\}$
où $\mathbb{Z}$ définit l'ensemble des entiers relatives.

Dans les modèles avec chronométres le systeme D n'est plus exprimable sous forme DBM (*Difference Bound Matrix*)[34]. En effet, le domaine des valuations des états appartenant à une classe ne peut être encodé seulement par des contraintes de type DBM. En fait, certaines contraintes générales (dites aussi polyédriques), non réductibles à une telle forme , sont nécessaires pour capturer l'espace d'état d'une classe. Ces contraintes formant le sous-système $\widehat{D}$, induisent une complexité supérieure, pouvant être exponentielle en nombre de variables.

4.2.1.2 Construction du graphe de classes

Le graphe de classes d'un $ITPN$ est calculé (comme pour un $RdPT$), par énumération des classes accessibles à partir de la classe initiale E^0, jusqu'à ce qu'il n y ait plus de de classes à explorer. Cependant, comme le nombre de classes accessibles peut être non borné, la terminaison de l'algorithme n'est pas garantie. Formellement, le graphe de classes exact d'un $ITPN$ est défini comme suit [47] :

Définition 27 *Le graphe de classes exact d'un $ITPN$, noté GR, est un triplet $(CE, E^0, \longmapsto)$ où :*
- CE est l'ensemble des classes accessibles dans GR ;
- $E^0 = (M^0, D^0)$ est la classe initiale telle que : $D^0 = \{ \quad \forall t_i \in Te(M^0), \quad tmin(t_i) \leq \underline{t_i} \leq tmax(t_i)$
- $\longmapsto$ est la relation de transition entre classes definie sur $CE \times T \times CE$, telle que
$((M, D), t_f, (M^{\uparrow}, D^{\uparrow})) \in \longmapsto$, ssi :

a) *t_f est activée et le système D augmenté par les contraintes de tir de t_f que nous écrivons $D_a = D \wedge (\forall t \in Ta(M), \quad \underline{t_f} \leq \underline{t})$ est consistant.*

b) *$\forall p \in P$, $M^{\uparrow}(p) := M(p) - Pre(p, t_f) + Post(p, t_f)$.*

c) *Le système $D^{\uparrow}$ est calculé à partir de D, en appliquant l'algorithme suivant :*

1. *Remplacer dans D_a, chaque variable $\underline{t}$ (relative à une transition activée pour M), par : $\underline{t} := \underline{t_f} + \underline{t'}$, exprimant ainsi la progression du temps.*
2. *Eliminer par subtitution dans le système obtenu, la variable $\underline{t_f}$ ainsi que toutes les variables relatives aux transitions désensibilisées suite au tir de t_f;*
3. *Rajouter au système obtenu, les contraintes des transitions nouvellement sensibilisées dans $M^{\uparrow}$: $\forall t_i \in New(M^{\uparrow}), \quad tmin(t_i) \leq \underline{t_i} \leq tmax(t_i)$*

La définition précédente montre, comme dans le cas des $RdPT$, comment est construit le graphe de classes d'un $ITPN$, nous voyons que pour les $ITPN$, les changements de variables ne sont effectués que pour les variables correspondantes à des transitions *actives*. Ce sont en effet les seules pour lesquelles le temps

s'écoule ; le reste est inchangé. La complexité de l'étape 2, dépend de la forme du système D ; en fait si D contient des contraintes polyédriques non réductibles en DBM, la complexité de l'algorithme est exponentielle, alors qu'elle est polynomiale dans le cas contraire. Il est à noter que le système initial D_0 est en forme DBM et que les contraintes polyédriques apparaissent dans les systèmes des classes accessibles en cas de persistance des transitions inhibées [19]. Afin d'éliminer toutes les contraintes redondantes, les nouveaux systèmes sont calculés dans leur forme *minimale (canonique)*. D'ailleurs, il est prouvé, comme pour les systèmes DBM [10], que la forme minimale d'un système polyédrique est unique [7]. Cette propriété est importante car elle permet de détecter les classes équivalentes, en testant l'égalité de leur forme minimale. Ce test permet de mettre fin à l'exploration d'un comportement cyclique dans le graphe.

4.2.1.3 Les classes équivalentes

Le graphe de classes d'un $ITPN$ est construit de manière incrémentale par énumération de toutes les classes accessibles à partir de la classe initiale, telles que les classes équivalentes sont regroupées dans un même noeud du graphe. La relation d'équivalence peut être l'égalité : deux classes (M, D) et (M, D'), données dans leur forme minimale sont égales si $D = D'$, ou bien l'inclusion ; en d'autres terme, si $\rceil D \lceil$ désigne l'ensemble des solutions du système D, alors nous avons : $\rceil D \lceil \subseteq \rceil D' \lceil$. Il est à noter que l'égalité permet de préserver principalement le langage du modèle, alors que l'inclusion préserve les marquages accessibles.

4.2.1.4 Problématique

L'algorithme donné précédemment s'applique aussi à un $RdPT$ avec la particularité que les systèmes D se réduisent toujours à la forme DBM. De plus, il est prouvé que le nombre de systèmes DBM que l'on peut calculer dans le graphe de classes d'un $RdPT$ est fini [9]. Cette propriété est importante car elle implique que le graphe est nécessairement fini si le nombre de marquages accessibles est borné. Malheureusement, ces propriétés ne sont plus vraies en présence des chronomètres. Par conséquent, le graphe de classes d'un $ITPN$ dont le nombre de marquages accessibles est borné, peut être infini car le nombre de systèmes polyédriques différents que l'algorithme génère peut être infini.

Par ailleurs, le facteur coût pénalise fortement la construction exacte. En effet, un polyèdre nécessite des structures plus complexes pour être représenté en mémoire qu'une DBM (matrice). De plus le coût de manipulation d'un tel système est exponentiel au nombre de variables présentes. Un graphe de quelques centaines de milliers de classes peut ainsi nécessiter plusieurs heures de calcul, voire induire un crash mémoire.

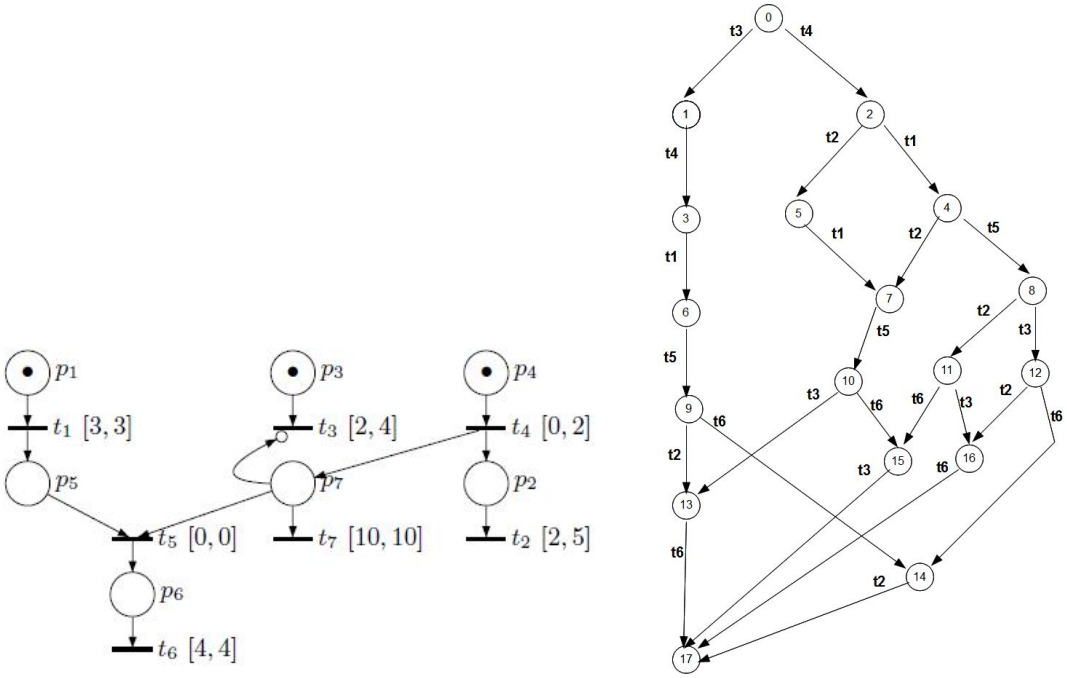


FIGURE 4.1 – Exemple 1 : a) $ITPN$ chap 2, b) Son graphe exact

Afin de réduire ce coût, il a été remarqué que le sous-système $\widehat{D}$ est souvent redondant, voire sans incidence sur la tirabilité des transitions. Celà a permis d'établir un compromis entre les deux représentations, et une approche hybride a été proposée par [51]. Cette dernière exploite une condition suffisante permettant de détecter les cas où le sous-système $\widehat{D}$ devient redondant dans D. A partir de là, la combinaison des deux représentations permet de construire le graphe de classes exact, en réduisant sensiblement les temps de calcul et l'espace mémoire alloué comparativement à l'approche basée sur une représentation exclusivement polyédrique [47].

4.2.1.5 Exemple

Considérons le réseau de Petri à arcs inhibiteur de la figure FIG 4.1a) du chapitre 2. Le graphe de classe exact généré aprés application de l'algorithme de la définition 27 est représenté sur la figure FIG 4.1b). Ce dernier contient 18 classes et 25 arcs. Il es est à noter que la méthode mixte de [51] donne le même résultat. (voir le détail des classes en ANNEXES)

4.3 Méthodes approchées pour le calcul de l'espace d'état d'un $ITPN$

4.3.1 Sur-approximation par DBM du SCG

Pour contourner les problématiques liées à la manipulation des systèmes polyédriques, la surapproximation par DBM a été proposée comme solution alternative pour l'analyse des systèmes préemptifs. Cette approche consiste en la suppression du sous-système $\widehat{D}$ lorsqu'il est généré, ne laissant que le sous-système $\overrightarrow{D}$ pour représenter ainsi une surapproximation de l'espace de D. Cette solution permet de construire un graphe moins riche et souvent grossier, mais néanmoins de coût sensiblement inférieur en termes de temps de calcul et d'occupation mémoire. De plus, du fait des propriétés des DBM énoncées précédemment, cette surapproximation peut produire un graphe fini, alors que le graphe exact est infini.

Pour plus de compréhension, nous appliquons la méthode du graphe de classes à l'exemple de la figure FIG 4.1a) du chapitre 2. Soit la classe $E' = (M', D')$, la classe accéssible dans le graphe de classes aprés le tir de la séquence $S = (t_4, t_1, t_5)$ à partir de la classe initiale $E^0 = (M^0, D^0)$.

$$E^0 = \left(\begin{array}{l} M^0 : p_1, p_3, p_4 \rightarrow 1 \\ D^0 : \left\{ \begin{array}{l} 3 \leq \underline{t_1} \leq 3 \\ 2 \leq \underline{t_3} \leq 4 \\ 0 \leq \underline{t_4} \leq 2 \end{array} \right. \end{array} \right) \qquad E' = \left(\begin{array}{l} M' : p_1, p_4, p_6 \rightarrow 1 \\ D' : \left\{ \begin{array}{ll} 0 \leq \underline{t_2} \leq 4 & 4 \leq \underline{t_6} \leq 4 \\ 0 \leq \underline{t_3} \leq 4 & 1 \leq \underline{t_2} + \underline{t_3} \leq 6 \end{array} \right. \end{array} \right)$$

Nous pouvons facilement remarquer que la transition t_6 est non tirable à partir de E' puisque t_2 ou t_3 doivent être tirées avant. Plus concrètement, le tir de t_6 nécessite que le système $D' \wedge (\underline{t_6} \leq \underline{t_3}) \wedge (\underline{t_6} \leq \underline{t_2})$ admette au moins une solution ; nous devons vérifier que $(\underline{t_2} = \underline{t_3} = \underline{t_6} = 4) \wedge (\underline{t_2} + \underline{t_3} \leq 6)$. Comme cette dernière inéquation est non satisfaite, alors t_6 ne peut être tirée. Le système D' a une forme polyédrique non réductible à une forme DBM. Son approximation consiste en la suppression des contraintes polyédriques $1 \leq \underline{t_2} + \underline{t_3} \leq 6$; produisant ainsi un système DBM compact dont la manipulation requiert une complexité polynomiale en nombre de variables. Cependant, par cette suppression, t_6 devient tirable puisque $\underline{t_6} = 4$ est satisfaite. Par conséquent, le système $\widetilde{D'} = \overrightarrow{D'}$ dénote une surapproximation du système D'. En d'autres termes nous rajoutons de nouveaux états dans la classe E' qui, au demeurant, ne sont pas accessibles. Neanmoins, cette construction permet de préserver un sous-ensemble de propriétés, pouvant être suffisant pour la vérification du système modélisé.

Différents algorithmes sont utilisés pour le calcul d'une approximation DBM de la classe E. Citons [60, 19] où chaque classe du graphe est calculée dans sa forme minimale (canonique). La démarche de telles méthodes est la suivante : le

calcul du polyèdre, élimination des contraintes non DBM tout en normalisant celles restantes, enfin calculer le système final dans sa forme minimale.

Toutefois, il a été montré dans [17] que le maintien de toutes les contraintes DBM, même celles redondantes, pour la représentation du système DBM associé à une classe d'un $RdPT$, permet de définir un algorithme de calcul d'une classe de complexité $O(m^2)$. Dans le même esprit, dans [3] l'auteur a adapté cette représentation et proposé un algorithme pour la construction d'une surapproximation par DBM qu'il prouve être la plus fine possible. Plus concrètement, l'auteur montre qu'en représentant toutes les contraintes DBM, on arrive à déduire un algorithme qui permet de calculer directement le système DBM dans sa forme normale sans passer par le polyèdre intermédiaire. En évitant la manipulation du polyèdre et les phases de normalisation et de minimisation, on optimise de manière significative l'implémentation du calcul d'une classe approchée, en évitant les défauts d'implémentation constatés dans les autres approches.

Nous donnons d'abord quelques définitions nécessaires pour la description de la méthode :

4.3.1.1 Classe d'état approchée par DBM

La classe $\widetilde{E} = (M, \widetilde{D})$ est une surapproximation de la classe $E = (M, D)$ accéssible dans GR si l'ensemble des états de E sont inclus dans $\widetilde{E}$; et nous avons : $\rceil D \lceil \subseteq \left\rceil \widetilde{D} \right\lceil$. Par conséquent, en subtituant $\widetilde{E}$ pour E dans le graphe GR, il en résulte que la classe $\widetilde{E}$ peut dériver des séquences additives non franchissables dans GR à partir de E. Nous obtenons ainsi un graphe approché constitué de classes approchés de GR, que nous pouvons construire comme défini ci-après. Plus formellement, la classe approchée par DBM d'un $ITPN$ est définie comme suit :

Définition 28 *Une classe approchée par DBM d'un $ITPN$, notée $\widetilde{E}$, est le couple $(M, \widetilde{D})$ tel que : M est le marquage et $\widetilde{D}$ est le système complet de contraintes DBM normalisées, impliquant toutes les variables des transitions sensibilisées pour M.*

$$\widetilde{D} = \left\{ \begin{array}{l} \bigwedge_{\forall (t_i,t_j) \in Te(M)^2} \quad (\underline{t_j} - \underline{t_i} \ \leq \ \widetilde{d_{ij}}) \\ \bigwedge_{\forall t_i \in Te(M)} \quad (\widetilde{d_{i\bullet}} \leq \ \underline{t_i} \ \leq \widetilde{d_{\bullet i}}) \end{array} \right.$$

avec $(t_j \neq t_i)$, $\widetilde{d_{ij}} \in \mathbb{Q} \cup \{\infty\}$, $\widetilde{d_{\bullet i}} \in \mathbb{Q}^+ \cup \{\infty\}$, $\widetilde{d_{i\bullet}} \in \mathbb{Q}^+$:

tel que chaque inéquation est sous sa forme normale :
$\forall x, y, z \in Te(M)$, $\left(\widetilde{d_{xy}} \leq \widetilde{d_{xz}} + \widetilde{d_{zy}}\right) \wedge \left(\widetilde{d_{xy}} \leq \widetilde{d_{\bullet y}} - \widetilde{d_{x\bullet}}\right)$.

Le domaine d'une classe approchée d'un $ITPN$ est induit par le système de contraintes $\widetilde{D}$ 4.3.1.2, est donné sous sa forme normale unique.

4.3.1.2 Représentation matricielle complète d'un système DBM

Soit $\widetilde{D}$ un système complet de contraintes DBM normalisées , $\widetilde{D}$ peut être encodé par une matrice carrée où chaque ligne, et colonne correspondante, sont indexées par un élément de l'ensemble $Te(M) \cup \{\bullet\}$. Concrétement, nous avons : $\forall (t_i, t_j) \in Te(M)^2 \wedge (t_i \neq t_j)$,

$\widetilde{D}[\bullet, t_i] := \widetilde{d_{\bullet i}}$; $\widetilde{D}[t_i, \bullet] := -\widetilde{d_{i\bullet}}$; $\widetilde{D}[t_i, t_j] := \widetilde{d_{ij}}$; $\widetilde{D}[t_i, t_i] := 0$; $\widetilde{D}[\bullet, \bullet] := 0$.

Ces notations matricielles sont utilisées pour dénoter les coefficients du système $\widetilde{D}$.

Par exemple, la matrice présentée dans le tableau 4.1 encode le système D^0=$\widetilde{D^0}$ associé à la classe initiale du graphe exact de $ITPN$ de la figure FIG. 2.5(chapitre.2). Nous notons que la classe approchée $\widetilde{E_0} = (M_0, \widetilde{D_0})$ est en forme DBM, et représente une approximation exacte de la classe initiale $E_0 = (M_0, D_0)$ du graphe GR. Par contre, la représentation minimale du système $\widetilde{D_0}$ est donnée par : $(t_1 = 3) \wedge (2 \leq t_3 \leq 4) \wedge (0 \leq t_4 \leq 2)$.

$\widetilde{D^0}$	$\bullet$	t_1	t_3	t_4
$\bullet$	0	3	4	2
t_1	-3	0	1	-1
t_3	-2	1	0	0
t_4	0	3	4	0

TABLE 4.1 – La représentation matricielle du système $\widetilde{D^0}$

4.3.1.3 Construction du graphe de classes approché par DBM [3]

Définition 29 *Le graphe de classes approché par DBM d'un $ITPN$, noté $\widetilde{GR}$, est un triplet $(\widetilde{CE}, \widetilde{E^0}, \rightsquigarrow)$, tel que :*

- *$\widetilde{CE}$ est l'ensemble des classes approchées accessibles dans $\widetilde{GR}$;*
- *$\widetilde{E^0} = (M^0, \widetilde{D^0}) \in \widetilde{CE}$ est la classe initiale, telle que :*

$$\widetilde{D^0} := \begin{cases} \forall t_i \in Te(M^0), & tmin(t_i) \leq \underline{t_i} \leq tmax(t_i) \\ \forall t_i \neq t_j \in Te(M^0), & \underline{t_j} - \underline{t_i} \leq tmax(t_j) - tmin(t_i) \end{cases}$$

- *$\rightsquigarrow$ est une relation de transition entre classes approchées définie sur $\widetilde{CE} \times T \times \widetilde{CE}$, telle que$((M, \widetilde{D}), t_f, (M^{\uparrow}, \widetilde{D^{\uparrow}})) \in \rightsquigarrow$, ssi :*

- *$(t_f \in Ta(M)) \wedge (\widetilde{\beta}[t_f] \geq 0)$ tel que : $\forall x \in Te(M) \cup \{\bullet\}$,* $\widetilde{\beta}[x] = \underset{\forall t \in Ta(M)}{MIN} \left\{ \widetilde{D}[x, t] \right\}$.
- *$\forall p \in P$, $M^{\uparrow}(p) := M(p) - Pre(p, t_f) + Post(p, t_f)$.*
- *Les coéfficients des contraintes DBM du système $\widetilde{D^{\uparrow}}$ sont calculés à partir de ceux de $\widetilde{D}$ en appliquant l'algorithme suivant :*

$\forall t \in Te(M^{\uparrow})$

$\widetilde{D^{\uparrow}}[t,t] := 0; \qquad \widetilde{D^{\uparrow}}[\bullet,\bullet] := 0;$

Si t est persistante

Si $t \in Ti(M)$ *(t est inhibée par M)*

$$\widetilde{D^{\uparrow}}[t,\bullet] := MIN\begin{pmatrix} \widetilde{D}[t,\bullet] \\ \widetilde{D}[t_f,\bullet] + \widetilde{\beta}[t] \end{pmatrix} \quad \widetilde{D^{\uparrow}}[\bullet,t] := MIN\begin{pmatrix} \widetilde{D}[\bullet,t] \\ \widetilde{D}[t_f,t] + \widetilde{\beta}[\bullet] \end{pmatrix}$$

Si $t \notin Ti(M)$ *(t est non inhibée par M)*

$\widetilde{D^{\uparrow}}[\bullet,t] := \widetilde{D}[t_f,t]$; $\qquad \widetilde{D^{\uparrow}}[t,\bullet] := \widetilde{\beta}[t].$

Si t est nouvellement sensibilisée.

$\widetilde{D^{\uparrow}}[\bullet,t] := tmax(t)$; $\qquad D^{\uparrow}[t,\bullet] := -tmin(t).$

$\forall (t_1,t_2) \in (Te(M^{\uparrow}))^2 \wedge (t_1 \neq t_2)$

Si t_1 *ou* t_2 *sont nouvellement sensibilisées.*

$\widetilde{D^{\uparrow}}[t_1,t_2] := \widetilde{D^{\uparrow}}[\bullet,t_2] + \widetilde{D^{\uparrow}}[t_1,\bullet].$

Si t_1 *and* t_2 *sont persistantes.*

Si $(t_1,t_2) \notin (Ti(M))^2$ *(t_1 and t_2 sont inhibées pour M)*

$\widetilde{D^{\uparrow}}[t_1,t_2] := MIN(\widetilde{D}[t_1,t_2], \quad \widetilde{D^{\uparrow}}[\bullet,t_2] + \widetilde{D^{\uparrow}}[t_1,\bullet]).$

Si $(t_1,t_2) \in (Ti(M))^2$ *(t_1t_2 sont inhibées par M)*

$\widetilde{D^{\uparrow}}[t_1,t_2] := MIN(\widetilde{D}[t_1,t_2], \quad \widetilde{D^{\uparrow}}[\bullet,t_2] + \widetilde{D^{\uparrow}}[t_1,\bullet]).$

Si $(t_1 \in Ti(M)) \wedge (t_2 \notin Ti(M))$ *(Seule t_1 est inhibée par M).*

$\widetilde{D^{\uparrow}}[t_1,t_2] := MIN(\widetilde{D}[t_1,t_2] + \widetilde{D}[t_f,\bullet], \quad \widetilde{D^{\uparrow}}[\bullet,t_2] + \widetilde{D^{\uparrow}}[t_1,\bullet]).$

Si $(t_1 \notin Ti(M)) \wedge (t_2 \in Ti(M))$ *(Seule t_2 est inhibée par M)*

$\widetilde{D^{\uparrow}}[t_1,t_2] := MIN(\widetilde{D}[t_1,t_2] + \widetilde{\beta}[\bullet], \quad \widetilde{D^{\uparrow}}[\bullet,t_2] + \widetilde{D^{\uparrow}}[t_1,\bullet]).$

Si t est une transition activée, alors $\widetilde{\beta}[t]$ désigne la distance de temps minimale qui sépare son instant de tir potentiel de n'importe quelle autre transition franchissable. Par ailleurs, $\widetilde{\beta}[\bullet]$ désigne le temps de séjour maximal dans la classe $\widetilde{E}$. Par conséquent, si $\widetilde{\beta}[t_f] < 0$ alors la transition activée t_f est non tirable à partir de $\widetilde{E}$ et de E. En d'autres termes, il n'existe pas d'état accessible dans $\widetilde{E}$ tel que la valuation du chronomètre de t_f ait surpassé la borne minimale $tmin(t_f)$.

L'algorithme précédent permet de construire la plus fine surapproximation par *DBM* en utilisant l'égalité comme condition suffisante pour la relation d'équivalence.

4.3.2 Exemple

Le graphe approché de notre exemple est representé sur la figure 4.2 avec 21 classes et 31 arcs.

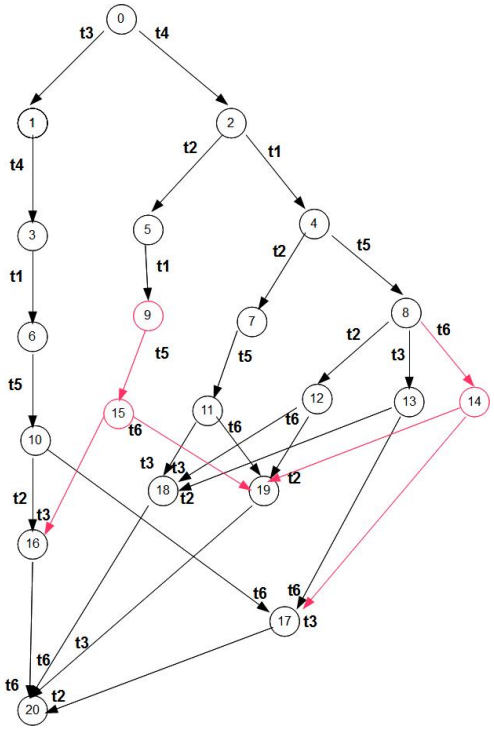


FIGURE 4.2 – Le graphe de classes approché par DBM de l'$ITPN$ précédent

Les classes en rouges sont celles rajoutées par cette surapproximation, en la comparant avec la méthode exact, nous avons :

– la classes 14 générée aprés tir de t_6 (alors que t_6 été non tirable dans le graphe exact) ;
– les classes 7 et 9 correspondent à la classe 7 du graphe exact ;
– Les classes 11 et 15 correspondent à la classe 11 du graphe exact.

On peut remarquer qu'on a 3 classes et 6 arcs en plus (voir détail des classes en ANNEXES).

4.3.3 Contraction du graphe de classes approché par DBM

Nous avons présenté dans la section précédente un algorithme calculant la plus fine surapproximation par DBM du graphe des classes d'un $ITPN$ [3]. Le même auteur a montré dans [1] que les formules de calcul de l'intervalle de tir $[-\widetilde{D}[t,\bullet], \widetilde{D}[\bullet,t]]$ d'une transition inhibée persistante pouvaient être relâchées dans la majorité des cas sans qu' il y ait une grande incidence sur l'approximation calculée. Ce nouvel algorithme permet ainsi de définir une relation d'équivalence moins restrictive que l'égalité compactant ainsi davantage le graphe approché et

réduisant son coût de calcul. Formellement, cette construction est définie comme suit :

Définition 30 *Le graphe approché par DBM contracté d'un ITPN, noté* $\widetilde{GRC}$*, est le tuple*
$(\widetilde{CEC}, \widetilde{E_c^0}, \hookrightarrow)$*, tel que :*

- $\widetilde{CEC}$ *est l'ensemble des classes accessible dans* $\widetilde{GRC}$;
- $\widetilde{E_c^0} = (M^0, \widetilde{D_c^0}) \in \widetilde{CEC}$ *est la classe initiale telle que* $\widetilde{D_c^0} = \widetilde{D^0} = D^0$.
- $\hookrightarrow$ *est la relation de transition entre classes definie sur* $\widetilde{CEC} \times T \times \widetilde{CEC}$,

telle que $((M, \widetilde{D_c}), t_f, (M^\uparrow, \widetilde{D_c^\uparrow})) \in \rightsquigarrow$*, ssi :*

- $(t_f \in Ta(M)) \wedge (\widetilde{\beta}_c[t_f] \geq 0)$ *telle que :* $\forall x \in Te(M) \cup \{\bullet\}, \widetilde{\beta}_c[x] = \underset{\forall t \in Ta(M)}{MIN} \left\{ \widetilde{D_c}[x,t] \right\}$.
- $\forall p \in P,\ M^\uparrow(p) := M(p) - Pre(p, t_f) + Post(p, t_f)$.
- *Les coéfficients du système* $\widetilde{D_c^\uparrow}$ *sont calculés à partir de ceux de* $\widetilde{D}_c$ *en appliquant l'algorithme suivant :*

  $\forall t \in Te(M^\uparrow)$

  $\widetilde{D_c^\uparrow}[t,t] := 0; \qquad \widetilde{D_c^\uparrow}[\bullet,\bullet] := 0.$

  Si t est persistante

  If $t \in Ti(M)$ $\quad \widetilde{D_c^\uparrow}[t,\bullet] := \widetilde{D_c}[t,\bullet]; \quad \widetilde{D_c^\uparrow}[\bullet,t] := \widetilde{D_c}[\bullet,t].$

  If $t \notin Ti(M)$ $\quad \widetilde{D_c^\uparrow}[\bullet,t] := \widetilde{D_c}[t_f,t]$; $\quad \widetilde{D_c^\uparrow}[t,\bullet] := \widetilde{\beta}_c[t].$

  Si t est nouvellement sensibilisée.

  $\widetilde{D_c^\uparrow}[\bullet,t] := tmax(t)$; $\qquad \widetilde{D_c^\uparrow}[t,\bullet] := -tmin(t).$

  $\forall (t_1,t_2) \in (Te(M^\uparrow))^2 \wedge (t_1 \neq t_2)$

  Si t_1 *ou* t_2 *sont nouvellement sensibilisées.* $\quad \widetilde{D_c^\uparrow}[t_1,t_2] := \widetilde{D_c^\uparrow}[\bullet,t_2] + \widetilde{D_c^\uparrow}[t_1,\bullet]$.

  Si t_1 *et* t_2 *sont persistantes.*

  Si $(t_1,t_2) \notin (Ti(M))^2 \vee (t_1,t_2) \in (Ti(M))^2$

  $\widetilde{D_c^\uparrow}[t_1,t_2] := MIN(\widetilde{D_c}[t_1,t_2], \quad \widetilde{D_c^\uparrow}[\bullet,t_2] + \widetilde{D_c^\uparrow}[t_1,\bullet]).$

  Si $(t_1,t_2) \notin (Ti(M))^2 \wedge (t_1 \in Ti(M)) \vee (t_2 \in Ti(M))$

  $\widetilde{D_c^\uparrow}[t_1,t_2] := \widetilde{D_c^\uparrow}[\bullet,t_2] + \widetilde{D_c^\uparrow}[t_1,\bullet].$

La construction du graphe $\widetilde{GRC}$, s'avère moins coûteuse que celle de $\widetilde{GR}$. Afin de réduire davantage ce coût et contracter le graphe, l'auteur a proposé une relation d'équivalence moins restrictive que l'égalité qu'il prouve être une bisimulation. Il a été démontré que certaines contraintes temporelles relatives à un couple de transitions en conflit, ou conjointement inhibées, voire activées pouvaient être ignorées lors du test d'équivalence sous certaines conditions ; ces dernières n'influençant pas le processus d'énumération. Plus formellement, cette

relation d'équivalence est définie plus bas. Mais avant, nous introduisons quelques notations :

– Une transition t_j est dite *inhibitrice pour* t_i si $\exists p \in P, \quad 0 < IH(p,t_i) \leq B(p,t_j)$. Ceci signifie que si la transition t_j est sensibilisée pour un marquage, alors t_i ne peut être activée pour ce marquage. On note par $Inhib$ la relation définie sur T^2, telle que $(t_i,t_j) \in Inhib$, si t_j est *inhibitrice pour* t_i. Il est à noter que la relation $Inhib$ est non symmétrique ; cependant si (t_i,t_j) et $(t_j,t_i) \in Inhib$, cela signifie que t_i et t_j sont toujours inhibées lorsqu'elles sont sensibilisées.

– On note AT l'ensemble des transitions de T qui ne sont connectées à aucun arc inhibiteur : $t \in AT$, si $\nexists p \in P, IH(p,t) \neq 0$.

– Deux transitions t_i et t_j sont dites *jumelles*, si $\forall p \in P, \quad IH(p,t_i) = IH(p,t_j)$. Celà signifie que deux transitions jumelles sont soit inhibées ou activées ensemble lorsqu'elles sont senbilisées pour un même marquage. On dénote par $Twin$ la relation définie sur T^2, telle que $(t_i,t_j) \in Twin$, si t_i et t_j sont *jumelles*.

Définition 31 *Soit $\simeq$ la relation définie sur le graphe $\widetilde{GRC}$, par $((M,\widetilde{D_c}),(M',\widetilde{D'_c})) \in \simeq$ ssi :*

$(i) \quad M = M'$

$(ii) \quad \forall t \in Ti(M) \quad \widetilde{D_c}[\bullet,t] = \widetilde{D'_c}[\bullet,t], \quad \widetilde{D_c}[t,\bullet] = \widetilde{D'_c}[t,\bullet]$

$(iii) \quad \forall (t,t') \in Twin \cap Conf(M)$

$$\begin{cases} sg(\widetilde{D_c}[t,t']) = sg(\widetilde{D'_c}[t,t']) \\ \widetilde{D_c}[t,t'] = \widetilde{D'_c}[t,t'] \qquad\qquad si\ sg(\widetilde{D_c}[t,t']) = <_0 \end{cases}$$

.

(iv) $\forall (t,t') \in Te(M)^2 - (Twin \cap Conf(M))$ *tel que* $(t',t),(t,t') \notin Inhib$,
$\widetilde{D_c}[t,t'] = \widetilde{D'_c}[t,t']$.

Où $sg(v)$ est la fonction qui rend le signe de la valeur de v, $sg : \mathbb{Q} \cup \{\infty\} \longrightarrow \{\geq_0, <_0\}$ tel que $\geq_0$ (resp, $<_0$), dénote "positive ou nulle" (resp, strictement négative).

Concrètement, deux classes $(M,\widetilde{D_c})$ et $(M',\widetilde{D'_c})$ sont dans la relation $\simeq$, ssi : (i) Ils ont le même marquage ; (ii) seuls les temps résiduels maximum et minimum des transitions inhibées doivent être égaux dans les deux classes ; (iii) pour chaque couple de transition jumelles en conflit pour M, la distance de franchissement entre les deux transitions doit être de même signe dans les deux classes, et leurs valeurs doivent être égales que lorsque elle sont négatives ; (iv) pour les couples de transitions qui ne sont pas dans la relation $Inhib$, leur distance de franchissement doit être la même dans les deux classes.

Le graphe $\widetilde{GRC}$ construit par la relation $\simeq$ préserve marquages et les séquences de tirs, alors qu' il est souvent plus compact que $\widetilde{GR}$. La construction de $\widetilde{GRC}$

serait mieux appropriée que $\widetilde{GR}$ lorsque le nombre de transitions inhibitrices et en conflit est important.

4.3.3.1 Exemple

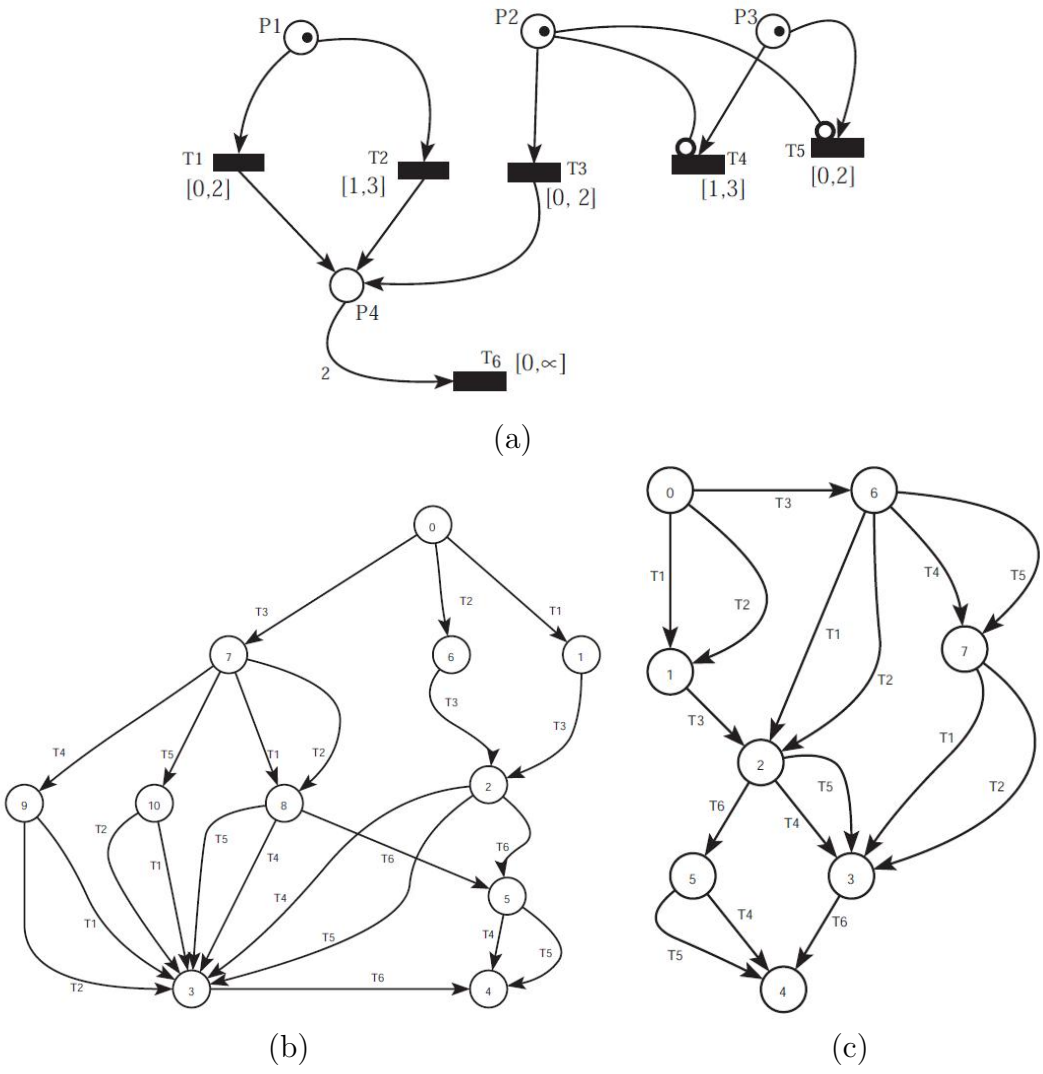


FIGURE 4.3 – Un $ITPN$ avec deux transitions jumelles inhibées, et son graphe d'accessibilité

La contraction précédente appliquée à l'exemple de la FIG 4.4 produit le même graphe que l'approximation par DBM. Ceci est du que l'$ITPN$ considéré n'est pas représentatif de ces cas de transitions. Pour illustrer cette contraction, nous

considérons l'$ITPN$ de la figure FIG 4.3a) peu plus loin, où nous avons $Inhib = \{(t_4, t_3), (t_5, t_3)\}$ et par conséquent les distances $\widetilde{D_c}[t_4, t_3]$, $\widetilde{D_c}[t_3, t_4], \widetilde{D_c}[t_3, t_5]$ et $\widetilde{D_c}[t_5, t_3]$ seront ignorées pendant le calcul de n'importe quelle classe du graphe. De plus, nous avons $AT = \{t_1, t_2, t_3, t_6\}$ et $Twin = AT^2 \cup \{(t_4, t_5), (t_5, t_4)\}$. Parmi les éléments de $Twin$, seules les transitions t_1 et t_2, d'une part, et t_4 et t_5, d'autre part , sont en conflit pour le marquage initiale. De là, les distances $\widetilde{D_c^0}[t_1, t_2]$, $\widetilde{D_c^0}[t_2, t_1], \widetilde{D_c^0}[t_4, t_5]$ et $\widetilde{D_c^0}[t_5, t_4]$ sont données positives dans $\widetilde{D_c^0}$, donc nous n'avons pas besoin de re-calculer ces valeurs ni de les comparer pour le test d'équivalence aussi longtemps que les transitions concernées restent pérsistantes.

La construction exacte GR, la sur-approximation par DBM $\widetilde{GR}$ et l'abstraction $\widetilde{GRC}$ produisent tous le même graphe de la figure FIG 4.3b). L'application de la nouvelle relation d'équivalence permet de contracter davantage le graphe $\widetilde{GRC}$ comme montré sur la figure FIG 4.3c), où le graphe est réduit tout en restant bisimilaire au graphe exact. Il rassemble des classes qui dérivent les mêmes séquences de tir [1]. Par exemple, tirer t_1 (resp, t_2), dans $\widetilde{GRC}$ à partir de la classes initiale $\widetilde{E_c^0}$ mène à la classe [2]

$\widetilde{E_c^1}$ (resp, $\widetilde{E_c^6}$). Dans ce cas, les distances $\widetilde{D_c}[\bullet, t_3], \widetilde{D_c}[t_4, t_3]$ et $\widetilde{D_c}[t_5, t_3]$ empéchant l'égalité, sont ignorées lorsque t_3 est inhibitrice de t_4 et t_5. Plus encore, les classes $\widetilde{E_c^2}$ et $\widetilde{E_c^8}$ deviennent équivalentes lorsque la valeur du temps résiduel minimum de t_4 est ignoré. Finalement, les classes $\widetilde{E_c^{10}}$ et $\widetilde{E_c^9}$ peuvent être rassemblées car la distance $\widetilde{D_c}[t_1, t_2]$ est positive et peut être igorée ; t_1 et t_2 sont deux transitions jumelles en conflit.

On obtient un graphe plus compacte de 8 classes et 16 arcs, alors que les autres constructions produisent un graphe de 11 classes et 22 arcs.

$\widetilde{E_c^0} =$

$M^0 : p_1, p_2, p_3 \rightarrow 1$

$\widetilde{D_c^0}$	$\bullet$	t_1	t_2	t_3	t_4	t_5
$\bullet$	0	2	3	2	3	2
t_1	0	0	3	2	3	2
t_2	-1	1	0	1	2	1
t_3	0	2	3	0	3	2
t_4	-1	2	2	1	0	1
t_5	0	2	3	2	3	0

1. Ces classes ne sont pas égales, mais sont bissimilaires.
2. La classe $\widetilde{E_c^i}$ correspond au noeud (i) dans le graphe.

$\widetilde{E_c^1}=$ $M^1 : p_2, p_3, p_4 \to 1$

$\widetilde{D_c^1}$	$\bullet$	t_3	t_4	t_5
$\bullet$	0	2	3	2
t_3	0	0	3	0
t_4	1	1	0	1
t_5	0	2	3	0

$\widetilde{E_c^6}=$ $M^6 : p_2, p_3, p_4 \to 1$

$\widetilde{D_c^6}$	$\bullet$	t_3	t_4	t_5
$\bullet$	0	1	3	2
t_3	0	0	3	0
t_4	1	0	0	1
t_5	0	1	3	0

$\widetilde{E_c^2}=$ $M^2 : p_4 \to 1; p_4 \to 2$

$\widetilde{D_c^2}$	$\bullet$	t_4	t_5	t_6
$\bullet$	0	3	2	∞
t_4	-1	0	1	∞
t_5	0	3	0	∞
t_6	0	3	2	0

$\widetilde{E_c^8}=$ $M^8 : p_4 \to 1; p_4 \to 2$

$\widetilde{D_c^8}$	$\bullet$	t_4	t_5	t_6
$\bullet$	0	3	2	∞
t_4	0	0	1	∞
t_5	0	3	0	∞
t_6	0	3	2	0

$\widetilde{E_c^9}=$ $M^9 : p_1, p_4 \to 1$

$\widetilde{D_c^9}$	$\bullet$	t_1	t_2
$\bullet$	0	1	2
t_1	0	0	2
t_2	0	1	0

$\widetilde{E_c^{10}}=$ $M^{10} : p_1, p_4 \to 1$

$\widetilde{D_c^{10}}$	$\bullet$	t_1	t_2
$\bullet$	0	2	3
t_1	0	0	3
t_2	0	1	0

4.3.4 Automate temporisé d'un ITPN.

Dans, [49] les auteurs proposent une méthode pour la traduction d'un $ITPN$ vers un automate temporisé. Cette traduction est une approximation par DBM. Elle permet par la suite d'exploiter le AT généré pour l'analyse des propriétés aussi bien linéaires que de branchement en utilisant des outils tels que $UPAAL$ ou $HyTech$. Cette approche est jugée avantageuse par les auteurs car elle permet de minimiser le nombre d'horloges de l'automate généré, induisant un coût moindre lors de la phase de vérification.

4.3.5 Exemple

Reprenons le même réseau de Petri à arcs inhibiteurs précédent, son automate généré est représenté sur la figure FIG 4.4. Ce dernier contient 18 localés différentes et 27 arcs.

En comparant avec le graphe exact, on remarque que la classe 16 et l'arc sortant sont en plus par rapport à ce dernier automate, et pour l'automate on a le tir de t_6 qui génére par rapport au graphe exact la localité 12 avec l'arc entrant et les deux arcs sortants, donc si le graphe exact a une classe différente avec son arc sortant, l'automate en a une localité et 3 arcs en plus. Ce qui fait qu'ils ont

le même nombre de localités mais que l'automate a deux arcs en plus induits par sur-approximation DBM.

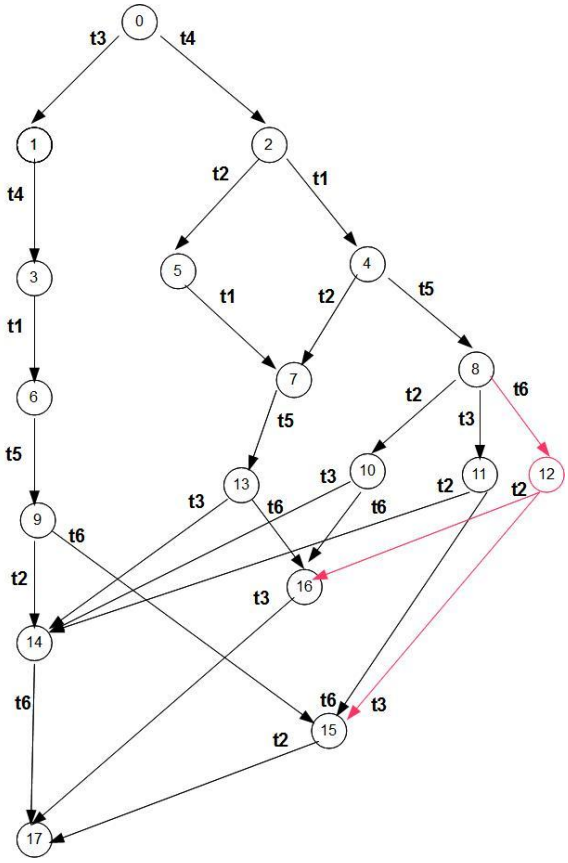


FIGURE 4.4 – L'Automate correspondant à l'$ITPN$ précédent.

4.3.6 Construction du graphe de classes approché par la K-grille

Dans [12, 13], les auteurs avaient proposé une autre méthode de sur-approximation, basée sur une quantification des polyèdres représentant l'information temporelle dans les classes. En ajustant un paramètre, le comportement exact du réseau de Petri à chronomêtres peut être approché aussi étroitement que souhaité. Nous ne détaillons pas plus la méthode mais nous allons voir et comparer les résultats avec les méthodes représentée ci-dessus.

4.3.6.1 Exemple

Si l'on considére le même réseau de Petri de la figure (FIG 4.1a), les graphes représentants la méthode de sur-approximation, basée sur une quantification des polyèdres ci-dessus avec le paramétre K=0 on obtient le graphe exact de la figure (FIG 4.1b).

Il a été remarqué que le comportement approximé devient non borné quand la taille de la grille est choisie plus grande que 1. Cette approximation reste néanmoins plus coûteuse que les approximations par DBM, mais conduit à des approximations plus précises, et sa précision peut être paramétrée. Enfin, notons que la technique de quantification est aussi applicable aux $RdPT$ (sans chronomètres), elle permet là de contracter les graphes par approximation.

4.3.7 Le calcul d'une approximation de la réponse du temps borné d'une séquence de tir

Pour l'analyse des systéms temps réel préemptifs, l'auteur [1], a proposé un algorithme pour le calcul d'une approximation du temps minimal et maximal d'une séquence de tir. L'avantage d'une telle méthode, contrairement à d'autres techniques existantes [13, 62, 19], est que l'algorithme satisfait une complexité linéaire en coût et améliore la vérification par la construction à la volée sans étendre le modèle avec des *observateurs* (voir chapitre 1).

Les résultats montrent que le temps nécessaire pour estimer les valeurs approximées de la réponse du temps borné ($BCRT$ et $WCRT$: Best et Worst cases response times) d'une tâche sont moins importants que ceux nécéssaires pour calculer les graphes $\widetilde{GR}$. Les valeurs $BCRT$ calculés sont les même pour tout le graphe quelque soit la méthode, alors que les $WCRT$ calculés sont différents. Il a été remarqué que l'algorithme produit des résultats plus performant en l'appliquant sur le GR aulieu de de l'appliquer sur le $\widetilde{GR}$.

La limite de cette méthode réside dans le fait que les séquences additionnelles rajoutées dans certaines méthodes par rapport aux autres produisenst des cas supplémentaire dans les calcules détruisent ainsi l'estimation de $WCRT$.

4.4 Discussion

Nous comparons dans cette section les différentes constructions présentées plus haut, nous constatons que :

La méthode exacte basée sur une représentation exclusivement polyédrique, induit des temps de calcul prohibitifs, voir des débordement de mémoire, lorsque le système d'inéquation prend des formes élaborées. Le facteur coût pénalise fortement la construction exacte. En effet, un polyèdre nécessite des structures plus

complexes pour être représenté en mémoire. De plus le coût de manipulation d'un tel système est exponentiel au nombre des variables.

Pour contourner les problématiques liées à la manipulation des systèmes polyédriques, la sur-approximation par DBM a été proposée comme solution alternative. Cette solution permet de construire un graphe moins riche et souvent grossier, mais néanmoins de coût sensiblement inférieur en termes de temps de calcul et d'occupation mémoire. De plus, les propriétés des DBM permet que cette surapproximation calcule un graphe fini, alors que le graphe exact est infini. Ces méthodes permettent de déduire un algorithme qui permet de calculer directement le système DBM dans sa forme normale sans passer par le polyèdre intermédiaire. En évitant la manipulation du polyèdre et les phases de normalisation et de minimisation, une optimisation significative pour l'implémentation du calcul d'une classe approchée, en évitant les défauts d'implémentation constatés pour les autres approches. La contraction de cette dernière méthode permet de définir une relation d'équivalence moins restrictive que l'égalité compactant ainsi davantage le graphe approché et réduisant son coût de calcul.

Il faut néanmoins souligner que si les méthodes de sur-approximation par DBM sont trop grossières et non paramétrables (car elle peuvent être améliorée suivant ce point) par rapport à la méthode basées sur la quantification des polyèdres, elle ne sont pas aussi fines. En effet, si cette dernière est paramétrable et plus précise, elle est plus coûteuse que les autres méthodes en matière d'espace mémoire et temps de calcul.

Les résultats expérimentaux [2], montrent que : Le temps de calcul est en faveur de l'algorithme proposé, spécialement en calculant le $\widetilde{GRC}$. La construction $\widetilde{GR}$ réduit sensiblement la taille du graphe aussi bien que le calcul de ce dernier. Cette contraction produit un graphe trés compact qui est souvent plus petit que le graphe exacte. En effet, ceci ne voudrait pas dire que $\widetilde{GRC}$ est plus précis dans que $\widetilde{GR}$, mais que certaines classes inégales dans $\widetilde{GR}$ et GR sont bissimilaire et sont regroupées dans le $\widetilde{GRC}$ par application de la relation d'équivalence.

Dans certains cas, le calcul du graphe exact échoue car le nombre de polyédres inégaux calculés est non borné, alors que les approches basées sur l'approximation par DBM convergent car le nombre des systèmes DBM obtenu par approximation est toujours borné.

La méthode [1] présente certains avantages. En effet, cette dernière ne nécessitant pas d'étendre le modèle avec des observateurs pour la vérification des propriétés quantitatives du modèle et l'algorithme satisfait une complexité linéaire en coût par rapport aux autres méthodes existantes. La limite réside dans les cas supplémentaires générés par les différentes méthodes utilisées réduisant l'estimation de la valeur WRT calculé.

Les méthodes sus-citées permettent d'avoir des graphes préservant certaines propriétés telles que les propriétés linéaire LTL, cependant la construction exacte

ou approchée du graphe de classes atomique n'a pas encore été explorée. La vérification de propriétés $TCTL$ pour les systèmes avec préemption a été introduite à travers la construction de l'AT approché d'un $ITPN$.

4.5 Conclusion

Dans ce chapitre, nous avons présenté les différentes méthodes pour la construction d'abstractions exactes ou approchées de l'éspace d'état des systémes préemptifs modélisés au moyen d'un $ITPN$. Les méthodes exactes présentent des avantages mais aussi des limites par rapport au coût et l'indécidabilité de la finitude du graphe pour les modèles bornés. Les approximations par DBM permettent de pallier à ce dernier problème mais ont l'inconvénient de devenir grossiers lorsque les graphes deviennent volumineux. Pour la vérification des propriétés quantitatives des systèmes temps réels préemptifs la technique standard était l'utilisation des observateurs, d'autres méthodes ont été proposées pour le calcul d'une approximation du temps minimal et maximal d'une séquence de tir. La question qu'on pourrait se poser est : "que ce qu'il en ait de la construction du grapheC pour l'abstraction de l'espace d'etats d'un système préemptif modèlsé au moyen d'un $ITPN$ en sachant que cette derniére méthode s'interesse aussi à l'aspect quantitative du modèle ? Ceci fait l'object du chapitre 5.

Chapitre 5

Cas d'Etude : $ITPN$

5.1 Introduction

La méthode du graphe des classes, initialement appliquée aux RdPT [10](chapitre 4), préservant marquage et propriétés LTL a été étendue au cas des réseaux de Petri à arcs inhibiteurs [12, 60, 13]. Dans ce chapitre, nous explorons d'autres abstractions, préservant non seulement les propriétés CTL^* mais aussi, les contraintes temporelles quantitatives d'un $ITPN$. Enfin, les différentes méthodes étudiées dans ce rapport sont comparées en pratique en utilisant les outils : Tina, Romeo et GraphC.

5.2 Graphe de classes d'états d'un $ITPN$ préservant les propriétés CTL^* et contraintes quantitatives

L'approche discutée dans cette section est celle proposée initialement dans [21, 20] pour les *RdPT*, nous justifions notre choix par le fait que cette méthode a l'avantage, non seulement, d'expliciter les contraintes temporelles quantitatives du modèle, mais en plus, elle présente un raffinement par rapport aux autres méthodes basées classes d'états, qualitativement parlant, Nous préservons en plus des propriétés CTL^* (voir chapitre 3), l'accessibilité des chemins déterministes dans le graphe obtenu.

Cependant, la méthode grapheC est basée sur la représentation de l'espace d'état d'une classe par un réseau de contraintes encodé par des intervalles. Ces derniers définissent des contraintes DBM qui par conséquent ne peuvent capturer certaines informations temporelles synthétisée par les contraintes polyhédriques.

Cette abstraction donc est une approximation par DBM du graphe de classes atomiques exact. Ce dernier peut être construit de la même manière que pour un $RdPT$ mais avec un coût d'ordre exponentiel. Son approximation par DBM permet de ramener son coût à un ordre polynomial mais en tolérant le franchissement éventuelle de séquences additionnelles.

Nous allons tout d'abord définir quelques notions de bases, nécessaires pour explorer la méthode en modeC (C : Contraintes). Nous explorons par la suite, la méthode en l'appliquant, pas à pas, sur un $ITPN$ (Exemple1). Une solution est proposée par la suite, par rapport à laquelle les définitions du chapitre 3 ont été étendues.

Dans un réseau de Petri à arcs inhibiteurs, après le tir d'une transition activée t_i, une transition t_k est soit *Activée*, soit *Inhibée*, cette derniére peut changer d'état :

- t_k Activée $\rightarrow$ t_k Inhibée ;
- t_k inhibée $\rightarrow$ t_k Activée : là on a besoin d'étendre la méthode car des informations manquent pour construire le graphe de classe au complet (voir l'exemple1 qui montre le besoin d'étendre la méthode).

Ou pas :

- t_k Activée $\rightarrow$ t_k Activée ;
- t_k inhibée $\rightarrow$ t_k inhibée :

Les événements à considérer (et donc les variables temporelles associées aux dates de ces événements) sont :

- L'activation d'une transition ;
- Le début de l'intervalle de tir ;
- La fin de l'intervalle de tir ;
- Le franchissement effectif de la transition ;
- Désactivation (Inhibition) d'une transition ;
- Réactivation d'une transition inhibée dans le passé.

En fait, on s'intéresse plus particulièrement, au cas où t_k est réactivée (après avoir été inhibée). En effet, pendant la construction d'un réseau de contraintes temporelles, à un moment donné, t_k devient active et dois être considérée pour continuer la construction du graphe. L'exemple1 traite un cas générale où, la transition t_1 passe de l'état activée à inhibée et vis-vers-ça. Mais avant d'explorer la construction d'un graphe de classes pour un $ITPN$, des notions de bases nécessaires sont à définir :

5.2.1 Notions de bases

Dans cette section, nous allons présenter briévement des notions nécessaires pour comprendre le principe de la construction d'un graphe basé sur les STN et

la théorie des intervalles [22].

La construction du graphe de classes d'un réseau de Petri temporel est basée sur deux notions : *l'intervalle de tir* d'une transition, et le *domaine temporel* des transitions qui restent sensibilisées aprés le tir d'une transition. Une classe est caractérisée par un marquage et un domaine temporel ; l'arc reliant deux classes C et $C^{'}$ du graphe est étiqueté par l'intervalle de tir $D(t_i)$ de la transition t_i qui provoque le changement d'état. Le domaine temporel de cette nouvelle classe d'état $C^{'}$ est donné par :

– L'intervalle de tir $I^{'}(t_k)$ de toutes les transitions t_k activées par le marquage de $C^{'}$;

– Les contraintes de temps $[t_k, t_i]$ entre tous les couples de transitions (t_k, t_i) qui restent activées.

Pour que t_i puisse être franchie avant t_k, cette contrainte doit être un intervalle temporel positif. Elle doit prendre en compte aussi les contraintes dans les classes précédentes (mémoire du passé).

Dans [21, 20], les auteurs ont exprimé le calcul du domaine temporel d'une classe ainsi que le calcul de l'intervalle de tir en utilisant la théorie des intervalles, au lieu d'utiliser le graphe de régions comme [8].

Soient :

– $I_0(t_i) = [a_i, b_i]$ l'intervalle temporel statique associé à une transition t_i ;

– $I^j(t_i)$ son intervalle temporel dynamique dans la classe C_j ; et

– $[t_k, t_i]_0 = (I^0(t_k) - (I^0(t_i)) \cap [0, \infty) = [a_k - b_i, b_k - a_i] \cap [0, \infty)$

la contrainte temporelle dans la classe initiale C_0. On considére qu'il n'y a pas de contrainte avant l'instant 0, $[t_k, t_i]_{-\infty} \subseteq (-\infty, \infty)$. La contrainte de temps entre deux transitions t_i et t_k dans une classe C_j , $j \geq 0$ est donnée par :

$$[t_k, t_i]_j = (I^j(t_k) - (I^j(t_i)) \cap [t_k, t_j]_{j-1} \cap [0, \infty) \tag{5.1}$$

Si $[t_k, t_i]_j \neq \emptyset$ alors t_i peut être franchie avant t_k. Si $[t_k, t_j]_{j-1} \subseteq I^j(t_k) - I^j(t_i)$, alors la restriction dans le passé est redondante.

Comme on utilise la sémantique forte, une transition t_i peut être franchie à partir de C_j pendant son intervalle de tir, mais avant la plus petite borne supérieur de toutes les transitions activées dans C_j :

$$D^j(t_i) = [a_i, b_i] \bigcap_k [0, b_k] \tag{5.2}$$

Si l'on note $sUB = min_k(b_k)$ la plus petite borne supérieure de toutes les transitions activées dans la classe C_j , l'équation (5.3) correspond à $D^j(t_i) =$

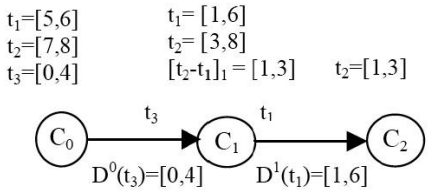


FIGURE 5.1 – Graphe de classes d'un RdPT.

$[a_i, sUB]$. Aprés le tir d'une transition t_i à partir d'une classe C_j, l'intervalle de temps des autres transitions t_k doit être mis à jour dans la classe C_{j+1} :

$$I^{j+1}(t_k) = [t_k, t_i]_j \cap (I^j(t_k) - D^j(t_i) \cap [o, \infty) \tag{5.3}$$

Soit par exemple le cas d'un réseau de Petri temporel avec trois transitions paralléles t_1, t_2 et t_3 activées dans la classe initiale C_0, avec comme intervalle statique $I_0(t_1) = [5, 6]$, $I_0(t_2) = [7, 8]$ et $I_0(t_3) = [0, 4]$. En appliquant les équations (5.1), (5.3) et (5.1), (5.4) on calcule la nouvelle classe C_1, puis en appliquant les équations (5.2), (5.3) et (5.4), la classe C_2. Le graphe de classes d'état résultant est représenté figure FIG 5.1.

Les opérations de base de ce calcul sont la différence d'intervalles et l'intersection d'intervalles.

Définition 32 (intervalle dynamique) *[20] Soient t_k une transition activée suite au tir de la transition $t_{s(k)}$, dans la classe C, son temps d'activation initiale est son intervalle statique $I(t_k)$. Si t_k reste activée pendant le tir de t_i, les dates de tir possible pour t_k sont exprimées dans son intervalle dynamique donnée dans la classe accessible, noté $I_d^c(t_k)$. En prenant la même origine du temps $x_{s(k)}$ et $I(t_k) : D(t_i) = C_{s(k),i}$ (avec des valeurs positifs du temps).*

$$I_d^c(t_k) = (I(t_k) - C_{s(k),i}(x_{s(k)}, x_i)) \cap [0 \ \ \infty[\tag{5.4}$$

Dans le cas d'un $ITPN$, il faut prendre en considération qu'une transition qui a été inhibée n'a plus la même origine du temps que le système (x_0), en d'autre termes, la date de création des jetons et la date d'activation des transitions activées n'est pas forcément la même.

5.2.2 Limites de la méthode pour un $ITPN$

Considérons l'$ITPN$ de la figure Fig. 5.2.a) et le $RdPT$ sous-jacent Fig. 5.2.b), étudiés avant dans [50]. Les graphes de classes modeC sont représentés par

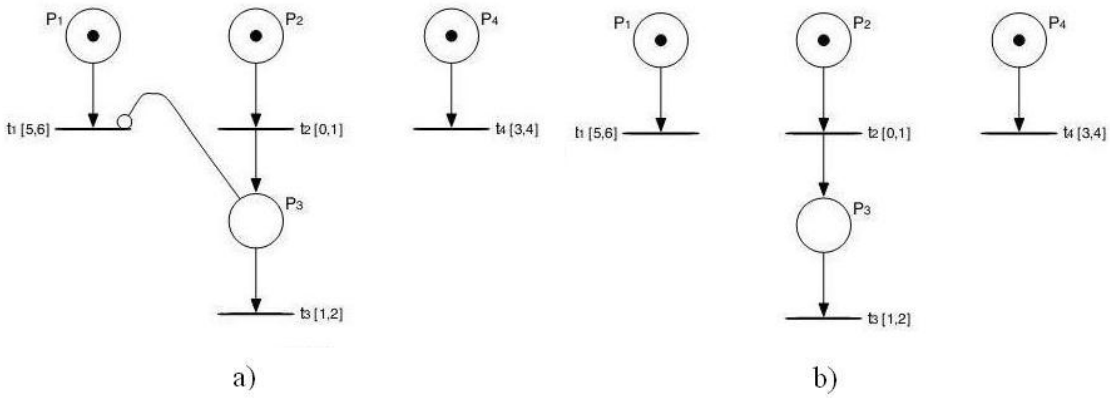


FIGURE 5.2 – Exemple1 : a) $ITPN$; b) $RdPT$ sous-jacent

Fig. 5.7 a) et b) respectivement. Dans cet exemple la transition t_1 passe de l'état activée à inhibée, puis réactivée à nouveau par x_0, x_2 et x_3 respectivement.

Les transitions t_1, t_2 et t_4 sont sensibilisées initialement où seule t_2 est tirable pour ce $RdPT$ (sémantique forte). Pour l'$ITPN$, t_1, t_2 et t_4 sont activées et seule t_2 peut être tirée. Construisons le graphe modeC pour cet $ITPN$..
Intuitivement, la classe initiale C_0 correspond à l'état initial définie par le marquage initiale $M_0 = p_1, p_2, p_4$ et le réseau de contraintes Nc_0 formé d'un seul noeud x_0 correspondant à un événement initial de création des jetons du marquage initial, il représente (comme pour un $RdPT$) le début du temps.

A partir de la classe initiale C_0 on ne peut tirer que t_2 délimitée par le réseau de contraintes $Nt_{2,0}$ en gras sur la figure FIG 5.3 a). x_0 étant le réseau de contraintes de la classe précédente, la contrainte entre x_0 et x_2 est l'intervalle statique de t_2 : $(x_0, x_2) = [1 \;\; 2]$. La variable x_2 correspond au tir de t_2 contrainte par t_1 et t_4 (t_2 ne peut être tirée avant la fin de l'intervalle temporel de t_1 et t_4 représenté par les variables y_1 et y_4), la valeur des arcs sont respectivement $(x_0, y_1) = [6 \;\; 6]$, $(x_0, y_4) = [4 \;\; 4]$, qui représente les dates maximales possibles de tir de t_1 et t_4.

Après franchissement de t_2 on arrive à la classe C_1 avec le marquage M_1 : $p_1, p_3, p_4 \longmapsto 1$ et le réseau de contraintes Nc_1 défini par le réseau Nc_1 représenté par la contrainte $(x_0, x_2) = [0 \;\; 1]$; x_2 étant la variable qui représente l'instant du dernier franchissement qui a activé t_3 ; x_0 l'instant de l'activation de t_4. Dans cette classe on ne considère que les transitions activées t_3 et t_4, t_1 est inhibée et ne contraint pas le tir des deux autres transitions activées :

– Le tir de t_3 avec le réseau de contraintes $Nt_{3,1}$, est défini par : le réseau

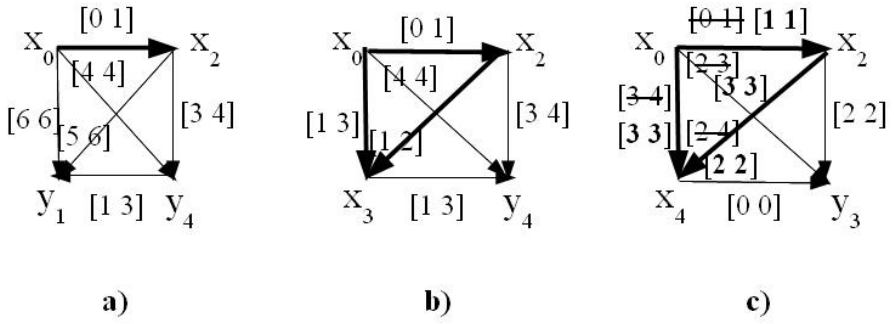


FIGURE 5.3 – Construction des réseaux de contraintes : a)$Nt_{2,0}$; b)$Nt_{3,1}$; c)$Nt_{4,1}$

de contraintes précédent, $Nc_1 : (x_0, x_2) = [0 \ \ 1]$, la variable x_3 qui représente l'instant de franchissement et y_4 la variable qui représente les dates maximales de tir de $t_4 : (x_0, y_4) = [4 \ \ 4]$; t_3 étant activée suite au tir de t_2, donc nous obtenons : $(x_2, x_3) = [1 \ \ 2]$ (l'intervalle statique de t_3). Après application de l'algorithme de Floyd-Warschall et suppression de y_4 on obtient la classe C_2 avec le marquage $M_2 : p_1 p_4 \longmapsto 1$ et le réseau de contraintes Nc_2, en gras sur la figure FIG 5.3 b).

– Revenons à la classe C_1, où t_4 est aussi tirable, le réseau de contraintes $Nt_{4,1}$ FIG 5.3.c), est défini par le réseau de contraintes Nc_1 précédent $(x_0, x_2) = [0 \ \ 1]$, la variable x_4 qui représente l'instant du dernier franchissement et y_3 la variable qui représente les dates maximales de tir de t_3, $((x_2, y_3) = [2 \ \ 2])$, après application de l'algorithme de Floyd-Warschall, on obtient un réseau plus restreint, on retient notamment la contraintes $(x_0, x_2) = [1 \ \ 1]$ (qui va être considérée plus tard pour le calcul des classes restreintes). On aboutit à la classe C_3 avec comme marquage $M_3 : p_1, p_3 \longmapsto 1$ et le réseau des contraintes Nc_3 : $(x_0, x_2) = [1 \ \ 1]$, $(x_0, x_4) = [3 \ \ 3]$, $(x_2, x_4) = [2 \ \ 2]$. La variable x_4 représente le dernier franchissement, x_2 la variable qui active t_3 et x_0 la variable qui a activé t_1 dans le passé.
On ne peut tirer que t_3 à partir de C_3 car t_1 est inhibée ; $Nt_{3,3}$ est défini par $(x_2, x_4) = [2 \ \ 2]$, $(x_2, x_3) = [2 \ \ 2]$, $(x_3, x_4) = [0 \ \ 0]$ et $(x_0, x_2) = [1 \ \ 1]$, $(x_0, x_4) = [3 \ \ 3]$, $(x_0, x_3) = [3 \ \ 3]$. Il est à noter qu'on garde en mémoire d'un passé un peu loin pour la variable inhibée t_1 (voir figure FIG 5.6.f). On obtient la classe C_5 avec le marquage $M_5 : p_1 \longmapsto 1$ et le réseau de contraintes $(x_0, x_3) = [3 \ \ 3]$; x_3 étant la variable qui représente le temps du dernier franchissement et la variable x_0 l'instant d'activation de t_1.

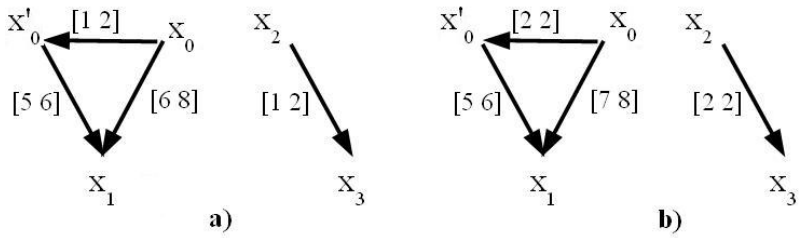


FIGURE 5.4 – Inhibition de t_1 : a) La séquence σ_1 ; b) La séquence σ_2

La construction du graphe des classes précédent est confronté aux problématiques suivantes :

1. On n'a pas assez d'informations temporelles pour le tir de t_1 réactivée depuis les classes C_2 et C_5,
2. Pour les deux séquences de tir, t_1 est inhibée différement.

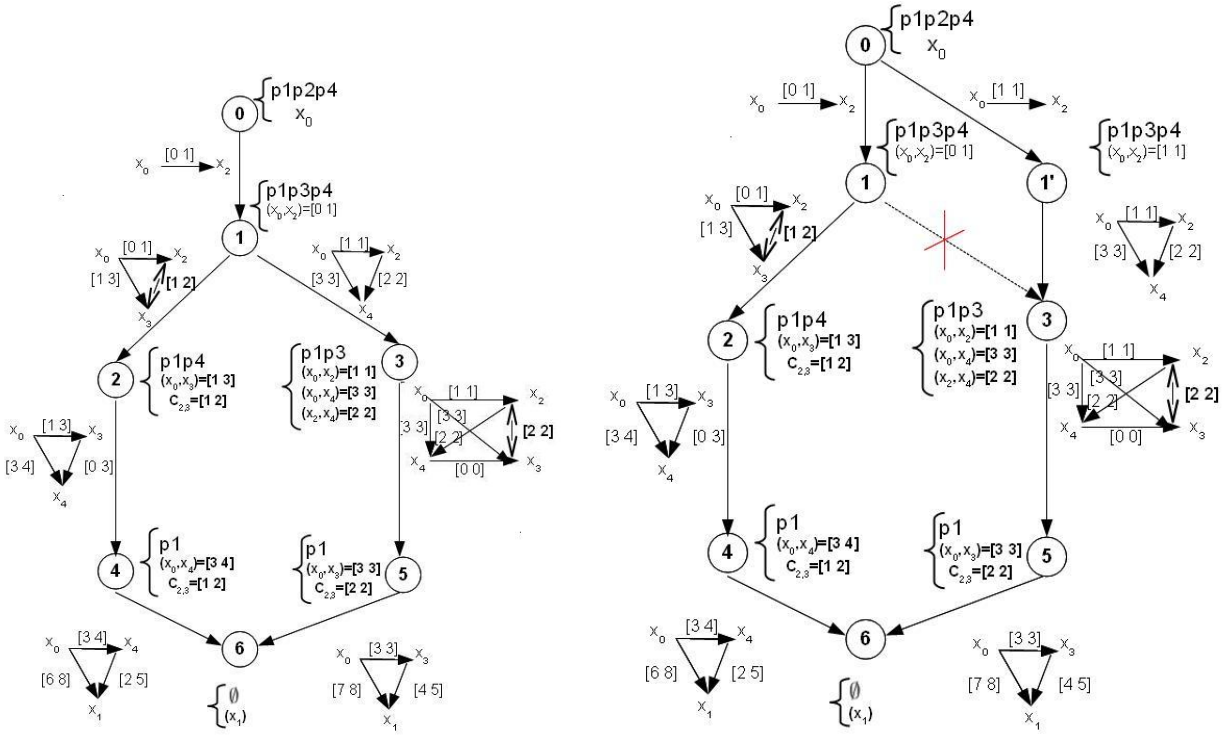
Soit x_0 la date de création des jetons de t_1, comme pour le système, et x'_0 la date de réactivation de t_1. Soit les deux séquences de tir : $\sigma_1 = t_2; t_3; t_4$ et $\sigma_2 = t_2; t_4; t_3$. Pour la premiére séquence, t_1 est inhibée durant $(x_2; x_3) = [1\ \ 2]$ et peut être tirée durant l'intervalle : $[6\ \ 8] = [5\ \ 6] + [1\ \ 2]$ $(I(t_1) + (x_2; x_3))$, pour la deuxiéme séquence $(x_2; x_3) = [2\ \ 2]$ et t_1 est tirée durant $[7\ \ 8] = [5\ \ 6] + [2\ \ 2]$ $(I(t_1) + (x_2; x_3))$ voir la figure 5.4.a) et b).

La transition t_1 n'est plus tirée dans son intevalle statique $I(t_1)$ mais plutôt durant son intervalle : $I(t_1)'$ qui est calculé de la maniére suivante :
$I(t_1)' = I(t_1) + In_{c,1}$;

$In_{c,1}$ correspond à l'intevalle d'inhibition de t_1 dans la classe C, qui n'est rien d'autre que la distance (la contrainte), entre la variable qui représente l'instant de tir de la transition qui a inhibé, et celle qui a réactivé t_1. Dans notre exemple, ceci correspond à x_2 et x_3 est représentée par des fléches particuliéres sur la (FIG 5.5.a) et b).

Une autre limite de cette méthode pourrait apparaitre en l'appliquant à un $ITPN$, on peut voir que dans la classe C_6, la transition t_1 n'a pas la même valeur $In_{c,1}$ pour σ_1 et σ_2, ceci pourrait être corrigé en éclatant la classe C_6 suivant cette nouvelle donnée (C_7) (FIG 5.5.a) et b). Le graphe final est représenté sur la figure 5.7.a) pour l'$ITPN$ et 5.7.b) pour le $RdPT$ sous-jasent.

Les paragraphes précedents illustrent la premiére phase de calcul du graphe en modeC de l'$ITPN$ de l'exemple1, la deuxiéme phase consiste en le calcul des



a) Phase (1) b) Phase (2)

FIGURE 5.5 – Calcul du graphe modeC pour un $ITPN$: a) Phase 1 b) Phase 2 (classes restreintes)

classes restreintes qui est décrit dans les paragraphes suivants.

Pour rappel, la nouvelle restriction est $(x_0, x_2) = [1 \ \ 1]$ au lieu de $(x_0, x_2) = [0 \ \ 1]$ durant le calcul du réseau de contrainte $Nt_{4,1}$. Ceci se traduit par la création d'une nouvelle classe restreinte de C_1 noté, C_1' avec le même marquage et ce nouvel intervalle comme contrainte noté Nc_1' (voir la figure FIG 5.5 b). On crée ensuite un arc reliant la classe initiale et cette derniére classe, ce dernier est représenté par le réseau de contraintes $Nt_{2',0}$: $(x_0, x_2) = [1 \ \ 1]$ et on supprime l'arc reliant la classe C_1 à C_3, celui-ci sera remplacé par l'arc C_1' à C_3.

Nous avons ajouté certaines informations pour une adaptation correcte, aboutissant à notre proposition :

5.2.3 Proposition

La construction du graphe des classes d'états modeC pour un $ITPN$ est la même sauf quelques changements. Si $t_s(k)$ est la transition qui a activé la tran-

sition $t(k)$, alors $t_h(k)$ est la transition qui l'a inhibé. Soit x_i la variable qui correspond à la transition activée t_i et tirable dans l'intervalle $I(t_i)'$.

Pour toute classe C appartenant au graphe modeC, il existe deux cas particulier :

– La transition activée t_i n'a jamais été inhibée ($In_{c,i} = [0\ \ 0]$) ; elle a la même origine du temps que le système x_0 et t_i est tirée dans $I(t_i)$;

– La transition t_i est réactivée et n'a plus le même origine du temps que le système, dans ce cas, t_i est tirée dans $I(t_i)'$; pour un $ITPN$.

Remarque 5 *Si toutes les transitions ont la même origine du temps que le système, on se trouve dans le cas particulier d'un RdPT.*

Le dernier franchissement (représenté par la variable x_i^{io}), peut correspondre à deux événements différents :

– Activation/ réactivation d'une transition inhibée ou pas ($x_i^{io} = x_{s(k)}^{osk}$) ;

– Désactivation d'une transition activée ; ($x_i^{io} = x_{h(k)}^{ohk}$)

Pour notre exemple, la transition t_1 a été inhibée par t_2 : $x_{h(k)}^{ohk} = x_2$, puis réactivée par le tir de t_3 : $x_{s(k)}^{osk} = x_3$.

L'information qu'on doit rajouter et qui est systématique est que le système a évolué pendant que t_1 a été inhibée, donc t_1 n'a plus la même origine du temps que le système (x_0), et cette valeur n'est rien d'autre que la distance entre $x_{h(k)}^{ohk}$ et $x_{s(k)}^{osk}$.

Remarque 6 *La transition inhibée n'influe pas sur le tir des transitions activées. Celle-ci ne contraint pas le domaine de tir.*

On a ajouté les informations suivantes :

– Les transitions activées sont tirées dans l'intervalle $I(t_i)'$;

– On doit différencier les classes par rapport à l'intevalle d'inhibition des transitions réactivées ;

– Les transitions inhibées ne contraignent pas les dates de tir, mais la variable $x_{h(i)}^{ohi}$ doit être considérée dans le calcul des STN_s jusqu'à détérmination de la valeur d'inhibition de cette derniére.

On peut à présent, continuer la construction du graphe précédent.

5.2.4 Illustration

Considérons le résrau de Petri à arcs inhibiteurs précédent, le tir de t_3 à partir de C_1, mène à la classe C_2 représentée par le marquage $M_2 : p_1, p_4 \longmapsto 1$, où t_1 et

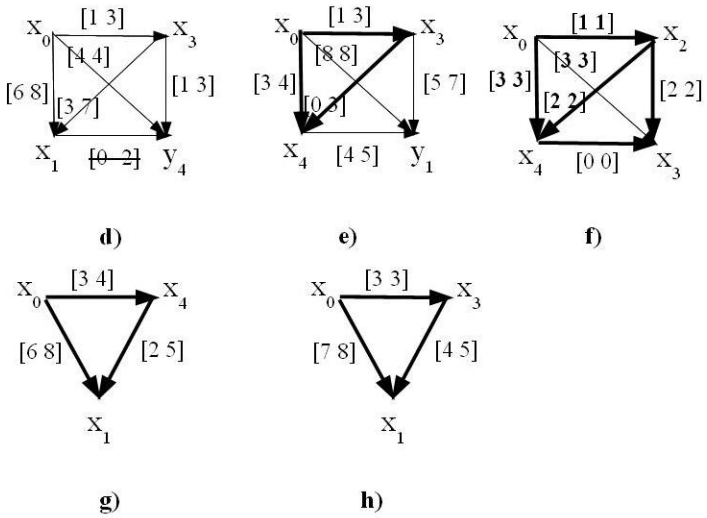


FIGURE 5.6 – Construction des réseaux de contraintes : e)$Nt_{4,2}$; f)$Nt_{3,3}$; g)$Nt_{1,4}$; h)$Nt_{1,5}$

t_4 sont activées et peuvent être tirées, et le réseau de contraintes Nc_2 est défini par $(x_0, x_3) = [1\ \ 3]$ (x_3 à la fois instant du dernier franchissement menant à C_2 et la variable qui représente la transition qui a réactivé t_1, x_0 variable qui a activé t_1 et t_4).

A partir de C_2, on peut tirer soit t_1 soit t_4, le tir de t_1 est représenté par le réseau de contraintes *non consistant* de la figure FIG 5.6 d)(qui peut exprimer le fait que t_1 n'est pas tirable avant le tir de t_4) à préciser que l'intervalle temporel de t_1 ne change pas quand cette dernière est inhibée : $I_d^1(t_1) = I_d^2(t_1)$ et que t_1 est inhibée par la tir de t_2 puis réactivée par le tir de t_3. Intuitivement, pour t_1 le temps ne s'est pas écoulé et la distance entre t_2 et t_3 est nulle, d'une autre maniére, l'instant du tir de t_1 est décalé par rapport à l'origine du temps du système (x_0) et la nouvelle distance (x_0, x_1) (en considérant son inhibition $In_{2,1}$ à partir de la classe C_2 est calculé de la maniére suivante :

$(x_0, x_1) = (x'_0, x_1) + (x_2, x_3) = [5\ \ 6] + [1\ \ 2] = [6\ \ 8]$, où (x_2, x_3) représente l'inihition de t_1 (distance entre la variable qui à inhibé t_1 (x_2) et la variable qui l'a réactivée (x_3) : $C_{2,3}$.

Considérons maintenant le tir de t_4 avec le réseau de contraintes $Nt_{4,2}$ (représenté en gras sur la figure FIG 5.6 e) avec $(x_0, x_3) = [1\ \ 3]$ le réseau de contraintes précédent (Nc_2). x_4 représente l'instant du tir de la transition t_4 et y_1 les dates maximales de tir de t_1 : $([6\ \ 8])$. La transition t_1 est activée par x_0 $(x_0, y_1) = [8\ \ 8])$. On obtient la classe C_4 avec marquage $M_4 : p_1 \longmapsto 1$ et le réseau de contraintes $Nc_4 : (x_0, x_4) = [3\ \ 4]$, où x_4 représente le dernier franchissement et x_0 la variable

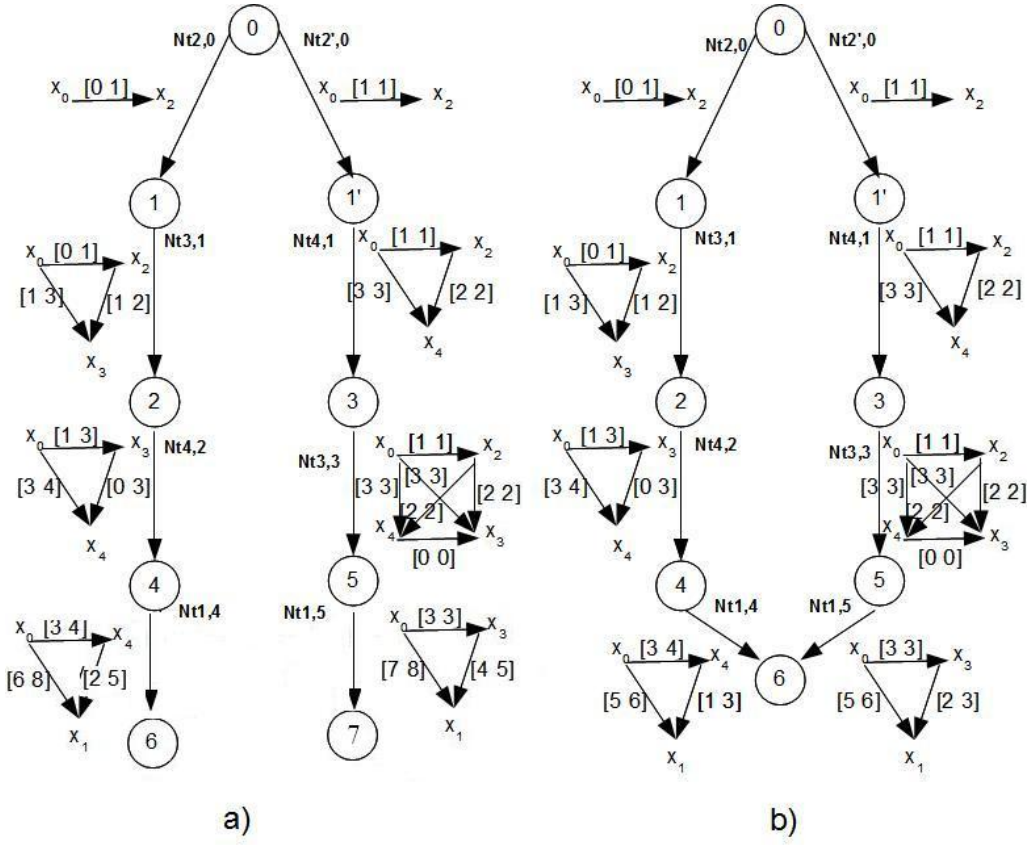


FIGURE 5.7 – Graphe de classe modeC de l'exemple1 : a) Pour l'$ITPN$; b) Pour le $RdPT$ sous-jacent

qui correspond à l'activation (décalée) de t_1 et $(x_2, x_3) = [1 \quad 2]$. à partir de là, seule la transition t_1 est tirable avec le réseau de contraintes $Nt_{1.4}$ en gras sur la figure FIG 5.6.g).

Le tir de t_1 à partir de C_5 est représenté par la figure FIG 5.6.h), en prenant en considération la distance (x_2, x_3) ($In_{5,1}$ de t_1), qui est égale de $C_{2,3} = [2 \quad 2]$ et qui n'est en fait que la distance entre l'origine du temps pour le système x_0 et l'origine du temps pour t_1 (voir figure FIG 5.4 a) et b)).

5.2.5 Construction du graphe modeC d'un $ITPN$

Dans cette section nous allons définir une adaptation simple de construction d'un graphe modeC pour un $ITPN$, en prenant en considération les transitions activées, celles-ci peuvent être inhibées dans le passé ou pas. En effet, ces derniéres

n'ont plus le même origine du temps que le système. Il est à noter que dans le cas où le système a le même origine du temps que toutes les transitions activées, la définition ne change pas, dans le cas contraire, il faut prendre en considération le décalage des transitions réactivées.

Soit la séquence de transition $\sigma = t_1, ...t_i$ pour un $ITPN$. t_i la derniére transition franchie, $t_s(k)$ et $t_h(k)$ les transitions qui ont respéctivement, activé (réactivé) et inhibé la transition t_k. Soit t_j la transition parmi les n activées et franchissables dans la classe $C = (M, Nc, In_{i,c})$. Soit $Nt_{j,c}$ le réseau de contraintes temporelle délimitant le franchissement de t_j à partir de C pour atteindre $C^{\uparrow}$. La construction du graphe de claasses modeC pour un $ITPN$ est la suivante :

Définition 33 *Le graphe de classes modeC d'un $ITPN$, noté $GR\mathcal{C}$, est un triplet $(C\mathcal{C}, \mathcal{C}_0, \twoheadrightarrow)$ où :*
- $C\mathcal{C}$ est l'ensemble des classes accessibles dans $GR\mathcal{C}$;
- $C_0 = (M_0, Nc_0, \emptyset)$ est une classe de $C\mathcal{C}$ appelée classe initiale où :
– M_0 est le marquage initial du $ITPN$.
– Nc_0 est le réseau de contraintes temporelles composé par la variable x_0 représentant l'origine de temps. Au départ aucune transition n'est encore réactivée.
- $\twoheadrightarrow$ est la relation de transition entre classes définie sur $C\mathcal{C} \times (C\mathcal{A}, T) \times C\mathcal{C}$, telle que
$((M, Nc, In_{j,c}), (Nt_{j,c}, t_j), (M^{\uparrow}, Nc^{\uparrow}, In^{\uparrow}_{j,c})) \in \twoheadrightarrow$, si et seulement si :
$\forall p \in P, M(p) \leq Pre(p, t_j)$, et le réseau $NF_{j,c}$ est consistant, tel que $NF_{j,c}$ est obtenu comme suit :

a) *Au départ, $NF_{j,c} = Nc$.*

1. *La variable x_j (tir de t_j) et son intervalle $I'(t_j)$ comme contraintes entre x^{osj}_{sj} et x^{oj}_j (associée au franchissement ayant activé t_j) ;*
2. *Les variables activées, $y^{ol}_l (l \neq j)$, correspondant aux dates maximales de franchissement des autres transitions activées. Ajouter comme contraintes entre chaque couple (x^{osl}_{sl}, y^{ol}_l) l'intervalle singleton $[d'_{Ml}, d'_{Ml}]$ correspondant à la borne maximale de l'intervalle $I'(t_l)$;*
3. *Ajouter la contrainte $[0, \infty]$ entre variables x^{oi}_i et x^{oj}_j pour exprimer le fait que t_j doit être franchie aprés t_i.*
4. *Ajouter la contrainte $[0, \infty]$ entre les couples de variable x^{oj}_j et $y^{ol}_l (l \neq j)$, pour exprimer le fait que t_j doit être franchie âvant la borne maximale de t_l.*

La classe accessible est obtenue comme suit :

b) $\forall p \in P$, $M^{\uparrow}(p) := M(p) - Pre(p, t_j) + Post(p, t_j)$.

c) *$Nt_{j,c}$ est obtenu à partir de $NF_{j,c}$ en effaçant les variables y^{ol}_l et les contraintes auxquelles elles sont directement liées.*

d) *$Nc^{\uparrow}$ représente un réseau de contraintes temporelles simple comprenant les variables et les contraintes suivantes :*

1. *la variable x_i^{oi} associée au dernier franchissement aprés tir de t_i ;*
2. *pour chaque transition activée / réactivée t_k, les variables $(x_{s(k)}^{osk})$, $(k : 1...n)$ associées aux transitions $t_{s(k)}$.*
3. *pour chaque transition inhibée t_k, les variables $(x_{h(k)}^{ohk})$, $(k : 1...n)$ associées aux transitions $t_{h(k)}$.*
4. *les contraintes temporelles entre ces derniéres variables(STN minimal et complet).*
5. *pour chaque transition réactivée, $In_{i,c}$*

e) *$In_{j,c}^{\uparrow}$ représente un fragment de $Nt_{j,c}$ comprenant les deux variables et la contrainte suivantes : pour chaque transition réactivée t_k dans C, les variables $(x_{h(k)}^{ohk}), (x_{s(k)}^{osk})$ avec $(k : 1...n)$ associées aux transitions $t_{h(k)}$ resp.$t_{s(k)}$ et la contrainte entre ces deux variable.*

En plus de considérer les variables nouvelles dans la construstruction des réseaux de contraintes d).3, on a ajouté un nouveau paramétre e), qui est (comme Nc), lui même un fragment de $Nt_{j,c}$ qu'il faut déterminer dans toute classe contenant des transitions réactivées pas encore tirées (cette valeur n'est rien d'autre que la contrainte entre la variable qui a inhibé et celle qui a réactivé t_k), dans le cas contraire l'ensemble est vide.

Les classes avec leur marquage et leur réseau de contraintes sous forme textuelle sont représentées dans le tableau 5.8.a) et les réseaux de contraintes temporelles associées aux arcs sont représentées dans le tableau 5.9b).

Class	Marking	Constraints Nc	$In_{c,i}$
$\mathcal{C}_0$	$p_1p_2p_4$	x_0	$\emptyset$
$\mathcal{C}_1$	$p_1p_3p_4$	$0 \leq x_2 - x_0 \leq 1$	$\emptyset$
$\mathcal{C}_2$	p_1p_4	$1 \leq x_3 - x_0 \leq 3$	$In_{1,2} = [1\ \ 2]$
$\mathcal{C}_3$	p_1p_3	$1 \leq x_2 - x_0 \leq 1\,;\, 3 \leq x_4 - x_0 \leq 3\,;\, 2 \leq x_4 - x_2 \leq 2$	$\emptyset$
$\mathcal{C}_4$	p_1	$3 \leq x_0 - x_4 \leq 4$	$In_{1,4} = [1\ \ 2]$
$\mathcal{C}_5$	p_1	$3 \leq x_0 - x_3 \leq 3$	$In_{1,5} = [2\ \ 2]$
$\mathcal{C}_6$	$\emptyset$	x_1	$In_{1,6} = [1\ \ 2]$
$\mathcal{C}_7$	$\emptyset$	x_1	$In_{1,7} = [2\ \ 2]$

FIGURE 5.8 – Classes accessibles

Réseaux temporels $Nt_{i,k}$ de t_i à partir de $\mathcal{C}_k$
$Nt_{2,0}$, (x_0, x_2)=[0 1]
$Nt_{3,1}$, (x_0, x_2)=[0 1],(x_0, x_3)=[1 3],(x_2, x_3)=[1 2]
$Nt_{4,1}$, (x_0, x_2)=[1 1],(x_0, x_4)=[3 3],(x_2, x_4)=[2 2]
$Nt_{4,2}$, (x_0, x_3)=[1 3],(x_0, x_4)=[3 4],(x_3, x_4)=[0 3]
$Nt_{3,3}$, (x_0, x_2)=[1 1],(x_0, x_4)=[3 3],(x_0, x_3)=[3 3], (x_2, x_3)=[2 2],(x_2, x_4)=[2 2],(x_4, x_3)=[0 0],
$Nt_{1,4}$, (x_0, x_4)=[3 4],(x_0, x_1)=[6 8],(x_4, x_1)=[2 5]
$Nt_{1,5}$, (x_0, x_3)=[3 3],(x_0, x_1)=[7 8],(x_3, x_1)=[4 5]

FIGURE 5.9 – Réseaux de contraintes des arcs

5.2.6 Discussion

Considérons le graphe de classes d'états du réseau de Petri temporel à arc inhibiteur et son $RdPT$ sous-jacent de l'exemple1 illustré sur les figures FIG 5.10 et FIG 5.11 respectivement.

Si nous franchissons la transition t_2 à l'instant 0, alors la prochaine transition qui pourra être franchie sera nécessairement t_3 suivie de t_4. Par contre, si t_2 est franchie à l'instant 1 la transition t_4 peut être franchie avant t_3. Ainsi, suivant la date d'arrivée dans la classe C_1, l'évolution du réseau est modifiée. En effet, les deux intervalles de tir de ces transitions sont disjoints. Cette information n'apparaît pas dans le graphe des classes de [13] : dans C_1 , aucune information ne nous permet de déterminer quelle transition franchir. Ce pendant cette information est présente dans le graphe de classe modeC, en effet, ceci ce traduit par la création d'une nouvelle classe C_1' avec un intervalle de tir de t_2 plus restreint [1 1] et à partir de laquelle on est sure de tirer t_4 avant t_3.

Une autre information apparaît, mais cette fois-ci, elle concerne uniquement les réseaux de Petri à arcs inhibiteurs. L'intervalle de tir de t_1 à partir des classes C_4 et C_5 du TPN sous-jacent, noté : $D^4(t_1)$; $D^5(t_1)$ sont différents. En modeC, cet intervalle n'est rien d'autre que la distance entre dernier franchissement arrivant à C_4, C_5 $(t_4; t_3)$ resp. et la transition t_1. FIG 5.7.b) :

– Dernier franchissement arrivant à C_4 est t_4 représenté par la variable x_4 : $(x_4, x_1) = [1 \;\; 3]$;
– Dernier franchissement arrivant à C_5 est t_3 représenté par la variable x_3 : $(x_3, x_1) = [2 \;\; 3]$.

On a (pour le $RdPT$) :
– $D^4(t_1) = [1 \;\; 3]$
– $D^5(t_1) = [2 \;\; 3]$

Cet intervalle est modifié suite à l'inibition de t_1, en effet, pour l'ITPN initial de l'exemple1, nous avons :
– $(x_4, x_1) = [2 \;\; 5]$ et $D^4(t_1)' = [2 \;\; 5]$

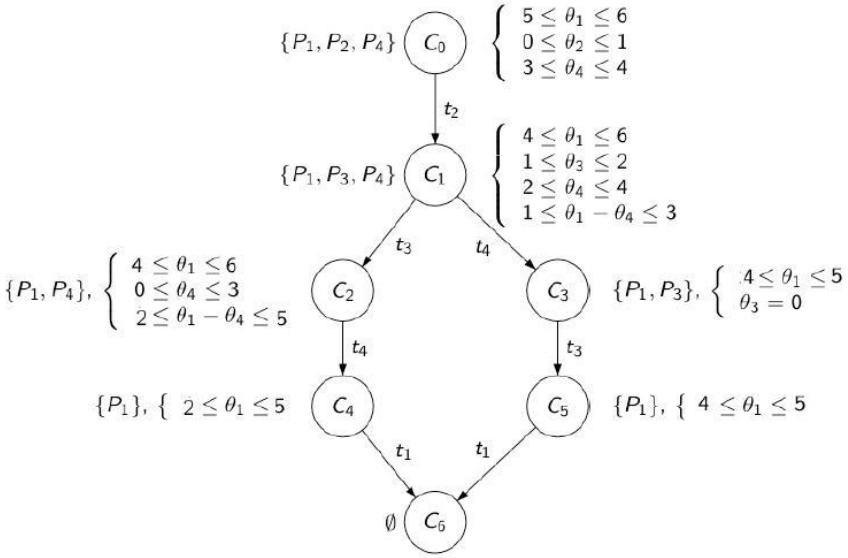


FIGURE 5.10 – Graphe de classes préservant les propriétés LTL (ITPN : Exemple1) [13]

– $(x_3, x_1) = [4\ \ 5]$ et $D^5(t_1)' = [4\ \ 5]$

on voit bien ce décalage du temps, mais aucune information n'y est associée, ceci ce traduit par l'intervalle d'inhibition (l'origine du temps de t_1 change par rapport à l'origine du temps pour le système),qui n'est rien d'autre que la distance entre les deux transitions qui ont inhibée et réactivé t_1, voir proposition plus haut.

– $D^4(t_1)' = D^4(t_1) + In_{1.2} = [13] + [12] = [2\ \ 5] / \ In_{1.2} = (x_2, x_3)$; pour σ_1
– $D^5(t_1)' = D^5(t_1) + In_{1.5} = [23] + [22] = [4\ \ 5] / \ In_{1.5} = (x_2, x_3)$; pour σ_2

5.2.7 EXEMPLE

Considérons le réseau de Petri à arcs inhibiteurs de la FIG 2.5 du chapitre 2. Le graphe de classes modeC généré lors de la première phase est représenté sur la figure FIG 5.14a). Le graphe final aprés calcul des classes restreintes est représenté sur la figure FIG 5.14b). Les séquences de franchissement $\sigma_1 = (t_4; t_1; t_2; t_5)$ et σ_2=$(t_4; t_2; t_1; t_5)$ à partir de la classe initial C_0, représentent deux chemins du graphe FIG 5.13, passant par les classes C_2, C_4, C_7 et C_{26}, C_5, C_9 respectivement, arrivant à la classe C_{11}. Il est a noter que la transition t_3 est inhibée depuis le tir de t_4 et réactivée à partir du tir de t_5, ce qui corresepond à la classe C_{11}. Cette derniére est caractérisée par le marquage M_{11}=$p_3, p_6 \mapsto 1$ et le réseau de contraintes Nc_{11} : $(x_0, x_5) = [3\ \ 3]$ et $In_{3,11} = [2\ \ 3]$, ce qui correspond à la

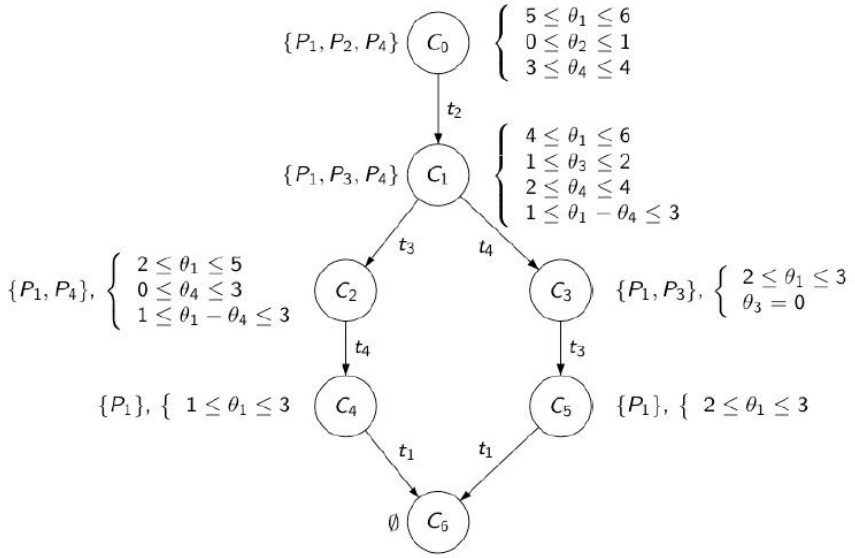


FIGURE 5.11 – Graphe de classes préservant les propriétés LTL(RdPT : Exemple1) [10]

distance (x_4, x_5).

A partir de C_{11}, on peut tirer soit t_3 dans l'intervalle $I'(t_3) = I(t_3) + In_{3,11} = [2\ \ 4] + [2\ \ 3] = [4\ \ 7]$ avec le réseau de contraintes $Nt_{3.11}$, on aboutit à C_{17} FIG 5.12 c). Soit tirer t_6 avec le réseau de contraintes $Nt_{6.11}$ arrivant à C_{18} FIG 5.12 e), voir le détail des classes leurs marquages et réseaux de contraintes ainsi que les résaux de contraintes associés aux arcs de la séquence σ_1 en ANNEXES.

5.2.8 Etude comparative

Considérons, en premier temps, le graphe de classe modeC calculé pendant la premiére phase en appliquant l'algorithme proposé plus haut FIG 5.14a).

Si l'on compare avec le graphe exact du chapitre 4 FIG 4.1, on voit clairement que c'est impossible de *distinguer les états dans le passé*. Prenons la classe E_7, accéssible aprés tir de t_2 ou t_1. On ne peut distinguer les étas accessibles depuis t_2 de ceux accessibles aprés tir de t_1 dans cette derniére classe ? En modeC, ceci se traduit par l'éclatement de E_7 en C_7 (ensembles des états accessible par le tir de t_2 et C_9 ensemble des états accessible aprés tir de t_1. En suivant ce point, nous constatons d'autres correspondances que voici :

– E_{13} correspond à C_{15} pour distinguer ; les états atteints après tir de t_2 et C_{17} pour ceux atteints aprés tir de t_3.

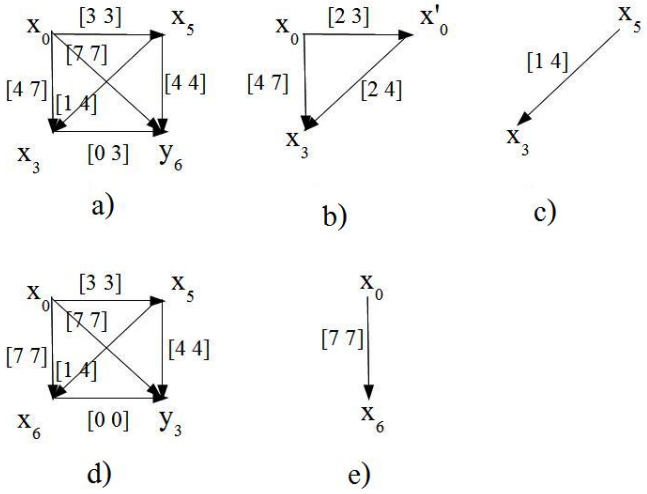


FIGURE 5.12 – Réseaux de contraintes : a) calcul $Nt_{3.11}$; b) décalage t_3 c) $Nt_{3.11}$ d) calcul $Nt_{6.11}$; e) $Nt_{6.11}$

– E_{16} correspond à C_{19} pour distinguer ; les états atteints après tir de t_3 et C_{20} pour ceux atteints aprés tir de t_2.

– E_{17} correspond à C_{23} ; C_{24} ; C_{25} pour distinguer ; les états atteints après tir de t_6 ; t_2 ; t_3 resp.

Cependant comme l'adaptation de la méthode de graphe est une approximation par DBM des contraintes temporelles de'une classe exact. Le tir de la transition t_6 à partir de la séquence $t_4; t_1; t_5$, est permis alors qu'il ne l'est pas dans le graphe exact. Comme pour les approximations par DBM illustrées en chapitre 4, on génére des classes supplémentaires que sont : C_{14} ; C_{21} et C_{22}. Le graphe final comporte 8 classes en modeC rajoutées par rapport au graphe exact, du à la surraproximation et à l'éclatement de certaines classes.

Par ailleurs, si nous comparons le graphe approximé par DBM du chapitre 4 FIG 4.2, avec le même graphe modeC phase 1 FIG 5.14a), nous avons les correspondances suivantes :

– $\widetilde{E_{16}}$ correspond à C_{15} et C_{17} ;
– $\widetilde{E_{17}}$ correspond à C_{16} et C_{22} ;
– $\widetilde{E_{18}}$ correspond à C_{17} ; et C_{20} ;
– $\widetilde{E_{19}}$ correspond à C_{18} ; et C_{22} ;
– $\widetilde{E_{20}}$ correspond à C_{23} ; C_{24} et C_{25}.

Le graphe approché par DBM distingue entre les deux classes $\widetilde{E_{11}}$ et $\widetilde{E_{15}}$ alors qu'on modeC ils sont équivalents suite à des restrictions nouvellement calculées.

Le grapheC de la premiére phase contient 26 classes et 32 arcs, distinguant ainsi les états dans le passé. Nous distinguons maintenant les graphes par la notion

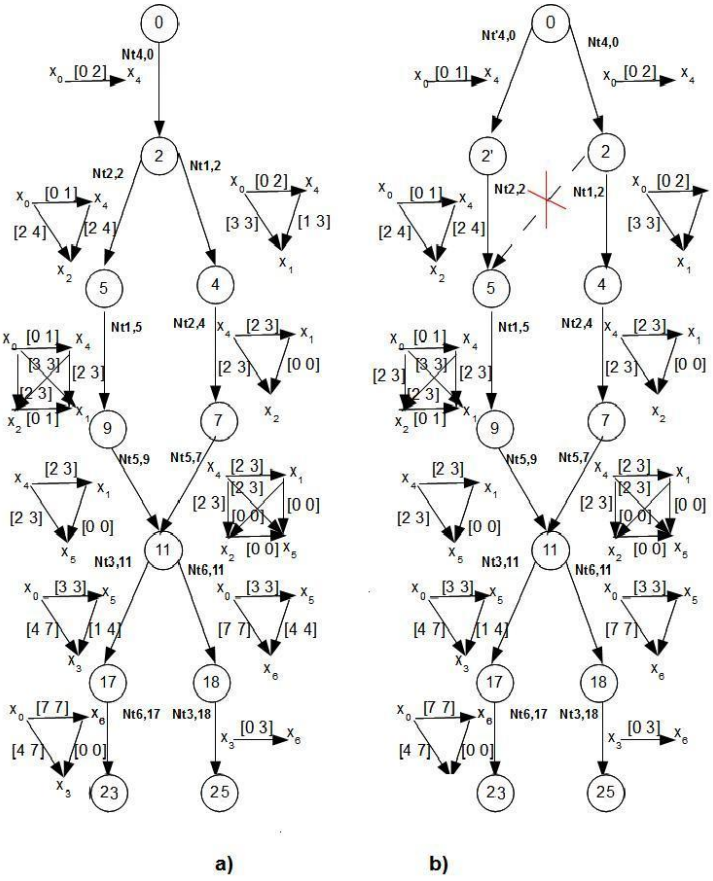


FIGURE 5.13 – Les séquences $\sigma_1; \sigma_2$ de l'$ITPN$ précédent

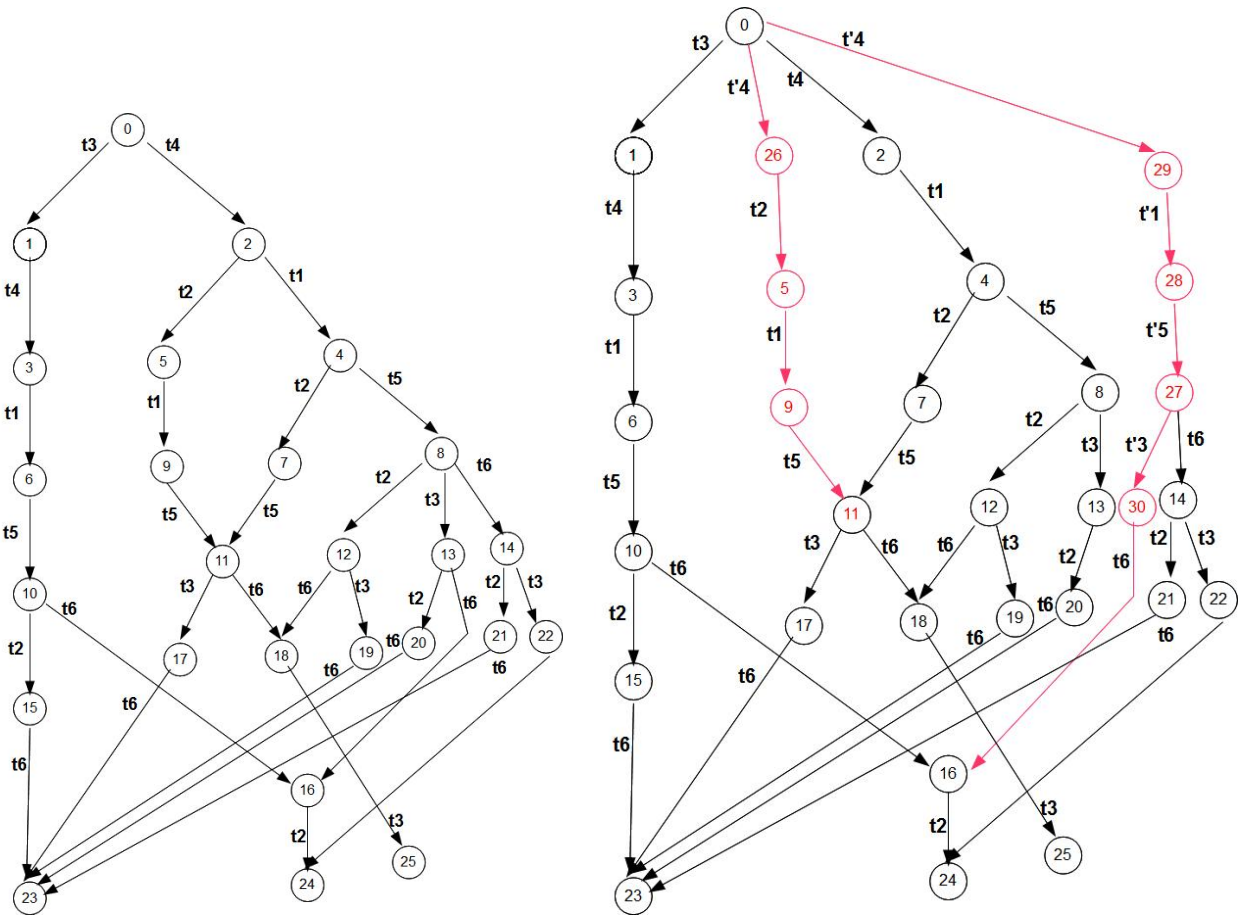


FIGURE 5.14 – Le graphe modeC pour le *ITPN* FIG 2.5 du chapitre 2 : a) Phase 1 b) Phase 2 (classes restreintes)

des chemins déterministes grâce à la notion des classes restreintes. Pour ce faire, nous considérons le graphe finale, phase 2, représenté sur la figure FIG 5.14b). Nous constatons les classes restreintes suivantes :

- C_{26} classe restreinte de C_2 avec la restriction $(x_0, x_4) = [0 \quad 1]$ au lieu de $(x_0, x_4) = [0 \quad 2]$;
- C_{29} classe restreinte de C_2 avec la restriction $(x_0, x_4) = [2 \quad 2]$ au lieu de $(x_0, x_4) = [0 \quad 2]$;
- C_{28} classe restreinte de C_4 ;
- C_{27} classe restreinte de C_8 ;
- C_{30} classe restreinte de C_{13}.

On voit clairement qu'on peut *distinguer les états dans le futur*, en effet, si on

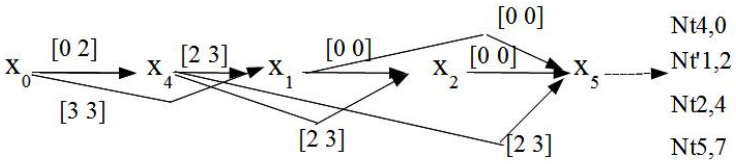


FIGURE 5.15 – Réseau de contraintes pour la séquence σ_1

tire t_4 durant [0 1] (aprés calcul de la classe restreinte C_{26} de C_2, on est sûr de franchir t_2 avant t_1 (voir FIG 5.14b).

Et si l'on tire t_4 durant [2 2] (aprés calcul des classes restreintes C_{29} ; C_{28} ; C_{27}, on peut tirer soit t_6 ou t_3 à partir de la classe 27 classe restreinte de 8, t_2 ou t_3 à partir de la classe 8.

Par rapport aux deux séquences étudiées plus haut, on pourrait dire que si on tire t_4 à une date plus restreinte que [0 2] et qui est [0 1], on est sûr de passer par la séquence σ_2 (en rouge).

Remarque 7 *Le graphe final (phase 2) comporte* 31 *classes et* 41 *arcs permettant de préserver les proprietés CTL*.*

5.2.9 Aspect quantitative du modèle

Comme pour un *RdpT*, une fois le graphe construit, si l'on veut obtenir globalement les contraintes concernant les dates des franchissements de toutes les transitions d'une séquence donnée, il faut concaténer les réseaux Nt correspondants. Ainsi, pour obtenir le graphe des contraintes temporelles de la séquence précédente : $\sigma_1 = (t_4; t_1; t_2; t_5)$ (Fig. 5.14b) il faut concatener $Nt_{4,0}, Nt'_{1,2}, Nt_{2,4}$ et $Nt_{5,7}$. On peut voir que certaines contraintes sont redondantes : x_4 et x_2, x_4 et x_5 (FIG 5.15).

5.3 Outils et Expérimentation

Dans cette section, nous allons présenter quelques outils pour l'analyse des *RdPT* et des *RdPT* étendus à chronométres, avec une étude comparative par rapport aux classes de propriétées préservées.

5.3.1 Outils

Tina : (TIme Network Analysis) [11]) est un environment logiciel permettant l'édition et l'analyse de réseaux de Petri et réseaux de Petri t-temporels, réalisé dans le groupe OLC du LAAS par B. Berthomieu, P.O. Ribet et F. Vernadat. Outre les fonctions classiques d'édition et d'analyse énumérative(graphe de marquage, arbre de couverture) ou structurelle(semi-flots), Tina propose la construction d'espace d'états abstraites permettant la vérification de classes spécifiques de propriétés. Les classes de propriétés proposées incluent : les propriétés générales d'accessibilité (absence de blocage, vivacité), les propriétés spécifiques basées sur la structure linéaire de l'espace d'état concrets (celles exprimables en logique temporelle linéaire(LTL), ou capturées par les équivalences de test) ou sur sa structure arborescente (celles exprimables en logiques temporelles arborescentes(CTL^*), ou capturées par la bisimulation). Un outil offrant plus d'avantages que Romeo dans le sens où il permet d'avoir un ($ASCG$) pour l'analyse les propriétées CTL^*.

Romeo [36], est à présent étendu pour exporter les $RdPT$ vers des outil de AT tel que (UPPAAL et KRONOS). La partie de l'outil avec la capacité de construire différents types de graphes et consacrée à l'analyse des $RdPT$ est notée par Romeo S_{TD} [25].

GraphC [21, 20] décrit le graphe de classes mode C, composé de noeuds et d'arcs reliant un noeud source à un noeud cible. Un noeud du graphe de classes représente une classe d'états caractérisée par son marquage et son réseau de contraintes Nc. Il posséde aussi une étiquette qui est affichée dans les sorties graphique et textuel (l'entrée textuelle est la même que Tina : .net).

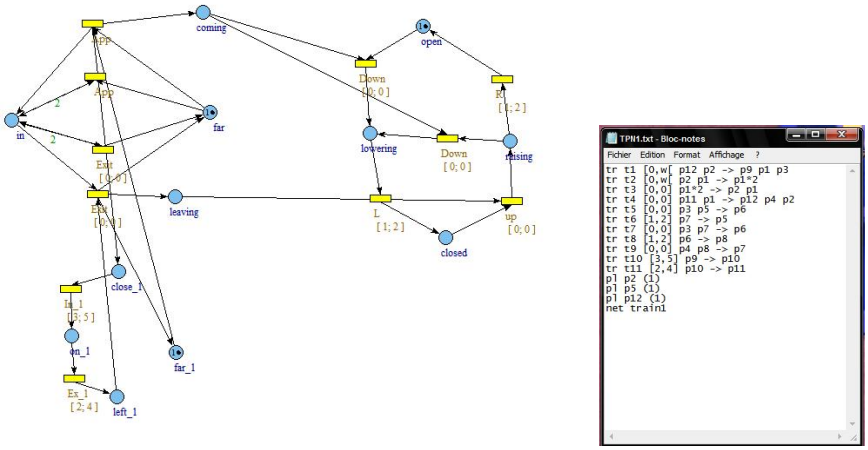
Le tableau suivant est une forme de comparaison entre les outils cités plus-haut, en terme de classes de propriétées (Accéssibilité, LTL, CTL, CTL^*, TCTL) préservées. Les outils UPPAAL et KRONOS sont cités dans la partie basée sur les traductions dans le but d'une vérification de propriétés quantitatives telle que celle exprimées en logique ($TCTL$). Dans ce même but et pour les $RdPT$ avec préemption, tel que les Schedulin-TPN, la méthode est basée sur les traductions de ces derniers vers des automates à chronométres (SWA) [33], basée sur une sur-aproximlation DBM, utilise un format de sortie de HYTECH.

5.3.2 Experimentations

Afin d'illustrer les avantages des méthodes implémentées dans les outils suscités, nous présentons dans les lignes qui suivent un tableau comparatif des résultats obtenus (en termes de temps de calcul et d'utilisation d'espace mémoire)

Propriétés / Outils	Tina	Roméo	GraphC
Accéssibilité	Graphe de Marquage	Graphe de Marquage, calcul à la volée	
LTL	SCG [9, 10]	SCG [9, 10]ou ZBG [37, 38]	
CTL			
CTL^*	ASCG [67, 8, 14]		GraphC [21, 20]
$TCTL$		Trad $TPN \rightarrow TA$ (Uppaal, Kronos)[25, 48]	

FIGURE 5.16 – Graphe de classes d'un TPN selon TINA, ROMEO et GraphC

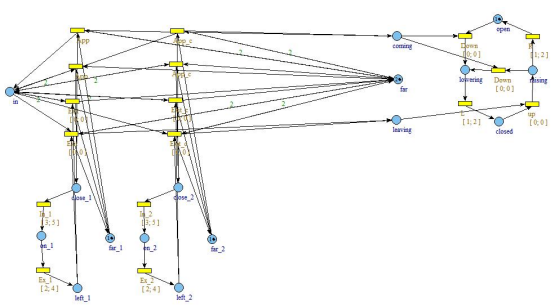


a) Format graphique b) Format textuel

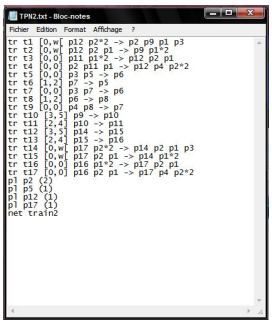
FIGURE 5.17 – "classical level crossing model" avec un train

par les méthodes présentées dans le chapitre 3. Les algorithmes ont été exécutés sur une large variété d'exemples. Les principaux résultats sont résumés dans le tableau 5.20 nous donnons le nombre de noeuds et de transitions du graphe des classes d'états résultant, le temps indiqué est en secondes. Ces calculs ont été éffectués sur un CoreI5 cadencé à 2.34GHZ et possédant 4G de mémoire vive. Les exemples sont tirés respéctivement du chapitre 3 et de [8, 18] représentées en format graphique et textuel sur la figure 5.17 ; 5.18 et 5.19 respectivement, qui représentent le modéle : *classical level crossing* ; avec 1 ; 2 et 3 trains. Les tests affirment bien certaines propriétés et avantages des méthodes décrites dans le chapitre 3.

Il est à noter que les premières colonnes décrivent des graphes basés classes d'état, et les deux qui suivent basés regions et que les graphes alternatives définit par Boucheneb sont marqués $:_b$. Les dernières colonnes décrivent les résultats du graphe modeC avant et après prise en charge des nouvelles classes générées (restreintes), noté modeCr. NA fait référence aux résults qui n'ont pas été obtenu aprés plus de 2H d'attente ou avec des erreurs dues à la taille du graphe généré. En effet, il est à noter que le grapheC est limité, avec une compléxité élevée due au nombres de variables et contraintes associées à chaque classe et arc.

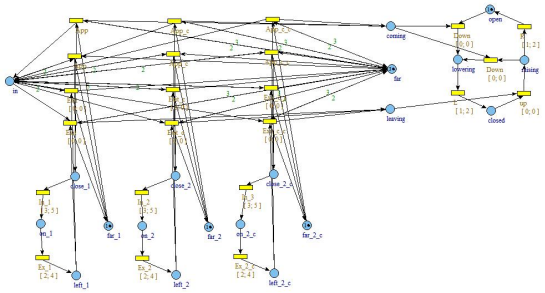


a) Format graphique

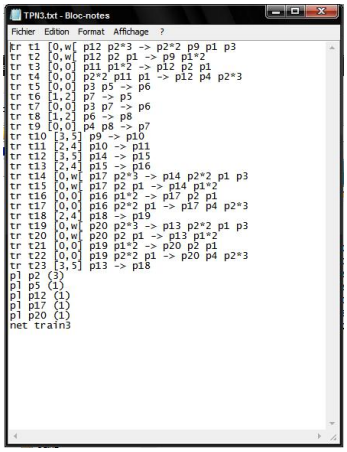


b) Format textuel

FIGURE 5.18 – "classical level crossing model" avec deux trains



a) Format graphique



b) Format textuel

FIGURE 5.19 – "classical level crossing model" avec trois trains

TPN	SCG	$SSCG_b(I)$	$SSCG(LTL)$	$ASCG$	$ASCG_b$	$BZG(I)$	$BZG_b(LTL)$	modeC	modeCr
Chap 3 : classes	13	13	18	16	21	12	14	23	26
arcs	21	21	29	28	37	19	24	35	38
CPUs	0.000s	0.000s	0.000s	0.000s	0.000s	0.000s	0.000s	0.000s	0.000s
train1 : classes	11	10	11	11	12	10	11	15	16
arcs	14	13	14	15	16	13	14	17	18
CPUs	0.000s	0.000s	0.000s	0.000s	0.000s	0.000s	0.000s	0.000s	0.000s
train2 : classes	123	41	141	192	195	30	114	187	NA
arcs	218	82	254	844	849	61	234	297	NA
CPUs	0.000s	0.000s	0.000s	0.031s	0.016s	0.000s	0.000s	1 min	NA
train3 : classes	3101	232	5051	6967	6974	94	2817	NA	NA
arcs	7754	672	13019	49826	49842	269	6944	NA	NA
CPUs	0.047s	0.000s	0.078s	1.404s	1.934ss	0.000s	0.000s	NA	NA

FIGURE 5.20 – Tests expérimentaux d'un TPN selon TINA, ROMEO et GraphC

Enfin, même si la connaissance exacte des contraintes temporelles associées à une séquence n'est pas nécessaire pour répondre si oui ou non, une propriété est vérifiée (pour cela la connaissance de la durée totale est suffisante), par contre elle est absolument nécessaire au concepteur qui souhaiterait ajuster les valeurs d'un ensemble de paramétres pour que la propriété soit vérifiée.

5.4 Conclusion

Dans ce chapitre, nous avons définit une abstarction de l'espace d'états d'un $ITPN$ (modeC étendue à chronométres). Cette dernière permettant la préservation des propriétés CTL^*, mais également les contraintes quantitatives du modèle. Une extension de la méthode du grapheC [21, 20] pour les réseaux de Petri à arcs inhibiteurs en prenant en considération l'activation (réactivation) et l'inhibition des transitions, nous avons comparé les résultats des graphes obtenus avec ceux basées sur la préservation de propriétés LTL de [13], les résultats sont intéressants. Nous avons aussi fait quelques expérimentations sur l'outil Tina, Roméo et graphC afin de faire une comparaison sur les résultats obtenus en pratique.

Chapitre 6

Conclusion Générale

Les réseaux informatiques sont devenus, au cours des trois dernières décennies, omniprésents dans l'industrie. Leur prolifération a fait de leur bon fonctionnement un facteur critique : une défaillance peut désormais avoir des conséquences tant humaines qu'économiques désastreuses. C'est pourquoi il est devenu primordial de vérifier a priori leur bon fonctionnement. Il convient ainsi de développer des formalismes permettant de décrire et de valider les interactions entre les activités des processeurs et les réseaux de communication. Les réseaux de Petri temporels sont un de ces formalismes. Ils permettent de modéliser la durée de différentes actions sous la forme d'un intervalle de temps. Ils peuvent par ailleurs être enrichis, via un modèle baptisé réseau de Petri à chronomètres, afin de représenter des tâches susceptibles d'être suspendues puis relancées.

Cependant, cette approche de vérification repose sur la génération de l'espace d'états, elle est complètement automatique "Model-Cheking" et elle offre une analyse fine et exhaustive. Le problème de l'explosion combinatoire de l'espace d'états est la seule limite de cette approche. En plus, avec la présence des chronomètres l'espace d'états est plus important. Nous avons présenté dans ce mémoire l'ensemble des méthodes qui existent dans la littérature permettant la détermination de l'espace d'états des systèmes temps réel modélisés par un réseau de Petri et celles pour la détermination de l'espace détats des systèmes temps réels préemptifs modélisés par les réseaux de Petri étendus à chronomètre ($ITPN$). Nous avons ensuite proposé une adaptation simple pour le calcul de l'espace d'états des systèmes temps réels préemptifs, préservant une gamme plus importante de propriétés. Dans notre adaptation, le graphe est plus affiné, avec des chemins déterministes. Même si la taille est plus importante (car il comporte plus d'informations) il permet néanmoins la vérification de propriétés de branchement CTL^* et temporelles quantitatives du modèle.

Cependant, même si la connaissance exacte des contraintes temporelles as-

sociées à une séquence n'est pas nécessaire pour répondre si oui ou non, une propriété est vérifiée (pour cela la connaissance de la durée totale est suffisante), par contre elle est absolument nécessaire au concepteur qui souhaiterait ajuster les valeurs d'un ensemble de paramétres pour que la propriété soit vérifiée.

Ce travail pourrait être complété par une optimisation de l'algorithme, celui-ci présentant une compléxité élevée due à la taille de la mémoire du passé (STN). En effet, la taille du graphe est un inconvénient majeur. L'efficacité de la méthode impose un ensemble de chantiers qui restent à explorer dans un futur proche que nous résumons à travers les points suivants :

– Réduire le nombre de contraintes sauvegardées dans le réseau. La majorité d'entre elles sont redondantes dans les classes mères ou filles.

– Remplacer l'algorithme de Floyd Warshall pour le calcul des STN, par un algorithme récursif, qui éliminerait les calculs redondants et introduirait une condition de franchissement simplifiée.

– Définir des relations d'équivalences moins restrictives. permettant de concaténer le graphe (Inclusion).

– Implémenter l'approche dans un environnement optimisé en exploitant des structures et des algorithmes d'exploration adaptés.

Bibliographie

[1] Abdelli Abdelkrim. Efficient computation of quantitative properties of real time preemptive systems. *in International journal of critical computer based systems*, 3(3) :187–209, 2012.

[2] Abdelli Abdelkrim. Improving the construction of the dbm over approximation of the state space of real-time preemptive systems. *Acta Cybernetica*, 20(3) :347–384, 2012.

[3] A. Abdelli. Optimisation de la construction d'une approximation de l'espace d'etat des systémes préemptifs. *In Journal Technique et Sciences Informatiques*, 28 :9 :1143–1170, 2009.

[4] Hamdani Abdia. Extention de l'outil : GraphC pour les réseaux de Petri t-temporels préservant les contraintes temporelles floues. Master Recherche (M2R), LAAS, University of Toulouse, France, 2006.

[5] R. Alu and D.L. Dill. A theory for timed automata. *Theoretical Computer Science*, 126(2) :183–235, 1994.

[6] R. Alur, C. Courcoubetis, and D.L. Dill. Model-checking for real-time systems. In *In 5th IEEE Symposium on Logic in Computer*, pages 414–425. IEEE Computer Society Press, 1990.

[7] D. Avis and S. Picozzi. On canonical representation of convex polyedra. In X.-S. Gao A. M. Cohen and N. Takayama editors, editors, *The first International Congress of Mathematical Soft-ware*, pages 350–360. Mathematical Soft-ware, 2002.

[8] Berthomieu Bernard and Vernadat François. State class constructions for branching analysis of time petri nets. In *Proceedings of TACAS'2003*, volume 9, pages 442–457, Warsaw, Poland, 2003. Springer Verlag LNCS 2619.

[9] Berthomieu Bernard and Menasche Miguel. An enumerative approach for analyzing time petri net. In *IFIP Congress Series*, 1983.

[10] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time petri nets. *IEEE Transactions on Software Engineering*, 17(3) :259–273, 1991.

[11] B. Berthomieu, P.-O. Ribet, and F. Vernadat. The tool tina – construction of abstract state spaces for Petri nets and time petri nets. *International Journal of Production Research*, 42(14), 2004.

[12] Bernard Berthomieu, Didier Lime, Olivier (H.) Roux, and Francois Vernadat. Problèmes d'accessibilité et espaces d'états abstraits des réseaux de Petri temporels à chronomètres. In *5ieme Colloque Francophone sur la Modélisation des Systèmes Réactifs, (MSR'05)*, Grenoble, France, October 2005.

[13] Bernard Berthomieu, Didier Lime, Olivier H. Roux, and Francois Vernadat. Reachability problems and abstract state spaces for time Petri nets with stopwatches. *Journal of Discrete Event Dynamic Systems - Theory and Applications (DEDS)*, 17(2) :133–158, 2007. Copyright Springer.

[14] H. Boucheneb and R. Hadjidji. Towards optimal CTL* model checking of time petri nets. In *Proc. of the International Workshop on Discrete Event Systems (WODES)*, 2004.

[15] H. Boucheneb and R. Hadjidji. Using inclusion abstraction to construct atomic state class graphs for time petri nets. *International journal of embedded systems*, 2/1-2 :128–139, 2005.

[16] H. Boucheneb and R. Hadjidji. CTL* model checking of time petri nets. *Theoretical Computer Science*, 353/1-3 :208–227, 2006.

[17] H. Boucheneb and J. Mullins. Analyse des réseaux temporels : Calcul des classes en $o(n^2)$ et des temps de chemin en $o(m \times n)$. *Revue des sciences et technologies de l'information*, 22/4 :435–459, 2003.

[18] Hanifa Boucheneb, Guillaume Gardey, and Olivier H. Roux. TCTL model checking of time Petri nets. *Journal of Logic and Computation*, 19(6) :1509–1540, December 2009. Copyright Oxford Press.

[19] G. Bucci, A. Fedeli, L. Sassoli, and E. Vicario. Timed state space analysis of real-time preemptive systems. *IEEE Transactions on Software Engineering*, 30(2) :97–111, February 2004.

[20] J. Cardoso, X. Mao, and R. Valette. A graph of classes preserving quantitatives temporal constraints considering unbounded transitions. In *PMCCS'05, 7th International Workshop on performability Modeling of computer and Communication Systems*, 2005.

[21] J. Cardoso, X. Mao, and R. Valette. Un nouveau graphe de classes pour la préservation des contraintes temporelles quantitatives. In *MSR'05, Modélisation des Systèmes Réactifs*, 5 2005.

[22] Janette. Cardoso. *Les Aspects Temporels Qualitatifs et Quantitatifs dans les Systémes embarqués*. Habilitation à Diriger les Recherches (HDR), LAAS-CNRS, Université Toulouse 1, France, 2007.

[23] Franck Cassez and Kim Larsen. The impressive power of stopwatches. In *In Proc. of CONCUR 2000 : Concurrency Theory*, pages 138–152. Springer, 1999.

[24] Franck Cassez and Olivier (H.) Roux. Traduction structurelle des réseaux de Petri temporels vers les automates temporisés. In *4ieme Colloque Francophone sur la Modélisation des Systèmes Réactifs, (MSR'03)*, Metz, France, October 2003. Copyright Hermes-Science.

[25] Franck Cassez and Olivier H. Roux. Structural translation from Time Petri Nets to Timed Automata – Model-Checking Time Petri Nets via Timed Automata. *The journal of Systems and Software*, 79(10) :1456–1468, 2006. Copyright Elsevier.

[26] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8 :244–263, 1986.

[27] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons from branching time temporal logic. In *In Proceedings of the Workshop on Logics of Programs*, Lecture Notes in Computer Science, pages 52–71. Springer Verlag, May 1981.

[28] G.B. Dantzig. Linear programming and extensions. *IEICE Trans. inf. Syst*, 1963.

[29] de Frutos Escrig D., Ruiz V.V., and O.M. Alonso. Decidability of properties of timedarc petri nets. In *In : 21st International Conference on Application and Theory of Petri Nets (ICATPN00)*, volume 1825 of *Lecture Notes in Computer Science*, page 187206, Aarhus,Denmark, 2000. Springer-Verlag.

[30] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49 :61–91, 1991.

[31] Oddoux Denis. *Utilisation des Automates Alternants pour un Model-Checking Efficace des Logiques Temporelles Linéraires.* Phd thesis, UFR d'Informatique, Université Paris 7, december, 2003.

[32] M. Diaz and P. Senac. Time stream Petri nets : a model for timed multimedia information. *Lecture Notes in Computer Science*, 815 :219–238, 1994.

[33] Lime Didier. *Vérification d'Applications Temps Réel à l'aide de Réseaux de Petri Temporels étendus.* Phd thesis, IRCCyN, University of Nantes, France, 2004.

[34] D.L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Workshop of Automatic Verification Methods for Finite-State Systems*, volume 407, pages 197–212, 1989.

[35] Robert Warshall Floyd. Algorithm 97 : Shortest path. *Communications of the ACM*, 5(6) :345, 1962.

[36] Guillaume Gardey, Didier Lime, Morgan Magnin, and Olivier (h. Roux. Roméo : A tool for analyzing time petri nets. In *In Proc. CAV05, vol. 3576 of LNCS*, pages 418–423. Springer, 2005.

[37] Guillaume Gardey, Olivier H. Roux, and Olivier (F.) Roux. A zone-based method for computing the state space of a time Petri net. In *Formal Modeling and Analysis of Timed Systems, (FORMATS'03)*, volume 2791 of *Lecture Notes in Computer Science*, pages 246–259, Marseille, France, September 2003. Springer. Copyright Springer-Verlag.

[38] Guillaume Gardey, Olivier (H). Roux, and Olivier (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*, 6(3) :301–320, 2006. Copyright Cambridge Press.

[39] R. Hadjidj and H. Boucheneb. On-the-fly tctl model checking for time petri nets using the state class method. In *1In Proc of the Sixth International Conference on Application of Concurrency to System Design (ACSD'06)*, pages 111–120. IEEE Computer Society Press, 2006.

[40] R. Hadjidj and H. Boucheneb. Zenoness detection and timed model checking for real time systems. In *1st international workshop of Verification, Evaluation of Computer and Communication Systems (VECos'07)*, British Computer Society, 2007.

[41] R. Hadjidji. *Analyse et validation Formelle des Systémes temps réel.* Phd thesis, Ecole polytechnique de Montréal, Université de Montréal, February, 2006.

[42] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2) :193244, 1994.

[43] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10) :576–580, 1969.

[44] N.D. Jones, L.H. Landweber, and Y.E. Lien. Complexity of some problems in Petri nets. *Theoretical Computer Science*, pages 277–299, 1977.

[45] J.P. Khansa, W.and Denat and S. Collart-Dutilleul. P-time petri nets for manufacturing systems. In *In : International Workshop on Discrete Event Systems, WODES96*, page 94102, Edinburgh (U.K.), 1996.

[46] J. Lilius. Efficient state space search for time petri nets. In *MFCS Workshop on Concurrency '98*, volume 18, pages 46–57. ENTCS. Elsevier, 1999.

[47] Didier Lime and Olivier H. Roux. Expressiveness and analysis of scheduling extended time Petri nets. In *5th IFAC International Conference on Fieldbus Systems and their Applications, (FET'03)*, pages 193–202, Aveiro, Portugal, July 2003. Elsevier Science. Copyright Elsevier Science.

[48] Didier Lime and Olivier H. Roux. Model checking of time Petri nets using the state class timed automaton. *Journal of Discrete Events Dynamic Systems - Theory and Applications (DEDS)*, 16(2) :179–205, 2006. Copyright Springer-Kluwer.

[49] Didier Lime and Olivier H. Roux. Formal verification of real-time systems with preemptive scheduling. *Journal of Real-Time Systems*, 41(2) :118–151, 2009. Copyright Springer.

[50] Morgan Magnin. *Réseaux de Petri à chronomètres ; Temps dense et temps discret*. Phd thesis, IRCCyN, University of Nantes, France, 2007.

[51] Morgan Magnin, Didier Lime, and Olivier H. Roux. An efficient method for computing exact state space of Petri nets with stopwatches. In *third International Workshop on Software Model-Checking (SoftMC'05)*, volume 144 of *Electronic Notes in Theoretical Computer Science*, pages 59–77, Edinburgh, Scotland, UK, July 2005. Elsevier.

[52] P. M. Merlin. *A Study of the Recoverability of Computing Systems*. Phd thesis, Department of Information and Computer Science, University of California, Irvine, CA, 1974.

[53] P. PAIGE and R.E. TARJAN. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6) :973–989, 1987.

[54] W. Penczek and A. Polrola. Abstraction and partial order reductions for checking branching properties of time petri nets. In *22st International Conference on Application and Theory of Petri Nets (ICATPN 2001)*, LNCS 2075, pages 323–342. Springer, 2001.

[55] C.A. Petri. *Communication with automata :Kommunikation mit Automaten*. Phd thesis, Darmstadt University of Technology, Bonn, 1962.

[56] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symp. on Foundations of Computer Science (FOCS'77)*, pages 46–57, 1977.

[57] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in cesar. In *In Proc. 5th Int. Symp. on Programming, Lecture Notes in Computer Science*, pages 337–351, 1982.

[58] C. Ramchandani. *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets*. Phd thesis, Massachusetts Institute of Technology, Cambridge, MA, 1974.

[59] Olivier (H.) Roux and Anne-Marie Déplanche. Extension des réseaux de Petri t-temporels pour la modélisation de l'ordonnancement de tâches temps-réel. In *3ieme Colloque Francophone sur la Modélisation des Systèmes Réactifs, (MSR'01)*, pages 327–342, Toulouse, France, October 2001. Hermes Science. Copyright Hermes-Science.

[60] Olivier H. Roux and Didier Lime. Time Petri nets with inhibitor hyperarcs. Formal semantics and state space computation. In *The 25th International Conference on Application and Theory of Petri Nets, (ICATPN'04)*, volume 3099 of *Lecture Notes in Computer Science*, pages 371–390, Bologna, Italy, June 2004. Springer.

[61] Henzinger Thomas. The theory of hybrid automata. In *In Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 96)*, volume 407, page 278292, New Brunswick, New Jersey, 1996. British Computer Society.

[62] Joel. Toussaint, Francoise. Simonot-Lion, and Jean-Pierre. Thomesse. Time constraint verifications methods based time petri nets. In *In 6th Workshop on Future Trends in Distributed Computing Systems (FTDCS97)*, page 262267, Tunis, Tunisia, 1997.

[63] M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *n Symposium on Logic in Computer Science (LICS'86)*, volume 3099 of *Lecture Notes in Computer Science*, pages 332–345, Washington, D.C., USA, 1986. IEEE Computer Society Press.

[64] Enrico Vicario. Static analysis and dynamic steering of time-dependent systems. *IEEE Trans. Software Eng*, 27(8) :728–748, 2001.

[65] Stephen Warshall. A theorem on boolean matrices. *Journal of the ACM*, pages 11–12, 1962.

[66] Wikipedia. Multitâche préemptif, the free encyclopedia, Novembre 2011.

[67] T. Yoneda and H. Ryuba. CTL model checking of time petri nets using geometric regions. *IEICE Trans. inf. Syst*, E81-D(3) :297–396, 1998.

Annexe A

Logique temporelle

A.1 logique temporelle Qualitative :

Définition 34 *(CTL* [56]) Soit* $\{ap_1, ..., ap_n\}$ *un ensemble de variables propositionnelles. Le langage CTL* est l'ensemble des formules d'états* φ_s *exprimées par des formules de chemins* φ_p *défini inductivement par la grammaire :*

$$\varphi_s := true \,|ap \mid \neg ap \,|\varphi_s \wedge \varphi_s \mid \varphi_s \vee \varphi_s|\forall\varphi_s \mid \exists\varphi_s$$

$$\varphi_p := true \,|\varphi_p \,|\varphi_p \wedge \varphi_p \mid \varphi_p \vee \varphi_p| \bigcirc \varphi_p \mid \varphi_p U \varphi_p$$

La formule $\varphi_p U \psi_p$ exprime la propriéte : ψ_p est vérifiée à partir d'un certain état et tous les états intermédiaires vérifient la propriété φ_p.
De maniére classique, les notations suivantes sont utilisées pour écrire des formules CTL courantes :
false= $\neg$true ; $\exists\Diamond\varphi = \exists \text{true} U\varphi$; $\forall\Diamond\varphi = \forall$ true $U\varphi$;$\exists\Box\varphi = \neg\forall\Diamond\neg\varphi$; $\forall\Box\varphi = \neg\exists\Diamond\neg\varphi$.

Plusieurs sous-classes de CTL* ont été proposées :

– CTL (Computation Tree Logic) est le fragment de CTL* constitué des combinaisons booléennes de :$\forall\varphi$ Uψ ; $\forall\varphi$ Rψ ; $\forall \bigcirc \varphi$; $\exists\varphi$ Uψ et $\exists \bigcirc \varphi$.

– ACTL (Universal Computation Tree Logic) est le fragment de CTL* constitué des combinaisons booléennes de : $\forall\varphi$ Uψ ; $\forall \bigcirc \varphi$.

– ECTL (Existential Computation Tree Logic) est le fragment de CTL* constitué des combinaisons booléennes de : $\exists\varphi$ Uψ et $\exists \bigcirc \varphi$.

– LTL (Linear Time Logic) est le fragment de CTL* constitué des formules de type $\forall\varphi$ où φ ne contient pas de quantificateurs $\forall$ ou $\exists$.

A.2 logique temporelle Quantitative :

Définition 35 *(TCTL : Time+ CTL). La logique TCTL est définie par la grammaire :*

$$\varphi := ap \mid \neg ap | \varphi \wedge \varphi \mid \varphi \vee \varphi | \forall \varphi U_I \varphi \mid \exists \varphi U_I \varphi$$

Où ap est une proposition atomique et I un intervalle de $\mathbb{R}^+$ avec bornes entiéres de la forme $[n, m], [n, m[,]n, m],]n, m[, ou [m, \infty[$ avec $n, m \in \mathbb{N}$.

De la même maniére que pour CTL, les notations suivantes sont utilisées couramment : false$= \neg$true ; $\exists \Diamond_I \varphi = \exists \text{true} U_I \varphi$; $\forall \Diamond_I \varphi = \forall$ true $U_I \varphi$;$\exists \Box_I \varphi = \neg \forall \Diamond_I \neg \varphi$; $\forall \Box_I \varphi = \neg \exists \Diamond_I \neg \varphi$.

Annexe B

Graphes de classes d'un TPN

$E_0 = \left\{ \begin{array}{l} M_0 : p_1, p_2, p_3 \to 1 \\ D_0 : \left\{ \begin{array}{l} 0 \leq \underline{t_1} \leq 3 \\ 0 \leq \underline{t_2} \leq 0 \\ 0 \leq \underline{t_3} \leq 3 \end{array} \right. \end{array} \right.$	$E_1 = \left\{ \begin{array}{l} M_1 : p_2, p_3, p_4 \to 1 \\ D_1 : \left\{ \begin{array}{l} 0 \leq \underline{t_2} \leq 0 \\ 0 \leq \underline{t_3} \leq 3 \end{array} \right. \end{array} \right.$	$E_2 = \left\{ \begin{array}{l} M_2 : p_1, p_3, p_5 \to 1 \\ D_2 : \left\{ \begin{array}{l} 0 \leq \underline{t_1} \leq 3 \\ 0 \leq \underline{t_3} \leq 3 \end{array} \right. \end{array} \right.$
$E_3 = \left\{ \begin{array}{l} M_3 : p_1, p_2, p_6 \to 1 \\ D_3 : \left\{ \begin{array}{l} 0 \leq \underline{t_1} \leq 3 \\ 0 \leq \underline{t_2} \leq 0 \end{array} \right. \end{array} \right.$	$E_4 = \left\{ \begin{array}{l} M_4 : p_3, p_4, p_5 \to 1 \\ D_4 : \left\{ \begin{array}{l} 0 \leq \underline{t_3} \leq 3 \\ 1 \leq \underline{t_4} \leq 2 \end{array} \right. \end{array} \right.$	$E_5 = \left\{ \begin{array}{l} M_5 : p_2, p_4, p_6 \to 1 \\ D_5 : \left\{ \begin{array}{l} 0 \leq \underline{t_2} \leq 0 \end{array} \right. \end{array} \right.$
$E_6 = \left\{ \begin{array}{l} M_6 : p_1, p_5, p_6 \to 1 \\ D_6 : \left\{ \begin{array}{l} 0 \leq \underline{t_1} \leq 3 \\ 0 \leq \underline{t_5} \leq 2 \end{array} \right. \end{array} \right.$	$E_7 = \left\{ \begin{array}{l} M_7 : p_4, p_5, p_6 \to 1 \\ D_7 : \left\{ \begin{array}{l} 0 \leq \underline{t_4} \leq 2 \\ 0 \leq \underline{t_5} \leq 2 \end{array} \right. \end{array} \right.$	$E_8 = \left\{ \begin{array}{l} M_8 : p_3, p_7 \to 1 \\ D_8 : \left\{ \begin{array}{l} 0 \leq \underline{t_3} \leq 2 \end{array} \right. \end{array} \right.$
$E_9 = \left\{ \begin{array}{l} M_9 : p_4, p_5, p_6 \to 1 \\ D_9 : \left\{ \begin{array}{l} 1 \leq \underline{t_4} \leq 2 \\ 0 \leq \underline{t_5} \leq 2 \end{array} \right. \end{array} \right.$	$E_{10} = \left\{ \begin{array}{l} M_{10} : p_1, p_7 \to 1 \\ D_{10} : \left\{ \begin{array}{l} 0 \leq \underline{t_1} \leq 3 \end{array} \right. \end{array} \right.$	$E_{11} = \left\{ \begin{array}{l} M_{11} : p_6, p_7 \to 1 \\ D_{11} : \left\{ \begin{array}{l} \emptyset \end{array} \right. \end{array} \right.$
$E_{12} = \left\{ \begin{array}{l} M_{12} : p_4, p_7 \to 1 \\ D_{12} : \left\{ \begin{array}{l} 0 \leq \underline{t_6} \leq 2 \end{array} \right. \end{array} \right.$		

FIGURE B.1 – L'ensemble des classes du graphe GR Fig 3.2b)
(Préservant marquage et propriétés LTL)

$S_0 = \left\{ \begin{array}{l} M_0 : p_1, p_2, p_3 \to 1 \\ Q_0 : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_1} \leq 0 \\ 0 \leq \underline{\gamma_2} \leq 0 \\ 0 \leq \underline{\gamma_3} \leq 0 \end{array} \right. \end{array} \right.$	$S_1 = \left\{ \begin{array}{l} M_1 : p_2, p_3, p_4 \to 1 \\ Q_1 : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_2} \leq 0 \\ 0 \leq \underline{\gamma_3} \leq 0 \end{array} \right. \end{array} \right.$	$S_2 = \left\{ \begin{array}{l} M_2 : p_1, p_3, p_5 \to 1 \\ Q_2 : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_1} \leq 0 \\ 0 \leq \underline{\gamma_3} \leq 0 \end{array} \right. \end{array} \right.$
$S_3 = \left\{ \begin{array}{l} M_3 : p_1, p_2, p_6 \to 1 \\ D_3 : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_1} \leq 0 \\ 0 \leq \underline{\gamma_2} \leq 0 \end{array} \right. \end{array} \right.$	$S_4 = \left\{ \begin{array}{l} M_4 : p_3, p_4, p_5 \to 1 \\ Q_4 : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_3} \leq 0 \\ 0 \leq \underline{\gamma_4} \leq 0 \end{array} \right. \end{array} \right.$	$S_5 = \left\{ \begin{array}{l} M_5 : p_2, p_4, p_6 \to 1 \\ Q_5 : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_2} \leq 0 \end{array} \right. \end{array} \right.$
$S_6 = \left\{ \begin{array}{l} M_6 : p_3, p_4, p_5 \to 1 \\ Q_6 : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_3} \leq 3 \\ 0 \leq \underline{\gamma_4} \leq 4 \end{array} \right. \end{array} \right.$	$S_7 = \left\{ \begin{array}{l} M_7 : p_1, p_5, p_6 \to 1 \\ Q_7 : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_1} \leq 3 \\ 0 \leq \underline{\gamma_5} \leq 0 \end{array} \right. \end{array} \right.$	$S_8 = \left\{ \begin{array}{l} M_8 : p_1, p_5, p_6 \to 1 \\ Q_8 : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_1} \leq 0 \\ 0 \leq \underline{\gamma_5} \leq 0 \end{array} \right. \end{array} \right.$
$S_9 = \left\{ \begin{array}{l} M_9 : p_4, p_5, p_6 \to 1 \\ Q_9 : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_4} \leq 2 \\ 0 \leq \underline{\gamma_5} \leq 0 \end{array} \right. \end{array} \right.$	$S_{10} = \left\{ \begin{array}{l} M_{10} : p_3, p_7 \to 1 \\ Q_{10} : \left\{ \begin{array}{l} 1 \leq \underline{\gamma_3} \leq 2 \end{array} \right. \end{array} \right.$	$S_{11} = \left\{ \begin{array}{l} M_{11} : p_4, p_5, p_6 \to 1 \\ Q_{11} : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_4} \leq 0 \\ 0 \leq \underline{\gamma_5} \leq 0 \end{array} \right. \end{array} \right.$
$S_{12} = \left\{ \begin{array}{l} M_{12} : p_3, p_7 \to 1 \\ Q_{12} : \left\{ \begin{array}{l} 1 \leq \underline{\gamma_3} \leq 3 \end{array} \right. \end{array} \right.$	$S_{13} = \left\{ \begin{array}{l} M_{13} : p_4, p_5, p_6 \to 1 \\ Q_{13} : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_4} \leq 0 \\ 0 \leq \underline{\gamma_5} \leq 2 \end{array} \right. \end{array} \right.$	$S_{14} = \left\{ \begin{array}{l} M_{14} : p_1, p_7 \to 1 \\ Q_{14} : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_1} \leq 3 \end{array} \right. \end{array} \right.$
$S_{15} = \left\{ \begin{array}{l} M_{15} : p_1, p_7 \to 1 \\ Q_{15} : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_1} \leq 2 \end{array} \right. \end{array} \right.$	$S_{16} = \left\{ \begin{array}{l} M_{16} : p_4, p_7 \to 1 \\ Q_{16} : \left\{ \begin{array}{l} \emptyset \end{array} \right. \end{array} \right.$	$S_{17} = \left\{ \begin{array}{l} M_{17} : p_4, p_7 \to 1 \\ Q_{17} : \left\{ \begin{array}{l} 0 \leq \underline{\gamma_6} \leq 0 \end{array} \right. \end{array} \right.$

FIGURE B.2 – L'ensemble des classes du graphe GR_s Fig 3.5b)
(Préservant états et propriétés LTL)

$A_0 = \left\{ \begin{array}{l} M_0 : p_1, p_2, p_3 \to 1 \\ Q_0 : \left\{ \begin{array}{l} 0 \le \underline{\gamma_1} \le 0 \\ 0 \le \underline{\gamma_2} \le 0 \\ 0 \le \underline{\gamma_3} \le 0 \end{array} \right. \end{array} \right.$	$A_1 = \left\{ \begin{array}{l} M_1 : p_2, p_3, p_4 \to 1 \\ Q_1 : \left\{ \begin{array}{l} 0 \le \underline{\gamma_2} \le 0 \\ 0 \le \underline{\gamma_3} \le 0 \end{array} \right. \end{array} \right.$	$A_2 = \left\{ \begin{array}{l} M_2 : p_1, p_3, p_5 \to 1 \\ Q_2 : \left\{ \begin{array}{l} 0 \le \underline{\gamma_1} \le 0 \\ 0 \le \underline{\gamma_3} \le 0 \end{array} \right. \end{array} \right.$
$A_3 = \left\{ \begin{array}{l} M_3 : p_1, p_2, p_6 \to 1 \\ Q_3 : \left\{ \begin{array}{l} 0 \le \underline{\gamma_1} \le 0 \\ 0 \le \underline{\gamma_2} \le 0 \end{array} \right. \end{array} \right.$	$A_4 = \left\{ \begin{array}{l} M_4 : p_3, p_4, p_5 \to 1 \\ Q_4 : \left\{ \begin{array}{l} 0 \le \underline{\gamma_3} \le 3 \\ 0 \le \underline{\gamma_4} \le 0 \end{array} \right. \end{array} \right.$	$A_5 = \left\{ \begin{array}{l} M_5 : p_2, p_4, p_6 \to 1 \\ Q_5 : \{ \ 0 \le \underline{\gamma_2} \le 0 \end{array} \right.$
$A_6 = \left\{ \begin{array}{l} M_6 : p_3, p_4, p_5 \to 1 \\ Q_6 : \left\{ \begin{array}{l} 0 \le \underline{\gamma_3} \le 2 \\ 0 \le \underline{\gamma_4} \le 0 \end{array} \right. \end{array} \right.$	$A_7 = \left\{ \begin{array}{l} M_7 : p_3, p_4, p_5 \to 1 \\ Q_7 : \left\{ \begin{array}{l} 2 \le \underline{\gamma_3} \le 3 \\ 0 \le \underline{\gamma_4} \le 0 \end{array} \right. \end{array} \right.$	$A_8 = \left\{ \begin{array}{l} M_8 : p_1, p_5, p_6 \to 1 \\ Q_8 : \left\{ \begin{array}{l} 0 \le \underline{\gamma_1} \le 2 \\ 0 \le \underline{\gamma_5} \le 0 \end{array} \right. \end{array} \right.$
$A_9 = \left\{ \begin{array}{l} M_9 : p_1, p_5, p_6 \to 1 \\ Q_9 : \left\{ \begin{array}{l} 2 \le \underline{\gamma_1} \le 3 \\ 0 \le \underline{\gamma_5} \le 0 \end{array} \right. \end{array} \right.$	$A_{10} = \left\{ \begin{array}{l} M_{10} : p_1, p_5, p_6 \to 1 \\ Q_{10} : \left\{ \begin{array}{l} 0 \le \underline{\gamma_1} \le 0 \\ 0 \le \underline{\gamma_5} \le 0 \end{array} \right. \end{array} \right.$	$A_{11} = \left\{ \begin{array}{l} M_{11} : p_4, p_5, p_6 \to 1 \\ Q_{11} : \left\{ \begin{array}{l} 0 \le \underline{\gamma_4} \le 2 \\ 0 \le \underline{\gamma_5} \le 0 \end{array} \right. \end{array} \right.$
$A_{12} = \left\{ \begin{array}{l} M_{12} : p_3, p_7 \to 1 \\ Q_{12} : \{ \ 1 \le \underline{\gamma_1} \le 2 \end{array} \right.$	$A_{13} = \left\{ \begin{array}{l} M_{13} : p_4, p_5, p_6 \to 1 \\ Q_{13} : \left\{ \begin{array}{l} 0 \le \underline{\gamma_4} \le 0 \\ 0 \le \underline{\gamma_5} \le 0 \end{array} \right. \end{array} \right.$	$A_{14} = \left\{ \begin{array}{l} M_{14} : p_3, p_7 \to 1 \\ Q_{14} : \{ \ 1 \le \underline{\gamma_3} \le 3 \end{array} \right.$
$A_{15} = \left\{ \begin{array}{l} M_{15} : p_4, p_5, p_6 \to 1 \\ Q_{15} : \left\{ \begin{array}{l} 0 \le \underline{\gamma_4} \le 0 \\ 0 \le \underline{\gamma_5} \le 1 \end{array} \right. \end{array} \right.$	$A_{16} = \left\{ \begin{array}{l} M_{16} : p_4, p_5, p_6 \to 1 \\ Q_{16} : \left\{ \begin{array}{l} 0 \le \underline{\gamma_4} \le 0 \\ 1 \le \underline{\gamma_5} \le 2 \end{array} \right. \end{array} \right.$	$A_{17} = \left\{ \begin{array}{l} M_{17} : p_1, p_7 \to 1 \\ Q_{17} : \{ \ 0 \le \underline{\gamma_1} \le 3 \end{array} \right.$
$A_{18} = \left\{ \begin{array}{l} M_{18} : p_1, p_7 \to 1 \\ Q_{18} : \{ \ 0 \le \underline{\gamma_1} \le 2 \end{array} \right.$	$A_{19} = \left\{ \begin{array}{l} M_{19} : p_6, p_7 \to 1 \\ Q_{19} : \{ \ \emptyset \end{array} \right.$	$A_{20} = \left\{ \begin{array}{l} M_{20} : p_4, p_7 \to 1 \\ Q_{20} : \{ \ 0 \le \underline{\gamma_6} \le 0 \end{array} \right.$

FIGURE B.3 – l'ensemble des classes du graphe GR_a fig 3.6b) (Préservant propriétés CTL^*)

Class	Marking	Constraints Nc
$\mathcal{C}_0$	$p_1p_2p_3$	x_0
$\mathcal{C}_1$	$p_2p_3p_4$	$0 \le x_1 - x_0 \le 0$
$\mathcal{C}_2$	$p_1p_3p_5$	$0 \le x_2 - x_0 \le 0$
$\mathcal{C}_3$	$p_1p_2p_6$	$0 \le x_3 - x_0 \le 0$
$\mathcal{C}_4$	$p_3p_4p_5$	$0 \le x_2 - x_0 \le 0$
$\mathcal{C}_5$	$p_2p_4p_6$	$0 \le x_3 - x_0 \le 0$
$\mathcal{C}_6$	$p_3p_4p_5$	$0 \le x_1 - x_0 \le 3$
$\mathcal{C}_7$	$p_1p_5p_6$	$0 \le x_3 - x_0 \le 3$
$\mathcal{C}_8$	$p_2p_4p_6$	$0 \le x_1 - x_0 \le 0$
$\mathcal{C}_9$	$p_1p_5p_6$	$0 \le x_2 - x_0 \le 0$
$\mathcal{C}_{10}$	$p_4p_5p_6$	$0 \le x_3 - x_2 \le 2$
$\mathcal{C}_{11}$	p_3p_7	$1 \le x_4 - x_0 \le 2$
$\mathcal{C}_{12}$	$p_4p_5p_6$	x_2
$\mathcal{C}_{13}$	$p_4p_5p_6$	$0 \le x_3 - x_1 \le 2$
$\mathcal{C}_{14}$	p_3p_7	$1 \le x_4 - x_0 \le 3$
$\mathcal{C}_{15}$	$p_4p_5p_6$	$0 \le x_1 - x_3 \le 2$
$\mathcal{C}_{16}$	p_1p_7	$0 \le x_5 - x_0 \le 3$
$\mathcal{C}_{17}$	$p_4p_5p_6$	$0 \le x_1 - x_2 \le 2$
$\mathcal{C}_{18}$	p_1p_7	$0 \le x_5 - x_0 \le 2$
$\mathcal{C}_{19}$	p_6p_7	x_4
$\mathcal{C}_{20}$	p_4p_7	x_5
$\mathcal{C}_{21}$	p_6p_7	x_3
$\mathcal{C}_{22}$	p_4p_7	x_1
$\mathcal{C}_{23}$	$p_3p_4p_5$	$0 \le x_1 - x_0 \le 2$
$\mathcal{C}_{24}$	$p_4p_5p_6$	$0 \le x_1 - x_3 \le 1$
$\mathcal{C}_{25}$	$p_4p_5p_6$	$0 \le x_1 - x_2 \le 1$

Réseaux temporels $Nt_{i,k}$ de t_i à partir de $\mathcal{C}_k$

$Nt_{1,0}$, (x_0, x_1)=[0 0]
$Nt_{2,0}$, (x_0, x_2)=[0 0]
$Nt_{3,0}$, (x_0, x_3)=[0 0]
$Nt_{2,1}$, (x_0, x_1)=(x_0, x_2)=(x_1, x_2)=[0 0]
$Nt_{3,1}$, (x_0, x_1)=(x_0, x_3)=(x_1, x_3)=[0 0]
$Nt_{1,2}$, (x_0, x_2)=[0 0],(x_0, x_1)=(x_2, x_1)=[0 3]
$Nt_{3,2}$, (x_0, x_2)=[0 0],(x_0, x_3)=(x_2, x_3)=[0 3]
$Nt_{1,3}$, (x_0, x_3)=(x_0, x_1)=(x_3, x_1)=[0 0]
$Nt_{2,3}$, (x_0, x_3)=(x_0, x_2)=(x_3, x_2)=[0 0]
$Nt_{3,4}$, (x_0, x_2)=[0 0],(x_0, x_3)=(x_2, x_3)=[0 2]
$Nt_{4,4}$, (x_0, x_2)=[0 0],(x_0, x_4)=(x_2, x_4)=[1 2]
$Nt_{2,5}$, (x_0, x_3)=(x_0, x_2)=(x_3, x_2)=[0 0]
$Nt_{3,6}$, (x_0, x_1)=(x_0, x_3)=[0 3],(x_1, x_3)=[0 2]
$Nt_{1,7}$, (x_0, x_3)=(x_0, x_1)=[0 3],(x_3, x_1)=[0 2]
$Nt_{5,7}$, (x_0, x_3)=(x_0, x_5)=[0 3],(x_3, x_5)=[0 2]
$Nt_{2,8}$, (x_0, x_1)=(x_0, x_2)=(x_1, x_2)=[0 0]
$Nt_{1,9}$, (x_0, x_2)=[0 0],(x_0, x_1)= (x_2, x_1)=[0 2]
$Nt_{5,9}$, (x_0, x_2)=[0 0],(x_0, x_5)=(x_2, x_5)=[0 2]
$Nt_{4,10}$, (x_2, x_3)=(x_3, x_4)=[0 2],(x_2, x_4)=[1 2]
$Nt_{5,10}$, (x_2, x_3)=(x_0, x_1)=(x_2, x_1)=[0 2]
$Nt_{4,12}$, (x_2, x_4)=[1 2]
$Nt_{5,12}$, (x_2, x_5)=[0 2]
$Nt_{4,13}$, (x_1, x_3)=(x_3, x_4)=[0 2],(x_1, x_4)=[1 2]
$Nt_{5,13}$, (x_1, x_3)=(x_1, x_5)=(x_3, x_5)=[0 2]
$Nt_{3,14}$, (x_0, x_4)=(x_0, x_3)=[1 3],(x_4, x_3)=[0 2]
$Nt_{5,15}$, (x_3, x_1)=(x_3, x_5)=(x_1, x_5)=[0 2]
$Nt_{1,16}$, (x_0, x_5)=(x_0, x_1)=(x_5, x_1)=[0 3]
$Nt_{5,17}$, (x_2, x_1)=(x_2, x_5)=(x_1, x_5)=[0 2]
$Nt_{1,18}$, (x_0, x_5)=[0 2],(x_0, x_1)=(x_5, x_1)=[0 3]
$Nt_{6,20}$, (x_5, x_6)=[0 2]
$Nt_{6,22}$, (x_1, x_6)=[0 2]
$Nt'_{1,2}$, (x_0, x_2)=[0 0],(x_0, x_1)=(x_2, x_1)=[0 2]
$Nt_{4,24}$, (x_3, x_1)=[0 1],(x_3, x_4)=(x_1, x_4)=[1 2]
$Nt'_{1,7}$, (x_0, x_3)=(x_0, x_1)=[0 3],(x_3, x_1)=[0 1]
$Nt_{4,25}$, (x_2, x_1)=[0 1],(x_2, x_4)=(x_1, x_4)=[1 2]
$Nt'_{1,9}$, (x_0, x_2)=[0 0],(x_0, x_1)=(x_2, x_1)=[0 1]
$Nt_{3,11}$, (x_0, x_4)=[1 2],(x_0, x_3)=[1 3],(x_4, x_3)=[0 2]
$Nt_{4,23}$, (x_0, x_1)=[0 2],(x_0, x_4)=[1 3],(x_1, x_4)=[1 2]

FIGURE B.4 – a)Tableau de classes, b)Réseaux de contraintes des arcs du graphe GR_c,FIG 3.10

$s_0 = \left\{ \begin{array}{l} M_0 : p_1, p_2, p_3 \to 1 \\ Z_0 : \left\{ \begin{array}{l} \alpha_1 \in [0 \;\; 0] \\ \alpha_2 \in [0 \;\; 0] \\ \alpha_3 \in [0 \;\; 0] \\ 0 \leq \alpha_1 - \alpha_2 \leq 0 \\ 0 \leq \alpha_1 - \alpha_3 \leq 0 \\ 0 \leq \alpha_3 - \alpha_4 \leq 0 \end{array} \right. \end{array} \right.$	$s_1 = \left\{ \begin{array}{l} M_1 : p_2, p_3, p_4 \to 1 \\ Z_1 : \left\{ \begin{array}{l} \alpha_2 \in [0 \;\; \infty] \\ \alpha_3 \in [0 \;\; \infty] \end{array} \right. \end{array} \right.$	$s_2 = \left\{ \begin{array}{l} M_2 : p_1, p_3, p_5 \to 1 \\ Z_2 : \left\{ \begin{array}{l} \alpha_1 \in [0 \;\; \infty] \\ \alpha_3 \in [0 \;\; \infty] \end{array} \right. \end{array} \right.$
$s_3 = \left\{ \begin{array}{l} M_3 : p_1, p_2, p_6 \to 1 \\ Z_3 : \left\{ \begin{array}{l} \alpha_1 \in [0 \;\; \infty] \\ \alpha_2 \in [0 \;\; \infty] \end{array} \right. \end{array} \right.$	$s_4 = \left\{ \begin{array}{l} M_4 : p_3, p_4, p_5 \to 1 \\ Z_4 : \left\{ \begin{array}{l} \alpha_3 \in [0 \;\; \infty] \\ \alpha_4 \in [0 \;\; \infty] \\ \alpha_4 - \alpha_3 \leq 0 \end{array} \right. \end{array} \right.$	$s_5 = \left\{ \begin{array}{l} M_5 : p_2, p_4, p_6 \to 1 \\ Z_5 : \{ \;\; \alpha_2 \in [0 \;\; \infty] \end{array} \right.$
$s_6 = \left\{ \begin{array}{l} M_6 : p_1, p_5, p_6 \to 1 \\ Z_6 : \left\{ \begin{array}{l} \alpha_1 \in [0 \;\; \infty] \\ \alpha_5 \in [0 \;\; \infty] \end{array} \right. \end{array} \right.$	$s_7 = \left\{ \begin{array}{l} M_7 : p_4, p_5, p_6 \to 1 \\ Z_7 : \left\{ \begin{array}{l} \alpha_4 \in [0 \;\; \infty] \\ \alpha_5 \in [0 \;\; \infty] \end{array} \right. \end{array} \right.$	$s_8 = \left\{ \begin{array}{l} M_8 : p_3, p_7 \to 1 \\ Z_8 : \{ \;\; \alpha_3 \in [1 \;\; \infty] \end{array} \right.$
$s_9 = \left\{ \begin{array}{l} M_9 : p_4, p_5, p_6 \to 1 \\ Z_9 : \left\{ \begin{array}{l} \alpha_4 \in [0 \;\; \infty] \\ \alpha_5 \in [0 \;\; \infty] \\ \alpha_4 - \alpha_5 \leq 0 \end{array} \right. \end{array} \right.$	$s_{10} = \left\{ \begin{array}{l} M_{10} : p_1, p_7 \to 1 \\ Z_{10} : \{ \;\; \alpha_1 \in [0 \;\; \infty] \end{array} \right.$	$s_{11} = \left\{ \begin{array}{l} M_{11} : p_6, p_7 \to 1 \\ Z_{11} : \{ \end{array} \right.$
$s_{12} = \left\{ \begin{array}{l} M_{12} : p_4, p_7 \to 1 \\ Z_{12} : \{ \;\; \alpha_6 \in [0 \;\; \infty] \end{array} \right.$	$s_{13} = \left\{ \begin{array}{l} M_{13} : p_1, p_2, p_3 \to 1 \\ Z_{13} : \left\{ \begin{array}{l} \alpha_1 \in [0 \;\; 0] \\ \alpha_2 \in [0 \;\; 0] \\ \alpha_3 \in [0 \;\; 0] \end{array} \right. \end{array} \right.$	

FIGURE B.5 – l'ensemble des classes du graphe GR_z fig 3.16b)
(Préservant propriétés LTL)

Annexe C

Graphes de classes d'un ITPN

$E_0 = \left\{ \begin{array}{l} M_0 : p_1, p_3, p_4 \rightarrow 1 \\ D_0 : \left\{ \begin{array}{l} 3 \leq \underline{t_1} \leq 3 \\ 2 \leq \underline{t_3} \leq 4 \\ 0 \leq \underline{t_4} \leq 2 \end{array} \right. \end{array} \right.$	$E_1 = \left\{ \begin{array}{l} M_1 : p_1, p_4 \rightarrow 1 \\ D_1 : \left\{ \begin{array}{l} 1 \leq \underline{t_1} \leq 1 \\ 0 \leq \underline{t_4} \leq 0 \end{array} \right. \end{array} \right.$	$E_2 = \left\{ \begin{array}{l} M_2 : p_1, p_2, p_3, p_7 \rightarrow 1 \\ D_2 : \left\{ \begin{array}{l} 1 \leq \underline{t_1} \leq 3 \\ 2 \leq \underline{t_2} \leq 5 \\ 10 \leq \underline{t_7} \leq 10 \\ \text{-1} \leq \underline{t_1} \text{- } \underline{t_3} \leq 1 \end{array} \right. \end{array} \right.$
$E_3 = \left\{ \begin{array}{l} M_3 : p_1, p_2, p_7 \rightarrow 1 \\ D_3 : \left\{ \begin{array}{l} 1 \leq \underline{t_1} \leq 1 \\ 2 \leq \underline{t_2} \leq 5 \\ 10 \leq \underline{t_7} \leq 10 \end{array} \right. \end{array} \right.$	$E_4 = \left\{ \begin{array}{l} M_4 : p_2, p_3, p_5, p_7 \rightarrow 1 \\ D_4 : \left\{ \begin{array}{l} 0 \leq \underline{t_5} \leq 0 \\ 9 \leq \underline{t_3} + \underline{t_7} \leq 11 \\ 7 \leq \underline{t_7} \leq 9 \\ 0 \leq \underline{t_2} \\ 5 \leq \underline{t_7} \text{- } \underline{t_2} \leq 8 \end{array} \right. \end{array} \right.$	$E_5 = \left\{ \begin{array}{l} M_5 : p_1, p_3, p_7 \rightarrow 1 \\ D_5 : \left\{ \begin{array}{l} \underline{t_7} \leq 8 \\ -\underline{t_1} + \underline{t_7} \geq 7 \\ 9 \leq -\underline{t_1} + \underline{t_3} + \underline{t_7} \leq 11 \\ \underline{t_1} \geq 0 \end{array} \right. \end{array} \right.$
$E_6 = \left\{ \begin{array}{l} M_6 : p_2, p_5, p_7 \rightarrow 1 \\ D_6 : \left\{ \begin{array}{l} 1 \leq \underline{t_2} \leq 4 \\ 0 \leq \underline{t_5} \leq 0 \\ 9 \leq \underline{t_7} \leq 9 \end{array} \right. \end{array} \right.$	$E_7 = \left\{ \begin{array}{l} M_7 : p_3, p_5, p_7 \rightarrow 1 \\ D_7 : \left\{ \begin{array}{l} 7 \leq \underline{t_7} \leq 8 \\ 9 \leq \underline{t_3} + \underline{t_7} \leq 11 \\ 0 \leq \underline{t_5} \leq 0 \end{array} \right. \end{array} \right.$	$E_8 = \left\{ \begin{array}{l} M_8 : p_2, p_3, p_6 \rightarrow 1 \\ D_8 : \left\{ \begin{array}{l} 0 \leq \underline{t_2} \leq 4 \\ 0 \leq \underline{t_3} \leq 4 \\ 4 \leq \underline{t_6} \leq 4 \\ 1 \leq \underline{t_2} + \underline{t_3} \leq 6 \end{array} \right. \end{array} \right.$
$E_9 = \left\{ \begin{array}{l} M_9 : p_2, p_6 \rightarrow 1 \\ D_9 : \left\{ \begin{array}{l} 1 \leq \underline{t_2} \leq 4 \\ 4 \leq \underline{t_6} \leq 4 \end{array} \right. \end{array} \right.$	$E_{10} = \left\{ \begin{array}{l} M_{10} : p_3, p_6 \rightarrow 1 \\ D_{10} : \left\{ \begin{array}{l} 1 \leq \underline{t_3} \leq 4 \\ 4 \leq \underline{t_6} \leq 4 \end{array} \right. \end{array} \right.$	$E_{11} = \left\{ \begin{array}{l} M_{11} : p_3, p_6 \rightarrow 1 \\ D_{11} : \left\{ \begin{array}{l} \underline{t_6} \leq 4 \\ -\underline{t_3} + \underline{t_6} \geq 0 \\ \underline{t_3} \geq 0 \\ 2 \leq 2^* \underline{t_6} \text{- } \underline{t_3} \leq 7 \end{array} \right. \end{array} \right.$
$E_{12} = \left\{ \begin{array}{l} M_{12} : p_2, p_6 \rightarrow 1 \\ D_{12} : \left\{ \begin{array}{l} \underline{t_6} \leq 4 \\ \underline{t_2} \geq 0 \\ 2 \leq 2^* \ \underline{t_6} \text{- } \underline{t_2} \leq 7 \end{array} \right. \end{array} \right.$	$E_{13} = \left\{ \begin{array}{l} M_{13} : p_6 \rightarrow 1 \\ D_{13} : \{ \ 0 \leq \underline{t_6} \leq 3 \end{array} \right.$	$E_{14} = \left\{ \begin{array}{l} M_{14} : p_2 \rightarrow 1 \\ D_{14} : \{ \ 0 \leq \underline{t_2} \leq 0 \end{array} \right.$
$E_{15} = \left\{ \begin{array}{l} M_{15} : p_3 \rightarrow 1 \\ D_{15} : \{ \ 0 \leq \underline{t_3} \leq 0 \end{array} \right.$	$E_{16} = \left\{ \begin{array}{l} M_{16} : p_6 \rightarrow 1 \\ D_{16} : \left\{ \begin{array}{l} 0 \leq \underline{t_6} \\ 2^* \underline{t_6} \leq 7 \end{array} \right. \end{array} \right.$	$E_{17} = \left\{ \begin{array}{l} M_{17} : \\ D_{17} : \{ \end{array} \right.$

FIGURE C.1 – L'ensemble des classes du graphe exact GR Fig 4.1b)

$E_0 = \left\{ \begin{array}{l} M_0 : p_1, p_3, p_4 \rightarrow 1 \\ D_0 : \left\{ \begin{array}{l} 3 \leq \underline{t_1} \leq 3 \\ 2 \leq \underline{t_3} \leq 4 \\ 0 \leq \underline{t_4} \leq 2 \\ \underline{t_1} - \underline{t_3} \leq 1 \\ \underline{t_1} - \underline{t_4} \leq 3 \\ \underline{t_3} - \underline{t_1} \leq 1 \\ \underline{t_3} - \underline{t_4} \leq 4 \\ \underline{t_4} - \underline{t_1} \leq -1 \\ \underline{t_4} - \underline{t_3} \leq 0 \end{array} \right. \end{array} \right.$

$E_1 = \left\{ \begin{array}{l} M_1 : p_1, p_4 \rightarrow 1 \\ D_1 : \left\{ \begin{array}{l} 1 \leq \underline{t_1} \leq 1 \\ 0 \leq \underline{t_4} \leq 0 \\ \underline{t_1} - \underline{t_4} \leq 1 \\ \underline{t_4} - \underline{t_1} \leq -1 \end{array} \right. \end{array} \right.$

$E_2 = \left\{ \begin{array}{l} M_2 : p_1, p_2, p_3, p_7 \rightarrow 1 \\ D_2 : \left\{ \begin{array}{l} 1 \leq \underline{t_1} \leq 3 \\ 2 \leq \underline{t_2} \leq 5 \\ 10 \leq \underline{t_7} \leq 10 \\ 0 \leq \underline{t_3} \leq 4 \\ -4 \leq \underline{t_1} - \underline{t_2} \leq 1 \\ -1 \leq \underline{t_1} - \underline{t_3} \leq 1 \\ 7 \leq \underline{t_7} - \underline{t_1} \leq 9 \\ 5 \leq \underline{t_7} - \underline{t_2} \leq 8 \\ -2 \leq \underline{t_2} - \underline{t_3} \leq 5 \\ 6 \leq \underline{t_7} - \underline{t_3} \leq 10 \end{array} \right. \end{array} \right.$

$E_3 = \left\{ \begin{array}{l} M_3 : p_1, p_2, p_7 \rightarrow 1 \\ D_3 : \left\{ \begin{array}{l} 1 \leq \underline{t_1} \leq 1 \\ 2 \leq \underline{t_2} \leq 5 \\ 10 \leq \underline{t_7} \leq 10 \\ 1 \leq \underline{t_2} - \underline{t_1} \geq 4 \\ 5 \leq \underline{t_7} - \underline{t_2} \geq 8 \\ 9 \leq \underline{t_7} - \underline{t_1} \geq 9 \end{array} \right. \end{array} \right.$

$E_4 = \left\{ \begin{array}{l} M_4 : p_2, p_3, p_5, p_7 \rightarrow 1 \\ D_4 : \left\{ \begin{array}{l} 0 \leq \underline{t_5} \leq 0 \\ 9 \leq \underline{t_3} + \underline{t_7} \leq 11 \\ 7 \leq \underline{t_7} \leq 9 \\ 5 \leq \underline{t_7} - \underline{t_2} \leq 8 \\ 0 \leq \underline{t_2} \end{array} \right. \end{array} \right.$

$E_5 = \left\{ \begin{array}{l} M_5 : p_1, p_3, p_7 \rightarrow 1 \\ D_5 : \left\{ \begin{array}{l} \underline{t_7} \leq 8 \\ 7 \leq \underline{t_7} - \underline{t_1} \\ 9 \leq \underline{t_3} - \underline{t_1} + \underline{t_7} \leq 11 \end{array} \right. \end{array} \right.$

$E_6 = \left\{ \begin{array}{l} M_6 : p_2, p_5, p_7 \rightarrow 1 \\ D_6 : \left\{ \begin{array}{l} 1 \leq \underline{t_2} \leq 4 \\ 0 \leq \underline{t_5} \leq 0 \\ 9 \leq \underline{t_7} \leq 9 \\ 1 \leq \underline{t_2} - \underline{t_5} \leq 4 \\ 5 \leq \underline{t_7} - \underline{t_2} \leq 8 \\ 9 \leq \underline{t_7} - \underline{t_5} \leq 9 \end{array} \right. \end{array} \right.$

$E_7 = \left\{ \begin{array}{l} M_7 : p_3, p_5, p_7 \rightarrow 1 \\ D_7 : \left\{ \begin{array}{l} 0 \leq \underline{t_5} \leq 0 \\ 7 \leq \underline{t_7} \leq 8 \\ 9 \leq \underline{t_7} + \underline{t_3} \leq 11 \end{array} \right. \end{array} \right.$

$E_8 = \left\{ \begin{array}{l} M_8 : p_2, p_3, p_6 \rightarrow 1 \\ D_8 : \left\{ \begin{array}{l} 0 \leq \underline{t_2} \leq 4 \\ 0 \leq \underline{t_3} \leq 4 \\ 4 \leq \underline{t_6} \leq 4 \\ 1 \leq \underline{t_3} + \underline{t_2} \leq 6 \end{array} \right. \end{array} \right.$

$E_9 = \left\{ \begin{array}{l} M_9 : p_2, p_6 \rightarrow 1 \\ D_9 : \left\{ \begin{array}{l} 1 \leq \underline{t_2} \leq 4 \\ 4 \leq \underline{t_6} \leq 4 \\ 0 \leq \underline{t_6} - \underline{t_2} \leq 3 \end{array} \right. \end{array} \right.$

$E_{10} = \left\{ \begin{array}{l} M_{10} : p_3, p_6 \rightarrow 1 \\ D_{10} : \left\{ \begin{array}{l} 1 \leq \underline{t_3} \leq 4 \\ 4 \leq \underline{t_6} \leq 4 \\ 0 \leq \underline{t_6} - \underline{t_3} \leq 3 \end{array} \right. \end{array} \right.$

$E_{11} = \left\{ \begin{array}{l} M_{11} : p_3, p_6 \rightarrow 1 \\ D_{11} : \left\{ \begin{array}{l} 0 \leq \underline{t_3} \\ \underline{t_6} \leq 4 \\ 0 \leq \underline{t_6} - \underline{t_3} \\ 2 \leq 2^* \ \underline{t_6} - \underline{t_3} \leq 7 \end{array} \right. \end{array} \right.$

$E_{12} = \left\{ \begin{array}{l} M_{12} : p_2, p_6 \rightarrow 1 \\ D_{12} : \left\{ \begin{array}{l} \underline{t_6} \leq 4 \\ 0 \leq \underline{t_6} - \underline{t_2} \\ 0 \leq \underline{t_2} \\ 2 \leq 2^* \ \underline{t_6} - \underline{t_2} \leq 7 \end{array} \right. \end{array} \right.$

$E_{13} = \left\{ \begin{array}{l} M_{13} : p_6 \rightarrow 1 \\ D_{13} : \{ \ 0 \leq \underline{t_6} \leq 3 \end{array} \right.$

$E_{14} = \left\{ \begin{array}{l} M_{14} : p_2 \rightarrow 1 \\ D_{14} : \{ \ 0 \leq \underline{t_2} \leq 0 \end{array} \right.$

$E_{15} = \left\{ \begin{array}{l} M_{15} : p_3 \rightarrow 1 \\ D_{15} : \{ \ 0 \leq \underline{t_3} \leq 0 \end{array} \right.$

$E_{16} = \left\{ \begin{array}{l} M_{16} : p_6 \rightarrow 1 \\ D_{16} : \left\{ \begin{array}{l} 0 \leq \underline{t_6} \\ 2^* \underline{t_6} \leq 7 \end{array} \right. \end{array} \right.$

$E_{17} = \left\{ \begin{array}{l} M_{17} : \\ D_{17} : \{ \end{array} \right.$

FIGURE C.2 – L'ensemble des classes du graphe exact (méthode mixte) Fig 4.1b)

$E_0 = \left\{ \begin{array}{l} M_0 : p_1, p_3, p_4 \to 1 \\ D_0 : \left\{ \begin{array}{l} 3 \leq \underline{t_1} \leq 3 \\ 2 \leq \underline{t_3} \leq 4 \\ 0 \leq \underline{t_4} \leq 2 \\ \underline{t_1} - \underline{t_3} \leq 1 \\ \underline{t_1} - \underline{t_4} \leq 3 \\ \underline{t_3} - \underline{t_1} \leq 1 \\ \underline{t_3} - \underline{t_4} \leq 4 \\ \underline{t_4} - \underline{t_1} \leq -1 \\ \underline{t_4} - \underline{t_3} \leq 0 \end{array} \right. \end{array} \right.$	$E_1 = \left\{ \begin{array}{l} M_1 : p_1, p_4 \to 1 \\ D_1 : \left\{ \begin{array}{l} 1 \leq \underline{t_1} \leq 1 \\ 0 \leq \underline{t_4} \leq 0 \\ \underline{t_1} - \underline{t_4} \leq 1 \\ \underline{t_4} - \underline{t_1} \leq -1 \end{array} \right. \end{array} \right.$	$E_2 = \left\{ \begin{array}{l} M_2 : p_1, p_2, p_3, p_7 \to 1 \\ D_2 : \left\{ \begin{array}{l} 1 \leq \underline{t_1} \leq 3 \\ 2 \leq \underline{t_2} \leq 5 \\ 10 \leq \underline{t_7} \leq 10 \\ 0 \leq \underline{t_3} \leq 4 \\ -4 \leq \underline{t_1} - \underline{t_2} \leq 1 \\ -1 \leq \underline{t_1} - \underline{t_3} \leq 1 \\ 7 \leq \underline{t_7} - \underline{t_1} \leq 9 \\ 5 \leq \underline{t_7} - \underline{t_2} \leq 8 \\ -2 \leq \underline{t_2} - \underline{t_3} \leq 5 \\ 6 \leq \underline{t_7} - \underline{t_3} \leq 10 \end{array} \right. \end{array} \right.$
$E_3 = \left\{ \begin{array}{l} M_3 : p_1, p_2, p_7 \to 1 \\ D_3 : \left\{ \begin{array}{l} 1 \leq \underline{t_1} \leq 1 \\ 2 \leq \underline{t_2} \leq 5 \\ 10 \leq \underline{t_7} \leq 10 \\ 1 \leq \underline{t_2} - \underline{t_1} \geq 4 \\ 5 \leq \underline{t_7} - \underline{t_2} \geq 8 \\ 9 \leq \underline{t_7} - \underline{t_1} \geq 9 \end{array} \right. \end{array} \right.$	$E_4 = \left\{ \begin{array}{l} M_4 : p_2, p_3, p_5, p_7 \to 1 \\ D_4 : \left\{ \begin{array}{l} 0 \leq \underline{t_2} \leq 4 \\ 0 \leq \underline{t_3} \leq 4 \\ 0 \leq \underline{t_5} \leq 0 \\ 7 \leq \underline{t_7} \leq 9 \\ -4 \leq \underline{t_2} - \underline{t_3} \leq 4 \\ 0 \leq \underline{t_2} - \underline{t_5} \leq 4 \\ 5 \leq \underline{t_7} - \underline{t_2} \leq 8 \\ 0 \leq \underline{t_3} - \underline{t_5} \leq 4 \\ 3 \leq \underline{t_7} - \underline{t_3} \leq 9 \\ 7 \leq \underline{t_7} - \underline{t_5} \leq 9 \end{array} \right. \end{array} \right.$	$E_5 = \left\{ \begin{array}{l} M_5 : p_1, p_3, p_7 \to 1 \\ D_5 : \left\{ \begin{array}{l} 0 \leq \underline{t_1} \leq 1 \\ 1 \leq \underline{t_3} \leq 4 \\ 7 \leq \underline{t_7} \leq 8 \\ 1 \leq \underline{t_3} - \underline{t_1} \leq 4 \\ 7 \leq \underline{t_7} - \underline{t_1} \leq 8 \\ 3 \leq \underline{t_7} - \underline{t_3} \leq 7 \end{array} \right. \end{array} \right.$
$E_6 = \left\{ \begin{array}{l} M_6 : p_2, p_5, p_7 \to 1 \\ D_6 : \left\{ \begin{array}{l} 1 \leq \underline{t_2} \leq 4 \\ 0 \leq \underline{t_5} \leq 0 \\ 9 \leq \underline{t_7} \leq 9 \\ 1 \leq \underline{t_2} - \underline{t_5} \leq 4 \\ 5 \leq \underline{t_7} - \underline{t_2} \leq 8 \\ 9 \leq \underline{t_7} - \underline{t_5} \leq 9 \end{array} \right. \end{array} \right.$	$E_7 = \left\{ \begin{array}{l} M_7 : p_3, p_5, p_7 \to 1 \\ D_7 : \left\{ \begin{array}{l} 0 \leq \underline{t_3} \leq 4 \\ 0 \leq \underline{t_5} \leq 0 \\ 7 \leq \underline{t_7} \leq 8 \\ 0 \leq \underline{t_3} - \underline{t_5} \leq 4 \\ 3 \leq \underline{t_7} - \underline{t_3} \leq 8 \\ 7 \leq \underline{t_7} - \underline{t_5} \leq 8 \end{array} \right. \end{array} \right.$	$E_8 = \left\{ \begin{array}{l} M_8 : p_2, p_3, p_6 \to 1 \\ D_8 : \left\{ \begin{array}{l} 0 \leq \underline{t_2} \leq 4 \\ 0 \leq \underline{t_3} \leq 4 \\ -4 \leq \underline{t_6} \leq 4 \\ 0 \leq \underline{t_6} - \underline{t_2} \leq 4 \\ 0 \leq \underline{t_6} - \underline{t_3} \leq 4 \\ -4 \leq \underline{t_3} - \underline{t_2} \leq 4 \end{array} \right. \end{array} \right.$
$E_9 = \left\{ \begin{array}{l} M_9 : p_2, p_6, p_7 \to 1 \\ D_9 : \left\{ \begin{array}{l} 0 \leq \underline{t_5} \leq 0 \\ 1 \leq \underline{t_3} \leq 4 \\ 7 \leq \underline{t_7} \leq 8 \\ 1 \leq \underline{t_3} - \underline{t_5} \leq 4 \\ 3 \leq \underline{t_7} - \underline{t_3} \leq 7 \\ 7 \leq \underline{t_7} - \underline{t_5} \leq 8 \end{array} \right. \end{array} \right.$	$E_{10} = \left\{ \begin{array}{l} M_{10} : p_2, p_6 \to 1 \\ D_{10} : \left\{ \begin{array}{l} 1 \leq \underline{t_2} \leq 4 \\ 4 \leq \underline{t_6} \leq 4 \\ 0 \leq \underline{t_6} - \underline{t_2} \leq 3 \end{array} \right. \end{array} \right.$	$E_{11} = \left\{ \begin{array}{l} M_{11} : p_3, p_6 \to 1 \\ D_{11} : \left\{ \begin{array}{l} 0 \leq \underline{t_3} \leq 4 \\ 4 \leq \underline{t_6} \leq 4 \\ 0 \leq \underline{t_6} - \underline{t_3} \leq 4 \end{array} \right. \end{array} \right.$
$E_{12} = \left\{ \begin{array}{l} M_{12} : p_3, p_6 \to 1 \\ D_{12} : \left\{ \begin{array}{l} 0 \leq \underline{t_3} \leq 4 \\ 0 \leq \underline{t_6} \leq 4 \\ 0 \leq \underline{t_6} - \underline{t_3} \leq 4 \end{array} \right. \end{array} \right.$	$E_{13} = \left\{ \begin{array}{l} M_{13} : p_2, p_6 \to 1 \\ D_{13} : \left\{ \begin{array}{l} 0 \leq \underline{t_2} \leq 4 \\ 0 \leq \underline{t_6} \leq 4 \\ 0 \leq \underline{t_6} - \underline{t_2} \leq 4 \end{array} \right. \end{array} \right.$	$E_{14} = \left\{ \begin{array}{l} M_{14} : p_2, p_3 \to 1 \\ D_{14} : \left\{ \begin{array}{l} 0 \leq \underline{t_2} \leq 0 \\ 0 \leq \underline{t_3} \leq 0 \\ 0 \leq \underline{t_3} - \underline{t_2} \leq 0 \end{array} \right. \end{array} \right.$
$E_{15} = \left\{ \begin{array}{l} M_{15} : p_3, p_6 \to 1 \\ D_{15} : \left\{ \begin{array}{l} 1 \leq \underline{t_3} \leq 4 \\ 4 \leq \underline{t_6} \leq 4 \\ 0 \leq \underline{t_6} - \underline{t_3} \leq 3 \end{array} \right. \end{array} \right.$	$E_{16} = \left\{ \begin{array}{l} M_{16} : p_6 \to 1 \\ D_{16} : \{ \ 0 \leq \underline{t_6} \leq 3 \end{array} \right.$	$E_{17} = \left\{ \begin{array}{l} M_{17} : p_2 \to 1 \\ D_{17} : \{ \ 0 \leq \underline{t_2} \leq 0 \end{array} \right.$
$E_{18} = \left\{ \begin{array}{l} M_{18} : p_6 \to 1 \\ D_{18} : \{ \ 0 \leq \underline{t_6} \leq 4 \end{array} \right.$	$E_{19} = \left\{ \begin{array}{l} M_{19} : p_3 \to 1 \\ D_{19} : \{ \ 0 \leq \underline{t_3} \leq 0 \end{array} \right.$	$E_2 = \left\{ \begin{array}{l} M_{20} : \\ D_{20} : \{ \end{array} \right.$

FIGURE C.3 – L'ensemble des classes du graphe approximé par DBM $\widetilde{GR}$; Fig 4.2

Class	Marking	Constraints Nc	$In_{i,c}$
$\mathcal{C}_0$	$p_1p_3p_4$	x_0	$\emptyset$
$\mathcal{C}_1$	p_1p_4	$2 \leq x_3 - x_0 \leq 2$	$\emptyset$
$\mathcal{C}_2$	$p_1p_2p_3p_7$	$0 \leq x_4 - x_0 \leq 2$	$\emptyset$
$\mathcal{C}_3$	$p_1p_2p_7$	$2 \leq x_4 - x_0 \leq 2$	$\emptyset$
$\mathcal{C}_4$	$p_2p_3p_5p_7$	$0 \leq x_4 - x_0 \leq 2\,;\, 3 \leq x_1 - x_0 \leq 3\,;\, 1 \leq x_1 - x_4 \leq 3$	$\emptyset$
$\mathcal{C}_5$	$p_1p_3p_7$	$0 \leq x_4 - x_0 \leq 1\,;\, 2 \leq x_2 - x_0 \leq 3\,;\, 2 \leq x_2 - x_4 \leq 4$	$\emptyset$
$\mathcal{C}_6$	$p_2p_5p_7$	$1 \leq x_1 - x_4 \leq 1$	$\emptyset$
$\mathcal{C}_7$	$p_3p_5p_7$	$2 \leq x_1 - x_4 \leq 3\,;\, 2 \leq x_2 - x_4 \leq 3\,;\, 0 \leq x_2 - x_1 \leq 0$	$\emptyset$
$\mathcal{C}_8$	$p_2p_3p_6$	$0 \leq x_4 - x_0 \leq 2\,;\, 1 \leq x_5 - x_4 \leq 3\,;\, 3 \leq x_5 - x_0 \leq 3$	$In_{3,8} = [1\ \ 3]$
$\mathcal{C}_9$	$p_3p_5p_7$	$0 \leq x_4 - x_0 \leq 1\,;\, 3 \leq x_1 - x_0 \leq 3\,;\, 2 \leq x_1 - x_4 \leq 3$	$\emptyset$
$\mathcal{C}_{10}$	p_2p_6	$1 \leq x_5 - x_4 \leq 1$	$\emptyset$
$\mathcal{C}_{11}$	p_3p_6	$3 \leq x_5 - x_0 \leq 3$	$In_{3,11} = [2\ \ 3]$
$\mathcal{C}_{12}$	p_3p_6	$3 \leq x_5 - x_0 \leq 3\,;\, 3 \leq x_2 - x_0 \leq 7\,;\, 0 \leq x_2 - x_5 \leq 4$	$In_{3,12} = [1\ \ 3]$
$\mathcal{C}_{13}$	p_2p_6	$1 \leq x_5 - x_4 \leq 3\,;\, 1 \leq x_3 - x_4 \leq 5\,;\, 0 \leq x_3 - x_5 \leq 4$	$\emptyset$
$\mathcal{C}_{14}$	p_2p_3	$2 \leq x_4 - x_0 \leq 2\,;\, 7 \leq x_6 - x_0 \leq 7\,;\, 5 \leq x_6 - x_4 \leq 5$	$In_{3,14} = [1\ \ 3]$
$\mathcal{C}_{15}$	p_6	$1 \leq x_2 - x_5 \leq 4$	$\emptyset$
$\mathcal{C}_{16}$	p_2	$5 \leq x_6 - x_4 \leq 5$	$\emptyset$
$\mathcal{C}_{17}$	p_6	$1 \leq x_3 - x_5 \leq 4$	$In_{3,17} = [2\ \ 3]$
$\mathcal{C}_{18}$	p_3	$7 \leq x_6 - x_0 \leq 7$	$In_{3,18} = [2\ \ 3]$
$\mathcal{C}_{19}$	p_6	$0 \leq x_3 - x_5 \leq 4$	$\emptyset$
$\mathcal{C}_{20}$	p_6	$0 \leq x_2 - x_5 \leq 4$	$\emptyset$
$\mathcal{C}_{21}$	p_3	$7 \leq x_2 - x_0 \leq 7$	$\emptyset$
$\mathcal{C}_{22}$	p_2	$5 \leq x_3 - x_4 \leq 7$	$\emptyset$
$\mathcal{C}_{23}$	$\emptyset$	x_6	$\emptyset$
$\mathcal{C}_{24}$	$\emptyset$	x_3	$\emptyset$
$\mathcal{C}_{25}$	$\emptyset$	x_2	$\emptyset$
$\mathcal{C}_{26}$	$p_1p_2p_3p_7$	$0 \leq x_4 - x_0 \leq 1$	$\emptyset$
$\mathcal{C}_{27}$	$p_2p_3p_6$	$2 \leq x_4 - x_0 \leq 2\,;\, 1 \leq x_5 - x_4 \leq 1\,;\, 3 \leq x_5 - x_0 \leq 3$	$In_{3,8} = [1\ \ 3]$
$\mathcal{C}_{28}$	$p_2p_3p_5p_7$	$2 \leq x_4 - x_0 \leq 2\,;\, 3 \leq x_1 - x_0 \leq 3\,;\, 1 \leq x_1 - x_4 \leq 1$	$\emptyset$
$\mathcal{C}_{19}$	$p_1p_2p_3p_7$	$2 \leq x_4 - x_0 \leq 2$	$\emptyset$
$\mathcal{C}_{30}$	p_2p_6	$1 \leq x_5 - x_4 \leq 1\,;\, 1 \leq x_3 - x_4 \leq 5\,;\, 0 \leq x_3 - x_5 \leq 4$	$\emptyset$

FIGURE C.4 – GraphC : Proposition de tableau de classes de l'$ITPN$ FIG : 5.14b)

Réseaux temporels $Nt_{i,k}$ de t_i à partir de $\mathcal{C}_k$
$Nt_{4,0}$, (x_0, x_4)=[0 2]
$Nt_{1,2}$, (x_0, x_4)=[0 2],(x_0, x_1)=[3 3],(x_4, x_1)=[1 3]
$Nt_{2,4}$, (x_4, x_1)=[2 3],(x_4, x_2)=[2 3],(x_1, x_2)=[0 0]
$Nt_{5,7}$, (x_4, x_1)=[2 3],(x_4, x_2)=[2 3],(x_4, x_5)=[2 3], ,(x_1, x_2)=[0 0],(x_1, x_5)=[0 0],(x_2, x_5)=[0 0]
$Nt_{3,11}$, (x_0, x_5)=[3 3],(x_0, x_3)=[4 7],(x_5, x_3)=[1 4]
$Nt_{6,11}$, (x_0, x_5)=[3 3],(x_0, x_6)=[7 7],(x_5, x_6)=[4 4]
$Nt_{3,17}$, (x_0, x_3)=[4 7],(x_0, x_6)=[7 7],(x_6, x_3)=[0 0]
$Nt_{6,18}$, (x_0, x_5)=[3 3],(x_0, x_6)=[7 7],(x_5, x_6)=[4 4]
$Nt'_{4,0}$, (x_0, x_4)=[0 1]

FIGURE C.5 – GraphC : Proposition de tableau d'arcs associée à la séquence σ_1 de l'$ITPN$ FIG : 5.14b)