N°d'ORDRE : 05/2009-D/IN

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET
DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE
<<HOUARI BOUMEDIEN>>
FACULTE D'ELECTRONIQUE ET D'INFORMATIQUE

THESE

Présentée pour l'obtention du diplôme de DOCTORAT

EN : INFORMATIQUE

Spécialité : Informatique

Par : DAHMANI /ép ZAOUCHE Djaouida

Sujet

EXTENSIONS DES RÉSEAUX DE PETRI RÉCURSIFS POUR L'ANALYSE TEMPORELLE DES SYSTÈMES À STRUCTURE DYNAMIQUE

Soutenue le 01/07/2009, devant le jury composé de :

Mr C.B. BEN YELLES	**Professeur**	**IUT Valence**	**Président**
Mme M. BOUKALA	**Professeur**	**USTHB (Alger)**	**Dir. de thèse**
Mme A. AISSANI	**Professeur**	**USTHB (Alger)**	**Examinateur**
Mr J.M. ILIE	**Maître de Conférences**	**PARIS 6**	**Examinateur**
Mr D. SAIDOUNI	**Maître de Conférences**	**Université de Constantine**	**Examinateur**
Mme N. BENSAOU	**Maître de Conférences**	**USTHB (Alger)**	**Examinateur**

REMERCIEMENTS

Je tiens tout d'abord à remercier Monsieur le Professeur Choukri Bey Ben-yelles pour l'honneur qu'il me fait en acceptant de juger ce travail et de présider le jury de cette thèses.

Je tiens à exprimer ma profonde reconnaissance à Madame le Professeur Malika Boukala Ioualalen, ma directrice de thèse, qui s'est toujours montrée disponible et surtout patiente. Je la remercie de m'avoir accueillie au sein de son équipe, pour sa gentillesse, la confiance qu'elle m'a accordée, ses encouragements, ses remarques pertinentes et son encadrement.

Merci à Jean Michel Ilié pour son aide précieuse et ses remarques pertinentes.

Je remercie Madame le Professeur Aicha Aissani Mokhtari et le Docteur Djamel-Eddine Sadouni d'avoir accepté de juger cette thèse.

Je remercie Madame le Professeur Aicha Aissani Mokhtari , le Docteur Djamel-Eddine Sadouni , Madame le Docteur Bensaou Nacéra, Monsieur le Professeur Jean –Michel Ilié pour l'intérêt qu'ils ont porté à ce travail en faisant l'honneur d'accepter de le juger et de participer au jury de cette thèse.

Je tiens à remercier Samia Mazouz pour la collaboration enrichissante que nous avons développée.

Merci également à toutes les personnes avec qui j'ai travaillé ou j'ai simplement eu le plaisir de les côtoyer, en particulier les membres de mon équipe de recherche: Salmi Nabila, Chérif Boukala, Nawel Gharbi , Abdenour Himrane et Meriem Taibi.

Merci à mes amis et collègues Karim Benabadji, Chouhed Bouabana, Abdelkader Belkhir, Baya Kadri, Malika mameri, Lies kaddouri, Karima Akli et Baha Nadia.

Je tiens à remercier mon mari Zine pour son soutien inconditionnel et constant. Il a toujours cru à l'aboutissement de cette thèse. Cette thèse lui doit énormément.

Merci à mes filles Mélissa et Maria d'avoir accepté une maman souvent absente et supporté mes changements d'humeurs occasionnés par ce travail.

Table des matières

Table des figures

Chapitre 1

Introduction

Dans ce chapitre, nous présentons brièvement les modèles construits autour des réseaux de Petri et dédiés à la modélisation des systèmes à structure dynamique. En particulier, nous abordons l'analyse temporelle de ces systèmes.

1.1 Réseaux de Petri de Haut Niveau

Les réseaux de Petri offrent un support graphique et une base sémantique claire pour la description des systèmes concurrents. Ce formalisme pionnier, introduit par C.A. Petri en 1962, a fourni les premières approches de modélisation. Ce modèle, considéré de bas niveau de nos jours, s'adapte uniquement pour la modélisation de systèmes simples. En revanche, lorsque les systèmes à modéliser sont complexes, on s'orientera plutôt vers d'autres formalismes tels que les spécifications algébriques ou des modèles de haut niveau comme les réseaux de Petri colorés [36]. Ces derniers permettent des modélisations très compactes. En effet, la possibilité d'associer une information (i.e. une couleur) aux jetons et de paramétrer le franchissement de transition, permet la représentation des systèmes complexes d'une manière très concise et détaillée, ce qui aurait nécessité des réseaux non-colorés beaucoup plus volumineux et probablement illisibles. Toutefois, il faut noter que les réseaux de Petri colorés, avec un domaine fini de couleurs, ne fournissent pas une véritable extension des réseaux de Petri mais une simple abbréviation de ces derniers. En revanche, les réseaux de Petri colorés, avec un domaine infini de couleurs, constituent une véritable extension des réseaux de Petri.

Le pouvoir d'expression des réseaux de Petri est proche d'un langage de programmation travaillant sur des entiers. Il manque cependant aux réseaux le test d'égalité entre le marquage d'une place et une valeur fixe pour parvenir à en faire un véritable langage de programmation. C'est pourquoi les réseaux de Petri ont été étendus par les arcs inhibiteurs [58]. Ce pouvoir d'expression étendu conduit inévitablement à l'indécidabilité de toutes les propriétés intéressantes (accessibilité, vivacité, caractère borné, couverture,

terminaison,....). Néanmoins, l'accessibilité reste, sous certaines conditions, décidable. En particulier, cette propriété est assurée en présence d'un seul arc inhibiteur. Par ailleurs, les réseaux auto-modifiants, introduits par R. Valk, constituent une autre extension intéressante des réseaux de Petri [62] [63]. Dans ce modèle étendu, les jetons à consommer et à produire dépendent du marquage courant. Dans le modèle initial des réseaux auto-modifiants, la valuation d'un arc s'exprime comme une combinaison linéaire du marquage des places plus une constante. Ces réseaux auto-modifiants ont la puissance d'une machine de Turing et ainsi les propriétés classiques telles que l'accessibilité sont indécidables [62] [63]. Des restrictions imposées à ce type de réseaux permettent de décider certaines propriétés (bornitude, couverture, terminaison,...) mais l'accessibilité, pour sa part, reste indécidable [23]. De plus, ces extensions n'offrent pas un cadre *adéquat* pour modéliser la *création dynamique* des objets. Pour remédier à cette limite, A. Kiehn a introduit un modèle appelé systèmes de réseaux [41]. Ces derniers sont un ensemble de réseaux de Petri dotés de transitions spéciales, tel que le tir de chacune déclenche l'exécution d'un de ces réseaux. Par la suite, les réseaux de Petri à objets de R. Valk [64] ont été proposés dans le but d'adapter plus les réseaux de Petri à la manipulation dynamique des objets. Dans ce modèle, les jetons sont eux-mêmes des réseaux de Petri et la création d'un nouveau processus est possible grâce à la production d'un nouveau jeton qui est un réseau. De plus, la hiérarchie est limitée à une profondeur de deux niveaux et le mécanisme de synchronisation ne peut avoir lieu qu'entre le niveau supérieur et un de ses sous-processus. Encore une fois, il a été prouvé que l'accessibilité est indécidable [42]. Un modèle similaire, appelé "nested Petri nets", a été introduit dans [46] présentant les caractéristiques suivantes : la profondeur de la hiérarchie est non bornée et la synchronisation entre les niveaux est plus générale. Ici, comme pour les modèles précédents, il a été prouvé que l'accessibilité et la bornitude sont indécidables, en revanche la terminaison et d'autres propriétés classiques restent décidables.

Les Réseaux de Petri Récursifs (RdPR) ont été proposés comme un autre modèle adapté à la manipulation dynamique des objets. En effet, ils se prêtent bien à la spécification et l'analyse des systèmes complexes à structure dynamique qui, de plus, sont potentiellement infinis [29] [32]. Aussi, ils permettent d'une part de modéliser les procédures, et d'autre part de simuler la création et la destruction des processus de manière *élégante* et *naturelle*. Il est à noter que les processus sont créés par un nouveau type de transitions, appelées transitions abstraites. Quand un processus tire une transition abstraite, il consomme les jetons en entrée de la transition comme pour une transition ordinaire puis il crée un nouveau processus fils. Ce dernier commence son exécution avec un marquage initial que l'on appelle le marquage de départ de la transition abstraite. La production des jetons de sortie de la transition abstraite tirée est retardée jusqu'à ce que le processus fils termine son exécution en atteignant un marquage final. Un tel marquage est caractérisé par son appartenance à un ensemble, attaché au réseau, dit ensemble des marquages finaux. D'autre part, l'opération de terminaison ainsi décrite est appelée une *coupure* et correspond à la *fermeture* de la transition abstraite tirée. Les RdPR ont été utilisés avec succès pour la spécification des plans des agents dans les systèmes multi-agents [61] [60] ; ils incluent des mécanismes complexes tels que les interruptions, les tolérances aux pannes et les appels de procédures distants. Par ailleurs, des procédures de décision de quelques propriétés fondamentales ont été développées pour les RdPR. En particulier, l'accessibilité, la bornitude et la finitude ont été prouvées décidables.

Dans le cadre de cette thèse, nous nous sommes intéressés à la vérification des *contraintes temporelles* des systèmes à structure dynamique. Avant de brosser un aperçu général de notre contribution, nous allons

d'abord présenter très briévement les principaux modèles temporisés.

1.2 Modèles Temporisés

Certains systèmes *temps réel* doivent réagir aux évènements externes ou internes avec un temps de réponse suffisamment petit. Parfois, ils sont soumis à des *contraintes temporelles* fortes. Il est donc important de s'assurer de leur correction non seulement *fonctionnelle*, mais aussi *temporelle*. De plus, les applications temps réel étant en général critiques, il est important de détecter d'éventuelles erreurs le plus en amont possible dans la phase de conception. Par ailleurs, la vérification des comportements temporels des systèmes exige l'usage de modèles dit temporisés pour lesquels le temps est manipulé de manière explicite. Il est à noter que les trois principales familles de modèles temporisés sont les extensions au temps des algèbres de processus, des automates finis et des réseaux de Petri. Dans le cadre de cette thèse, nous nous sommes portés vers les réseaux de Petri avec le temps en raison de leur puissance d'expression condenséee du parallélisme et leur adéquation au problème de la modélisation des applications temps réel. Notons que les deux extensions temporelles principales des réseaux de Petri sont :
- Les réseaux de Petri *temporisés* introduits par Ramchandani [57] qui associent une durée de tir à chaque transition. Ces réseaux ainsi que divers modèles similaires sont essentiellement utilisés pour l'analyse de performances.
- Les réseaux de Petri *temporels* introduits par Merlin [53] qui associent un intervalle de tir aux transitions contraignant leurs instants de tir.

Pour les réseaux de Petri temporisés, les temporisations ont d'abord été associées aux transitions (T-temporisés), puis aux places (P-temporisés). D'autre part, les principales classes de réseaux de Petri temporels sont les réseaux de Petri T-temporels [5], P-temporels [39] et A-temporels [33] [1] [25] où un intervalle de temps est associé respectivement aux transitions, aux places et aux arcs. De plus, il existe deux sémantiques pour ces modèles temporels : la sémantique faible pour laquelle les bornes temporelles supérieures peuvent être dépassées et la sémantique forte permettant la modélisation de l'urgence (i.e. les bornes supérieures ne peuvent pas être dépassées). Une transition est dite urgente dans un état donné, si elle ne peut pas être retardée. Ceci constitue une caractéristique essentielle pour les systèmes temps réel critiques.

Les RdP T-temporisés et P-temporisés sont équivalents. En revanche, ces classes de réseaux de Petri temporisés sont incluses dans les classes de réseaux de Petri temporels correspondantes [54]. Par conséquent, les systèmes temporels sont plus expressifs que les systèmes temporisés.

1.3 Contribution

Ce travail de thèse est centré sur des extensions des réseaux de Petri récursifs. Les principales contributions sont :

- **Extension temporelle des réseaux de Petri récursifs** : Les réseaux de Petri récursifs constitue un excellent modèle pour la description de systèmes complexe à structure dynamique pour lesquels la création d'un nombre *dynamique* de processus est requise. Afin de spécifier et d'analyser les *comportements temporels* de ces systèmes, nous avons pensé à étendre les réseaux récursifs avec le temps. Après une étude approfondie des systèmes temporisés, nous avons choisi d'adopter la sémantique des réseaux T-temporels aux réseaux récursifs. Bien que les T-temporels ne soient pas les réseaux les plus expressifs, ils sont très utilisés en raison de leur puissance d'expression condensée du parallélisme et leur adéquation au problème de la modélisation des applications temps réel des systèmes embarqués. Leur puissance réside principalement dans la *fiabilité* et l'*efficacité* de leurs outils d'analyse tels que ROMEO [27] et TINA [6]. Par ailleurs, ils constituent, par rapport aux autres variantes des réseaux temporels, un bon compromis entre puissance d'expression et outils d'analyse. Ainsi, les *Réseaux de Petri Récursifs Temporels* (RdPRT) que nous proposons dans cette thèse combinent les sémantiques des réseaux récursifs et des réseaux T-temporels. Dans ce modèle mixte, un intervalle de tir est associé non seulement à chaque transition (élémentaire ou abstraite) mais aussi à chaque coupure. Les tirs des transitions peuvent alors avoir lieu pendant un intervalle de temps à partir du moment où toutes leurs préconditions sont satisfaites. Alors que le tir d'une coupure peut alors avoir lieu pendant un intervalle de temps à partir du moment où un marquage final est atteint sensibilisant la coupure. Nous avons donné une sémantique formelle et claire du modèle étendu et montré les capacités de modélisation de ce nouveau modèle à travers des exemples. D'autre part, nous avons développé une technique de construction des espaces d'états des RdPRT finis. Il s'agit des graphes des classes d'états étendus. Le problème de la finitude de ces graphes est indécidable comme pour les réseaux T-temporels. Nous avons été amenés à définir des conditions de finitude de ces graphes. Enfin, nous avons proposé une deuxième méthode d'analyse des RdPRT que nous avons comparée à la première méthode d'analyse proposée pour le même modèle. Les capacités de modélisation des RdPRT ont été mises en valeur grâce à la modélisation de documents multimédias en vue d'en faire un contrôle de la cohérence temporelle. Les modèles RdPRT traduisant des documents multimédias sont hiérachiques, lisibles et compacts. L'analyse de la cohérence temporelle de tels documents se fait de *manière naturelle*.

- **Extension des réseaux de Petri récursifs avec des places partagées** : Dans les réseaux de Petri récursifs, et donc dans les RdPRT, chaque processus évolue de manière *indépendante* des autres processus. Ceci est un peu contraignant en terme de modélisation puisque, dans les systèmes multi-processus, les processus ont besoin de communiquer entre eux. Nous avons alors pensé à doter les processus d'un réseau de Petri récursif de moyens de communication entre eux. C'est un problème **complexe** car une telle extension rend les propriétés les plus classiques indécidables. Notre solution consiste d'une part à introduire la notion de places partagées par l'ensemble des processus, et d'autre part à définir des contraintes sémantiques à ce nouveau modèle étendu. Nous avons développé une procédure d'accessibilité pour ce modèle. Aussi, le modèle de temps des RdPRT peut être greffé naturellement à ce nouveau modèle augmentant ainsi les capacités de modélisation du nouveau modèle temporel.

1.4 Organisation du Document

Dans le chapitre 2, nous introduisons les réseaux de Petri récursifs qui font l'objet de cette thèse. Nous rappelons ainsi leurs sémantiques, principes de fonctionnement et résultats de décidabilité.

Le chapitre 3 donne une description du modèle des réseaux de Petri T-temporels qui fait également l'objet de cette thèse.
Dans le chapitre 4, une présentation générale d'autres modèles temporisés (automates temporisés, les P-temporels et les A-temporels) est donnée. Dans ce chapitre, nous exposons des travaux comparant l'expressivité des réseaux de Petri T-temporels et des autres modèles temporisés décrits dans le même chapitre.
Le chapitre 5 expose la démarche que nous avons retenue pour étendre les réseaux de Petri récursifs avec le temps. Nous donnons la syntaxe et la sémantique formelle du modèle temporel proposé (RdPRT). Dans ce chapitre, une application simple modélisant la prise en charge de transactions par un serveur distant illustre les capacités de modélisation des RdPRT. Un algorithme pour la construction du graphe des classes d'états étendu d'un RdPRT est présenté dans ce chapitre. Des propriétés théoriques des RdPRT sont également développées dans ce chapitre.
Le chapitre 6 présente des modèles RdPRT dédiés à des documents multimédias afin d'en faire un contrôle de la cohérence temporelle.
Le chapitre 7 propose une deuxième extension des réseaux de Petri récursifs qui permet de faire communiquer les processus d'un réseau via des places partagées. Le résultat principal de ce chapitre réside dans l'élaboration d'une procédure d'accessibilité pour ce nouveau modèle étendu.

Chapitre 2

Les Réseaux de Petri Récursifs

2.1 Introduction

Les réseaux de Petri permettent d'avoir une vue à la fois des états et des actions du système. Un réseau de Petri comporte des places (représentées par des cercles) et des transitions (représentées par des traits). Les places représentent une partie de l'état du système et contiennent des jetons indiquant le nombre d'occurrences de ce sous-état. Les transitions représentent les évènements pouvant avoir lieu. Les arcs en entrée d'une transition déterminent les conditions devant être satisfaites pour que l'action puisse avoir lieu (i.e. la pré-condition au franchissement de la transition). De même, les arcs en sortie de la transition indiquent la post-condition, i.e. le résultat du franchissement.

Les systèmes à modéliser étant de taille de plus en plus importante, le modèle de réseaux de Petri a évolué vers des réseaux de haut niveau. Des études théoriques ont étudié l'impact des extensions des réseaux de Petri sur les propriétés fondamentales de ces derniers. En particulier, la décidabilité du problème d'accessibilité est étudiée ; notons que cette propriété est *fondamentale* pour pouvoir définir un *outil de vérification*. Parmi les modèles des réseaux de Petri étendus , nous nous sommes intéressés, dans le cadre de cette thèse, aux *réseaux de Petri récursifs* [28] [31] [29] [30] [32] . Le modèle des réseaux de Petri récursifs (RdPR) a été introduit pour la modélisation des plans des agents dans les systèmes muti-agents [61] [60]. D'autre part, les RdPR permettent la modélisation de systèmes *dynamiques* pour lesquels la création de processus est requise. Dans un RdPR, l'ensemble des transitions est réparti en deux catégories : les transitons *élémentaires* et les transitions *abstraites*. A chaque transition abstraite est associé un marquage ordinaire, appelé *le marquage de départ* de la transition. D'autre part, des ensembles de marquages ordinaires, dits *des ensembles de terminaison*, sont associés à un RdPR et tout marquage appartenant à un de ces ensembles est dit *final*. Par ailleurs, un marquage final définit un état dans lequel un processus peut se terminer. Un ensemble de terminaison est caractérisé par un *indice* et doit être spécifié sous la forme d'une famille de représentations effective d'ensembles *semi-linéaires* de marquages. Un RdPR peut être muni de plusieurs ensembles de terminaison. L'indice associé à chacun de ces ensembles permet de distinguer les différents

types de terminaison possibles.

Alors qu'un réseau de Petri ordinaire est constitué d'un seul processus dont l'état change suite aux tirs des transitions élémentaires, un RdPR est constitué d'un arbre de processus ; chacun est caractérisé par un marquage ordinaire. Les processus dans les RdPR sont créés par des transitions abstraites. En effet, le tir d'une transition abstraite implique la création d'un nouveau processus, qui *débute* son exécution avec le *marquage de départ* de la transition abstraite. Du point de vue du marquage courant du processus dans lequel la transition est tirée, ce franchissement consiste à consommer les jetons spécifiés par la pré-condition de la transition abstraite. Lorsque le processus créé atteint un marquage final, il génère, dans le marquage du processus père, les jetons spécifiés par la post-condition de la transition abstraite qui lui a donné naissance puis met fin à son existence. Ce type de pas est appelé une *coupure*. Notons que la post-condition d'une transition abstraite dépend de l'indice de l'ensemble de terminaison contenant le marquage final atteint par le processus créé par la transition abstraite.

Une transition abstraite peut être sous le contrôle d'une transition élémentaire (concept de préemption). En effet, quand un processus Th tire une transition élémentaire d'une part, son marquage ordinaire est modifié de manière classique et d'autre part, ses processus descendants, ayant été créés par des transitions abstraites qui sont sous le contrôle de la transition élémentaire tirée, sont détruits. De plus, pour chaque processus détruit, le processus Th génère, au sein de son propre marquage, les jetons de sorties de la transition abstraite ayant donné naissance à ce processus détruit. Ici, des indices (parfois additionnels) sont utilisés pour préciser les arcs de sortie des transitions abstraites (préemptées) à utiliser.

Dès qu'un nouveau modèle est proposé, une question cruciale se pose : il s'agit de savoir si le modèle est une réelle extension des réseaux de Petri ou simplement une abréviation. Par exemple, le modèle des réseaux de Petri colorés avec un domaine fini de couleurs est une simple abréviation alors qu'avec un domaine de couleurs infini est une réelle extension [36]. Dans le cas des RdPR, il a été montré que ce modèle est une extension stricte des réseaux de Petri ordinaires [29]. Dans cet article, on montre que le modèle des RdPR reconnaît le langage des mots palindromes. Or, il a été montré que le modèle des réseaux de Petri ne reconnaît pas ce type de langage ; la preuve est donnée dans [35]. D'autre part, il a aussi été montré que les problèmes de l'accessibilité, la bornitude et la finitude des RdPR restaient décidables [32][29].

2.2 Description des Réseaux de Petri Récursifs

2.2.1 Préliminaires

Comme le modèle des RdPR est basé sur la notion d'ensembles semi-linéaires de marquages, nous introduisons brièvement leur définition. Un ensemble semi-linéaire de marquages est un sous-ensemble de

$\mathbb{N}^d$, avec $\mathbb{N}$ l'ensemble des entiers naturels et d un entier naturel non nul, défini comme une union finie d'ensembles *linéaires* de marquages. Un ensemble linéaire L est défini par un marquage m_0 et un ensemble fini de marquages $\{m_1, ..., m_k\}$ tel que $L = \{m | \exists(\lambda_1, ..., \lambda_k) \in \mathbb{N}^k, m = m_0 + \Sigma_{i=1,...,k} \lambda_i.m_i\}$. Une représentation effective est n'importe quelle représentation qui peut être ramenée (par un algorithme) à cette représentation standard. Par exemple, un système d'(in)équations définies sur le marquage des places est une représentation effective. Par ailleurs, étant donnée une représentation effective d'ensembles semi-linéaires, les opérations suivantes sont calculables : l'union, l'intersection, la projection et la complémentation. De plus, le test d'appartenance à un ensemble semi-linéaire est décidable (voir [24] pour plus de détails).

2.2.2 Syntaxe des Réseaux de Petri Récursifs

Nous présentons la syntaxe des RdPR correpondant à une version intermédiaire entre celle donnée dans [29] et celle dans [32].

Définition 2.1. (Réseau de Petri Récursif) Un réseau de Petri récursif est défini par un tuple $R = \langle P, T, I, W^-, W^+, \Omega, \Upsilon, K \rangle$ où :

1. P est un ensemble fini de places,
2. T est un ensemble fini de transitions, disjoint de P ($P \cap T = \emptyset$). Une transition de T est soit élémentaire ou abstraite. T_{el} et T_{ab} désignent respectivement l'ensemble des transitions élémentaires et abstraites,
3. $I = I_C \cup I_P$ est un ensemble fini d'indices, dédiés aux coupures et préemptions,
4. $W^- : P \times T \rightarrow \mathbb{N}$ est une fonction d'incidence arrière, reliant des places à des transitions,
5. $W^+ : P \times [T_{el} \cup (T_{ab} \times I)] \rightarrow \mathbb{N}$ est une fonction d'incidence avant, reliant des transitions à des places,
6. $\Omega : T_{ab} \rightarrow \mathbb{N}^P$ est une fonction de marquage de départ qui associe à chaque transition abstraite un marquage ordinaire (i.e. un élément de $\mathbb{N}^P$),
7. Υ est une famille indexée par I_C d'ensembles de terminaison. Chaque ensemble est spécifié sous la forme d'une représentaion effective d'ensembles semi-linéaires de marquages finaux.
8. $K : T_{el} \times T_{ab} \rightarrow I_P$ est une fonction partielle de contrôle qui permet de modéliser le concept de préemption externe.

Représentation d'un RdPR :
Considérons le RdPR présenté dans la partie gauche de la figure 2.1. Les éléments nouveaux par rapport au formalisme des réseaux de Petri ordinaires sont les suivants :

- La transition t_2 : cette transition est dite abstraite et représentée par un rectangle à bord double. Le franchissement de cette transition crée un nouveau processus (dont la structure de contrôle est le même RdPR). Le processus créé débute son exécution à partir du marquage initial (P_7) associé à la transition abstraite et représenté entre parenthèses. Du point de vue du marquage courant du processus dans lequel la transition abstraite est tirée, ce franchissement consiste à consommer les jetons désignés par le pré-condition de la

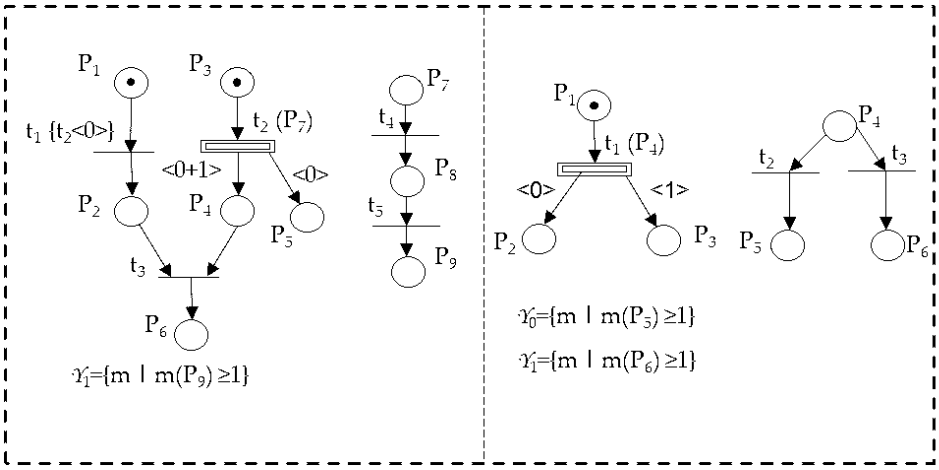

FIG. 2.1 – Deux réseaux de Petri récursifs simples

transition (ici un jeton de la place P_3).

- L'ensemble Υ_1 : c'est un ensemble de marquages ordinaires qui définit des configurations dans lequel les processus peuvent se terminer en effectuant une coupure. Un tel ensemble est appelé un ensemble de terminaison et représenté sous la forme d'une famille de représentations effective d'ensembles semi-linéaires de marquages. Dans notre exemple, l'ensemble Υ_1 est constitué de tous les marquages pour lesquels la place P_9 contient au moins un jeton. Un RdPR peut être doté de plusieurs ensembles de terminaison (non nécessairement disjoints). Les indices associés à ces ensembles permettent de distinguer les différents types de terminaison possibles. Dans notre exemple, un seul type de terminaison associé aux coupures est supporté et caractérisé par l'indice 1.

- $t_1\{t_2\langle 0\rangle\}$: cette notation spécifie d'une part que la transition abstraite t_2 est sous le contrôle de la transition élémentaire t_1 et d'autre part, que l'indice de terminaison de t_2 est nul si celle-ci est préemptée par t_1. D'une manière générale, le nom d'une transition élémentaire t peut être suivi par un ensemble termes $t'\langle i'\rangle \in T_{ab} \times I_P$. Chaque terme spécifie d'une part une transition abstraite t' qui est sous le contrôle de t et d'autre part, un indice de terminaison i' utilisé quand t' est préemptée par t (dans ce cas la valeur $K(t,t')$ est nécessairement définie et i' correspond à la valeur $K(t,t')$).

- Les valuations $\langle 0+1\rangle$ et $\langle 0\rangle$ portées respectivement par l'arc reliant la transition t_2 à la place P_4 et celui reliant la transition t_2 à la place P_5 : elles permettent de définir deux types de terminaison possibles. Un premier indicé par 0 qui peut être atteint que par une préemption (i.e. $I_P = \{0\}$) ; un deuxième caractérisé par l'indice 1 qui, pour sa part, peut être atteint que par le biais d'une coupure (i.e. $I_C = \{1\}$). Notons que la post-condition d'une transition abstraite dépend de l'indice associé au type de la terminaison utilisé. Ainsi, dans notre exemple, la terminaison, par le biais d'une coupure (i.e. correspondant à l'indice 1), d'un processus créé suite à un franchissement de t_2 conduit à la production, dans le marquage du processus père, d'un jeton dans la place P_4. En revanche, la préemption d'un tel processus (i.e. il s'agit d'une terminaison caractérisée par l'indice 0) met, dans le marquage du père, un jeton dans la place P_4 et un second dans la place P_5. Il est à noter que l'arc liant t_2 à P_4 concerne les deux types de terminaison. Ceci est spécifié

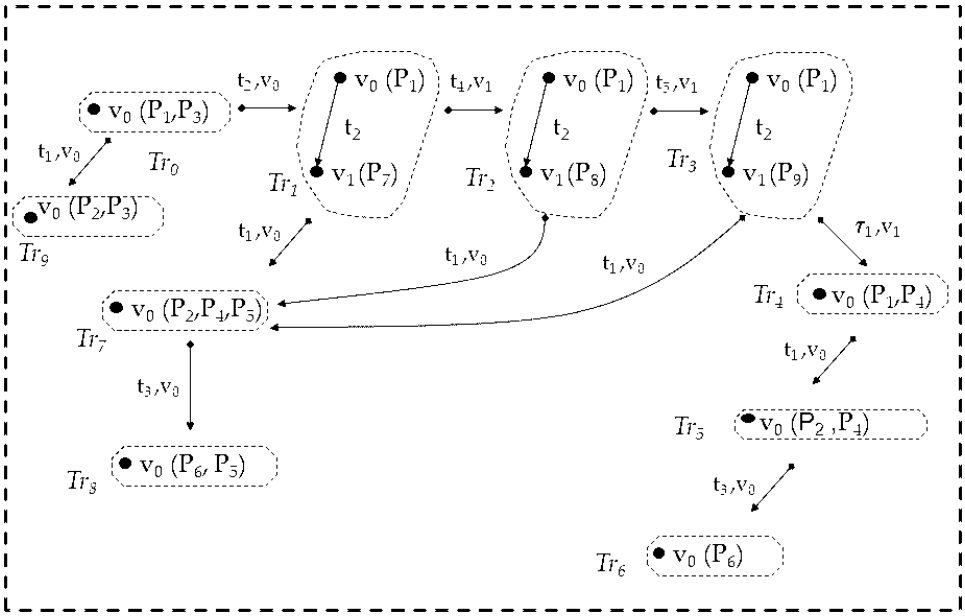

FIG. 2.2 – Séquences de tir d'un RdPR

dans le réseau par la valuation $\langle 0+1 \rangle$ portée par cet arc. Le deuxième arc sortant de la transition abstraite concerne uniquement la terminaison indicée par 0 et porte donc la valuation $\langle 0 \rangle$. D'une manière générale, un arc sortant d'une transition abstraite est étiqueté par la somme symbolique $\langle \sum_{i \in I} W^+(p,t,i).i \rangle$. Comme pour les réseaux de Petri classiques, le terme $W^+(p,t,i).i$ est omis quand il est nul et ce terme est abrégé à i lorsqu'il est égal à un.

2.2.3 Marquage Étendu d'un RdPR

Un état d'un RdPR mémorise l'état local de chaque processus. De plus, l'identité de la transition abstraite ayant conduit à la création de chacun (à l'exception du processus initial) est aussi mémorisé. Dans un RdPR, il existe deux types de marquages : les marquages étendus et les marquages ordinaires. Un marquage étendu définit l'état d'un RdPR. Un marquage ordinaire, pour sa part, représente un contexte d'exécution d'un processus. En d'autres termes, un état d'un RdPR est représenté par la liste des processus le composant (représenté sous la forme d'un arbre), où chaque processus est étiqueté par son marquage courant (un marquage ordinaire). Le nom de la transition abstraite ayant donné naissance à un processus est noté sur l'arc reliant le processus père à son fils. Plus formellement, un marquage étendu est défini comme suit :

Définition 2.2. (Marquage étendu) Un marquage étendu d'un RdPR $R = \langle P, T, I, W^-, W^+, \Omega, \Upsilon, K \rangle$ est un arbre étiqueté dirigé de la racine aux feuilles $Tr = \langle V, M, E, A \rangle$ tel que :

1. V est un ensemble fini de noeuds[1]. Quand il n'est pas vide, $v_0 \in V$ désigne la racine de l'arbre,
2. $M : V \rightarrow \mathbb{N}^P$ est une application associant un marquage ordinaire à chaque noeud,

[1]Cet ensemble peut être vide.

3. $E \subseteq V \times V$ est l'ensemble des arcs,
4. $A : E \rightarrow T_{ab}$ est une application associant une transition abstraite à chaque arc.

Notations et conventions :

1. Pour chaque noeud $v \in V$, $Succ(v)$ représente l'ensemble de ses successeurs directs et indirects incluant v. De plus, quand v n'est pas la racine de l'arbre, $pred(v)$ désigne son ***unique*** prédecesseur dans l'arbre.
2. Le marquage étendu initial est toujours noté par Tr_0 et le marquage étendu vide est caractérisé par l'ensemble vide V et noté par $\perp$.
3. Etant donné un marquage étendu, un chemin est noté par $v_0 \xrightarrow{t_1} v_1 \xrightarrow{t_2} ... \xrightarrow{t_n} v_n$ si et seulement si $\forall i \in 0..n-1$, (v_i, v_{i+1}) est un arc du marquage étendu étiqueté par la transition abstraite t_{i+1}. n est la longueur de ce chemin.
4. Quand $M(v) \in \Upsilon_i$ (avec $i \in I_C$), une coupure peut être effectuée à partir de v. Cette opération est dénotée par τ_i.
5. $Reach(R, m_0)$ donne l'ensemble des marquage étendus accessibles dans le RdPR marqué (R, m_0), où m_0 est le marquage initial ordinaire des places de R. D'autre part, $V(Tr)$ désigne l'ensemble des noeuds du marquage étendu Tr.
6. Un pas (ou un évènement) d'un RdPR marqué correspond soit au tir d'une transition élémentaire ou abstraite, soit au tir d'une coupure.

Définition 2.3. (Transition ou coupure sensibilisée) Une transition t est sensibilisée dans un noeud v (ou pour le marquage ordinaire $M(v)$) d'un marquage étendu $Tr \neq \perp$ (noté par $Tr \xrightarrow{t,v}$) si $\forall p \in P, M(v)(p) \geq W^-(p,t)$ et une coupure τ_i (avec $i \in I_C$) est sensibilisée dans v (ou pour le marquage ordinaire $M(v)$) (noté par $Tr \xrightarrow{\tau_i,v}$) si $M(v) \in \Upsilon_i$.

Définition 2.4. (Tir d'une transition abstraite) Le tir d'une transition abstraite t_f à partir d'un noeud v d'un marquage étendu $Tr = \langle V, M, E, A\rangle$ conduit au marquage étendu $Tr' = \langle V', M', E', A'\rangle$ (noté par $Tr \xrightarrow{t_f,v} Tr'$) tel que :
pour v' un identificateur nouveau, les cinq points suivants sont vérifiés :
1. $\forall p \in P$, $M(v)(p) \geq W^-(p, t_f)$,
2. $V' = V \cup \{v'\}$, $E' = E \cup \{(v, v')\}$, $\forall e \in E$, $A'(e) = A(e)$, $A'((v, v')) = t_f$,
3. $\forall v" \in V \setminus \{v\}$, $M'(v") = M(v")$,
4. $\forall p \in P$, $M'(v)(p) = M(v)(p) - W^-(p, t_f)$,
5. $M'(v') = \Omega(t_f)$.

Définition 2.5. (Tir d'une transition élémentaire) Le tir d'une transition élémentaire t_f à partir d'un noeud v d'un marquage étendu $Tr = \langle V, M, E, A\rangle$ conduit au marquage étendu $Tr' = \langle V', M', E', A'\rangle$ (noté par $Tr \xrightarrow{t_f,v} Tr'$) tel que :
pour $E" = \{(v, v') \in E \mid$ la valeur de $K(t_f, A((v, v')))$ est définie $\}$ et $V" = \{v' \in V | (v, v') \in E"\}$[2], les cinq points suivants sont vérifiés :

[2] $E"$ désigne l'ensemble des arcs partant de v et étiquetés par des transitions abstraites qui sont sous le contrôle de t_f. $V"$, pour sa part, est l'ensemble des noeuds d'arrivée des arcs de $E"$.

1. $\forall p \in P, M(v)(p) \geq W^-(p, t_f)$,
2. $V' = V \setminus (\cup_{v' \in V"} Succ(v'))$, $E' = E \cap (V' \times V')$,
3. $\forall e \in E', A'(e) = A(e)$,
4. $\forall v' \in V' \setminus \{v\}, M'(v') = M(v')$,
5. $\forall p \in P, M'(v)(p) = M(v)(p) - W^-(p, t_f) + W^+(p, t_f) + \sum_{e \in E"} W^+(p, A(e), K(t_f, A(e)))$.

Définition 2.6. (Tir d'une coupure) Le tir d'une coupure τ_i à partir d'un noeud v d'un marquage étendu $Tr = \langle V, M, E, A \rangle$ survient quand $M(v) \in \Upsilon_i$ et conduit au marquage étendu $Tr' = \langle V', M', E', A' \rangle$ (noté par $Tr \xrightarrow{\tau_i, v} Tr'$) tel que :
- $(v = v_0)$ implique que $Tr' = \perp$,
- $(v \neq v_0)$ implique les trois points suivants :
1. $V' = V \setminus Succ(v)$, $E' = E \cap (V' \times V')$, $\forall e \in E'$, $A'(e) = A(e)$,
2. $\forall v' \in V \setminus \{pred(v)\}$, $M'(v') = M(v')$,
3. $\forall p \in P$, $M'(pred(v))(p) = M(pred(v))(p) + W^+(p, A(pred(v), v), i)$.

Exemple 2.1. (RdPR avec deux ensembles de terminaison) Le RdPR de la partie droite de la figure 2.1 comporte deux ensembles de terminaison (Υ_0 et Υ_1). Le tir de la transition t_1 crée un processus Th qui débute son exécution à partir du marquage initial (P_4). Quand un tel processus atteint un marquage où la place P_5 (resp. P_6) contient au moins une marque (i.e. son marquage appartient à Υ_0 (resp. à Υ_1)), il peut effectuer une coupure τ_0 (resp. τ_1) en se détruisant. Le tir de τ_0 (resp. τ_1) permet de générer les sorties de t_1 (la transition qui a été à l'origine de la création de Th) indicées par 0 (resp. 1) ; il s'agit de P_2 (resp. P_3).

Exemple 2.2. (Graphe des marquages étendus) La figure 2.2. présente une partie du graphe des marquages étendus du RdPR marqué de la partie gauche de la figure 2.1. Il est évident que Tr_0 est le marquage étendu initial du réseau. On peut remarquer que le tir de la transition abstraite t_2 à partir du noeud v_0 de Tr_0 conduit au marquage étendu Tr_1, qui contient un nouveau noeud v_1 marqué par le marquage de départ de t_2. Par la suite, le tir de la transition élémentaire t_4 à partir de v_1 dans Tr_1 conduit au marquage étendu Tr_2, ayant une structure identique à celle de Tr_1 et, en particulier le même ensemble de noeuds. De plus, seul le marquage ordinaire associé à v_1 est modifié par ce tir. Le même raisonnement est valable pour le tir de t_5 à partir de v_1 dans Tr_2, donnant lieu à Tr_3. Nous pouvons aussi remarquer qu'un noeud peut être supprimé. Ainsi, le noeud v_1 est supprimé soit par le tir de la coupure τ_1 à partir de v_1 de Tr_3, soit par le tir de la transition t_1 à partir de v_0 de Tr_1, Tr_2 ou Tr_3. Dans le premier cas, le processus associé à v_1 a nécessairement atteint un marquage final, i.e. un marquage ordinaire tel que la place P_9 soit marqué (en d'autres termes $M(v_1)(P_9) \geq 1$). Alors que dans le second cas, le marquage de ce processus n'est pas forcément final (préemption). De plus dans les deux cas, la suppression de v_1 correspond à la destruction du processus en question et la génération des résultats de la transition t_2 ; cette production de jeton(s) concerne le contexte d'exécution du processus père (i.e. celui associé à v_0) (voir les marquages de v_0 dans Tr_7 et Tr_4). Notons que la longueur d'un chemin au sein de tout marquage étendu accessible de cet exemple est inférieure ou égal à 1.

2.3 Applications

Comme nous l'avons déjà mentionné, les RdPR ont été utilisés pour la modélisaion des plans des agents des systèmes multi-agents. Ici, nous allons passer en revue deux exemples modélisés à l'aide des RdPR et qui mettent en évidence leur puissance d'expression.

2.3.1 Programmes Dirigés par des Buts

Avec l'intelligence artificielle, de nouveaux paradigmes de programmation ont vu le jour avec leurs langages respectifs (on peut citer le langage Prolog). L'exécution d'un programme écrit dans ce style de programmation consiste à appliquer une succession de règles jusqu'à ce qu'un certain prédicat soit satisfait. On dit alors que l'exécution d'un tel programme est *dirigée par un but* ; dans ce cas, un ensemble de processus coopèrent dans le but de satisfaire un même but. Dès qu'il est satisfait par un des processus, tous les processus doivent terminer.
Le RdPR présenté dans la figure 2.3, présenté dans [29], modélise un programme dirigé par un but. Ce dernier est atteint lorsque le RdPR atteint le marquage étendu vide (i.e. $\perp$), et ceci à partir de son marquage initial composé d'un seul noeud marqué par $P_1 + P_2$. Il est à noter qu'à partir de cet état, les deux transitions abstraites t_1 et t_2 sont franchissables et leurs tirs respectifs conduisent à la création de deux processus indépendants. Aussitôt qu'un des deux processus termine son exécution, la place P_3 devient marquée. Une coupure au sein du processus racine est alors possible donnant lieu au marquage étendu vide $\perp$ (i.e. le but du programme est atteint). Nous pouvons remarquer qu'un des deux processus exécute un programme *simple* représenté par la transition élémentaire t_5. En revanche, le second processus exécute soit un programme simple (voir la transition élémentaire t_3), soit un *appel récursif* (voir la transition abstraite t_4).

2.3.2 Reprise sur Faute

Dans [32], une modélisation d'un système tolérant aux fautes est donnée ; elle comporte deux composants : un premier dédié au système nominal et un second au mécanisme de réparation. Notons que les auteurs se sont limités à une modélisation très simplifiée. La partie droite de la figure 2.4 correspond au système nominal ; l'activation de ce système se fait par le franchissement de la transition t_{start} et implique alors la création d'un processus fils chargé d'exécuter infiniment des instructions (la transition élémentaire t_{count}). Le marquage de la place P_{count} représente le nombre d'instructions ayant été exécutées depuis la création du processus fils.
La partie gauche de la figure 2.4. décrit le mécanisme de réparation. À partir de l'état initial, la transition t_{start} est franchie et l'exécution des instructions est alors prise en charge par le nouveau processus fils. Ce processus "exécute" les instructions infiniment mais peut, à tout moment *terminer* en franchissant une coupure (le marquage du processus créé est toujours final puisque la place P_{fault} est toujours marquée dans le contexte du processus fils). En d'autres termes, l'occurrence d'une *faute* est toujours possible. Du point de vue du marquage courant du processus père, après le franchissement d'une coupure, la place P_{repair} devient marquée signifiant que le système est en réparation. Quand la transition t_{repair} est franchie, le système

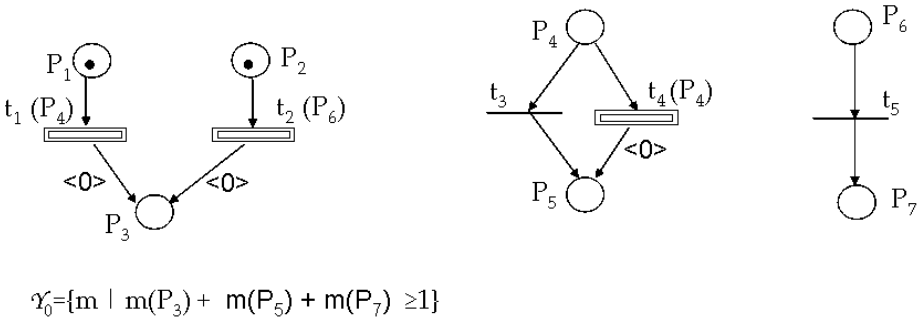

FIG. 2.3 – Modélisation d'un programme dirigé par un but

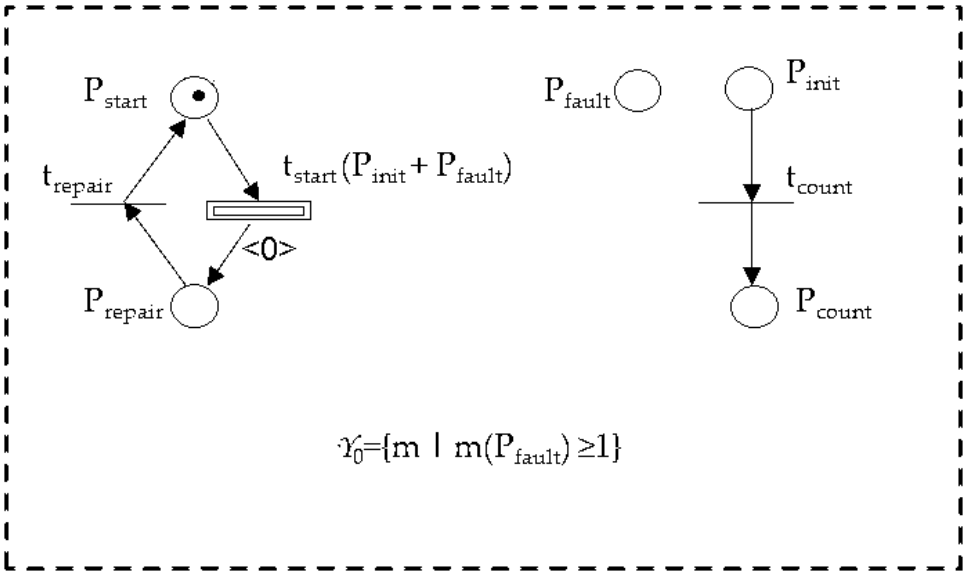

FIG. 2.4 – Modélisation d'un système tolérant aux fautes

devient opérationnel (i.e. le système nominal peut être activé à nouveau puisque la transition t_{start} devient franchissable).

Les états du RdPR de la figure 2.4 sont constitués soit d'un seul processus, soit d'un processus père et d'un fils. Le graphe d'accessibilité est présenté dans la figure 2.5. L'état initial de ce réseau est tr_{start} (composé d'un seul noeud marqué par (P_{start})) et le tir de la transition abstraite t_{start} à partir de cet état permet de passer à l'état tr_0 (composé d'un noeud racine dont le marquage est vide et d'un noeud fils marqué par (P_{init}, P_{fault}). D'autre part, les sommets tr_i (avec $i \geq 0$) correspondent aux états pour lesquels le système a exécuté i fois la transition t_{count}. Il est clair que l'état tr_{repair} peut être atteint par chacun des états tr_i. Ceci signifie que l'état tr_{repair} peut avoir un nombre *infini* d'arcs entrants (i.e. il a un degré entrant infini). Les réseaux de Petri n'ont pas cette capacité (voir théorème 1) puisque le degré entrant de chaque état de ce type de réseau est borné par le nombre de transitions du réseau. Cette limite est aussi valable pour les algèbres des processus [32].

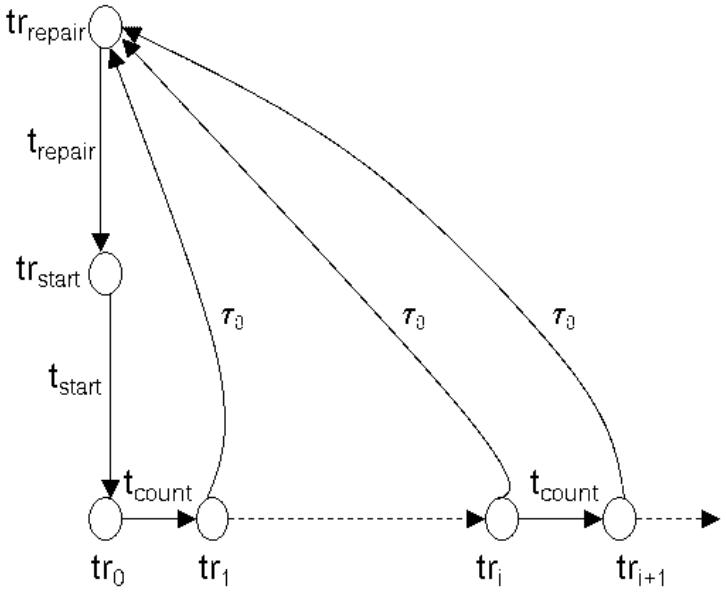

FIG. 2.5 – Graphe d'accessibilité (infini)

Theorème 2.1 (Degré entrant infini) Il n'existe aucun réseau de Petri dont le graphe de marquage soit isomorphe au graphe de la figure 2.5.

Preuve : Supposons qu'il existe un réseau de Petri ordinaire dont le graphe de marquage soit isomorphe au graphe de la figure 2.5. Soit m_i (resp. m_{repair}) le marquage de ce réseau correspondant au marquage étendu tr_i (resp. tr_{repair}). La séquence des marquages $m_0, m_1, m_2, ...$ est infinie. Il s'ensuit qu'on peut extraire une sous-séquence de *marquages strictement croissants* $m_{\alpha(0)}$,$m_{\alpha(1)}$,$m_{\alpha(2)}$,... tels que :
$\forall i < j, \forall p \in P, m_{\alpha(i)}(p) \leq m_{\alpha(j)}(p) \wedge \exists p \in P, m_{\alpha(i)}(p) < m_{\alpha(j)}(p)$.

Partant de $m_{\alpha(0)}$, on peut atteindre m_{repair} par la transition τ_0. Cette transition est aussi franchissable dans $m_{\alpha(1)}$ et conduit également à m_{repair}. Puisque $m_{\alpha(0)} \neq m_{\alpha(1)}$, les tirs de τ_0 à partir de $m_{\alpha(0)}$ et de $m_{\alpha(1)}$ conduisent forcément à *deux marquages distincts* et non au même marquage. Ceci contredit l'hypothèse que le tir de la transition τ_0 à partir de tout marquage m_i, avec $i > 0$, mène au marquage m_{repair}.

2.4 Procédure d'Accessibilité des RdPR

Etant donnés un RdPR et deux marquages étendus, le problème d'accessibilité consiste à vérifier si l'on peut atteindre l'un des deux marquages étendu (dit *destination*) à partir de l'autre (dit *source*). Ce problème a été prouvé decidable pour les réseaux de Petri ordinaire (voir [48] [43][44]). Le principe général de la procédure d'accessibilité dédiée aux RdPR consiste à ramener le problème posé à une suite de problèmes d'accessibilité dans les réseaux de Petri ordinaires. Cette procédure de décision est réalisée en deux étapes :

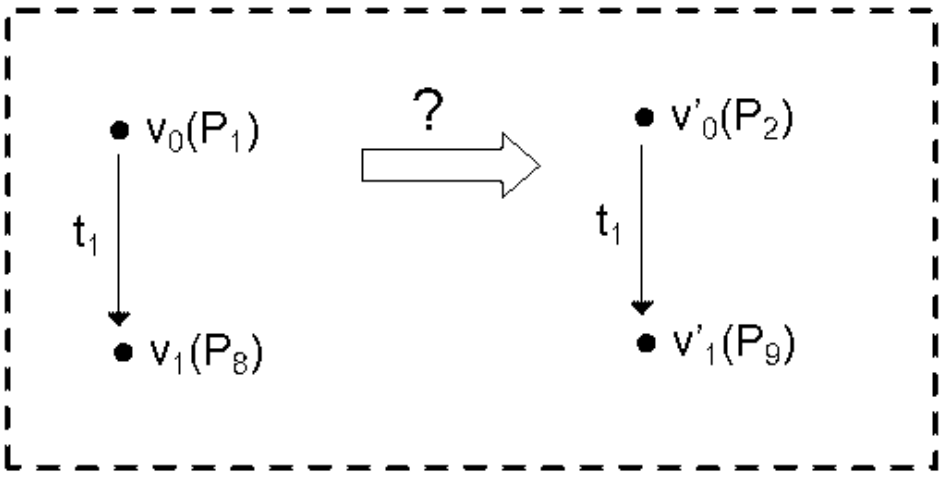

FIG. 2.6 – Prédictions possibles entre deux marquages étendus

- La première consiste à déterminer si un processus démarrant avec le marquage de départ d'une transition abstraite peut atteindre un marquage final issu d'un ensemble de terminaison donné. Une telle transition est dite "*fermable*" par rapport à cet ensemble de terminaison. Cette étude doit concerner chaque transition abstraite face à chaque ensemble de terminaison du réseau.

- La deuxième étape vise à prédire le *devenir* de chaque processus composant le marquage étendu source et aussi la *nature* des processus composant le marquage étendu destination.

2.4.1 Prédiction des Statuts des Processus

D'une manière générale, étant donnés deux marquages étendus Tr et Tr' à comparer, la procédure de décision "prédit" le devenir de chaque processus de Tr et la nature de chaque processus de Tr'. En général, le nombre de prédictions possibles que l'on peut associer aux noeuds de deux marquages étendus n'est pas forcément unique. En effet, un noeud (processus) v de Tr peut avoir deux "devenir" possibles :
- v a été supprimé et n'apparaît pas dans Tr'. Dans ce cas, on dit que v *a été fermé*.
- v existe dans Tr'. Dans ce cas, v est *persistant*.

De la même manière, un noeud v' de Tr' peut avoir deux natures possibles :
- v' n'existe pas dans Tr. Dans ce cas, on dit que v' *a été créé*.
- v' existe dans Tr et il est donc *persistant*.

Par ailleurs, il est évident que dans toute prédiction, un noeud persistant dans Tr, il doit l'être dans Tr'. Notons aussi que la racine du marquage étendu source est toujours prédite persistante. Le terme statut sera utilisé pour désigner le devenir ou la nature d'un processus. Considérons les deux marquages étendus de la figure 2.6. Nous pouvons définir seulement deux prédictions possibles :

- **Première prédiction :** v_0 est persistant (son image est v'_0). v_1 est aussi un noeud persistant et son image est v'_1.
- **Deuxième prédiction :** v_0 est persistant (son image est v'_0). v_1 a été fermé et v'_1 a été créé.

2.4.2 Première Étape de la Procédure : Transitions Abstraites Fermables

Nous nous focalisons à présent sur la détermination des transitions abstraites "fermables". Une transition abstraite est "fermable" si et seulement si un processus créé par cette transition peut, à partir du marquage de départ de la transition, atteindre un marquage appartenant à un ensemble de terminaison. Rappelons qu'un tel ensemble est semi-linéaire et que le problème d'appartenance à un ensemble semi-linéaire peut être réduit à un simple problème d'accessibilité.

Le calcul de l'ensemble F des transitions abstraites fermables se fait de manière itérative : à l'itération i, avec $i \geq 0$, l'ensemble de transitions abstraites fermables calculé est noté F_i. Initialement, on procède par le calcul de F_0 en testant si , à partir du marquage de départ de la transition abstraite à traiter, on peut atteindre un marquage final et ceci en utilisant le réseau ordinaire obtenu à partir du RdPR en supprimant toutes les transitions abstraites. Pour le calcul de F_1, les transitions abstraites appartenant à F_0 sont *simulées* par des transitions élémentaires équivalentes (ayant les mêmes pré et post conditions). Chacune de ces transitions permet de simuler la création d'un sous-processus suivi de sa terminaison. Ce processus de calcul est répété jusqu'à ce les ensembles F_i se stabilisent.
A chaque itération, la procédure traite au moins une transition abstraite. Le nombre de celles-ci est fini, ceci démontre la terminaison de la procédure.

2.4.3 Deuxième Étape de la Procédure : Comparaison des Marquages Étendus

Une fois l'ensemble de transitions abstraites "fermables" est calculé, nous considérons le réseau ordinaire obtenu à la fin du calcul itératif que nous venons de décrire. Les grandes lignes de la procédure d'accessibilité des réseaux de Petri récursifs se résument comme suit :
Soient Tr l'état source et Tr' l'état destination.

1. Si $Tr = \perp$ alors la propriété d'accessibilité est vérifiée si et seulement si $Tr' = \perp$.
2. Si $Tr \neq \perp$ et $Tr' = \perp$ alors le problème est similaire à celui consistant à décider si une transition abstraite est fermable. En d'autres termes, il s'agit de vérifier si le processus associé au noeud racine de Tr atteint un *marquage final*. Toutefois, ce processus racine n'est plus nécessairement seul dans l'arbre Tr. En conséquence, le problème se ramène à déterminer si l'ensemble de ces processus peuvent atteindre des marquages finaux. Il s'ensuit que ce problème d'accessibilité se ramène à un ensemble de problèmes de terminaison.

3. Enfin, lorsque $Tr \neq \perp$ et $Tr' \neq \perp$, la décision peut être réduite à un problème d'accessibilté dans un réseau ordinaire. Le principe consiste à chercher si il existe une prédiction des noeuds de Tr et Tr' qui peut être satisfaite. Notons que pour un processus associé à un noeud prédit fermé, on se ramène à un problème de terminaison. Alors que pour un processus associé à un noeud persistant ou créé, on se ramène à un problème d'accessibilité utilisant le réseau ordinaire construit à l'étape 1 de la procédure. Une description détaillée de cette procédure d'accessibilité est donné dans [32].

2.5 Bornitude et Finitude des RdPR

Dans cette section, nous allons nous intéresser à la bornitude et la finitude des RdPR. La propriété de bornitude assure que chaque place de n'importe quel marquage étendu accessible admet une *borne* et la propriété de finitude énonce que le nombre de marquages étendus acessibles est *fini*. Dans les réseaux de Petri classiques, ces deux propriétés sont équivalentes et décidables. Dans les RdPR, ces deux propriétés ne sont pas équivalentes mais restent décidables. Par exemple, le RdPR de la figure 2.3 est borné (toutes les places du RdPR sont bornées) mais le nombre de ses marquages étendus accessibles est infini. En effet, la transition t_4 peut être tirée une infinité de fois. Chaque tir de t_4 crée un nouveau noeud et donc un nouveau marquage étendu. Il s'ensuit que le nombre des marquages étendus est *infini*.

Proposition 2.1. (Bornitude d'un RdPR) La propriété de bortinude d'un RdPR est décidable.

Supposons qu'une place p ne soit pas bornée pour un RdPR R, alors quelle que soit une valeur entière n, il existe un marquage étendu accessible tel que le marquage de la place p, dans un certain noeud de ce marquage étendu, soit supérieur à n. Supposons que p soit non bornée dans un noeud v, nous pouvons remarquer soit que v figure dans le marquage étendu initial, soit qu'il a été créé. Dans le premier cas, le marquage initial de v correspond à son marquage dans le marquage étendu initial. Alors que dans le second cas, le marquage initial de v correspond forcément au marquage de départ de la transition abstraite qui a été à l'orgine de la création du noeud v. Ainsi la place p est non bornée dans la racine d'un sous-arbre dont l'origine est soit un noeud créé et marqué par le marquage départ d'une certaine transition abstraite , soit un noeud du marquage étendu initial. La première étape de la procédure consiste, étant donnés un RdPR marqué et une transition abstraite t, à vérifier si la transition sera *sensibilisée* dans un certain noeud d'un marquage étendu accessible (une telle transition est dite atteignable). Ce problème est réduit à celui d'accessibilité. La seconde étape de la procédure consiste à construire un ensemble $Tree$ de marquages étendus. $Tree$ contient d'une part le marquage étendu initial et tous ses sous-arbres (directs et indirects) et d'autre part, des marquages étendus dont chacun est réduit à un seul noeud marqué par le marquage de départ de toute transition abstraite atteignable. Enfin, la troisième étape de la procédure consiste à vérifier si la place p est non bornée dans la racine d'un certain marquage étendu de $Tree$. Il suffit de chercher une séquence infinie qui fait croître le marquage de la place p (pour plus de détails voir [32]).

Proposition 2.2. (Finitude d'un RdPR) La propriété de finitude d'un RdPR est décidable.

L'ensemble des marquages étendus accessibles est infini si et seulement si soit le RdPR n'est pas bor-

née, soit il est borné mais la profondeur des marquage étendus accessibles n'est pas bornée. Il a été montré dans [32] qu'un RdPR a une profondeur infinie si et seulement si il existe un marquage étendu accessible tel qu'un de ses chemins admet une répétition d'étiquettes. En d'autres termes, il existe au moins deux arcs de ce chemin labellés par une même transition abstraite. Donc, pour un RdPR borné, il suffit de décider si la profondeur des marquage étendus accessibles est bornée. Pour cela, on construit un graphe d'accessibilité jusqu'à ce que soit l'on finisse la construction du graphe soit l'on trouve un marquage étendu accessible Tr contenant deux noeuds dont l'un est descendant de l'autre et provenant des tirs de la même transition abstraite. Comme le RdPR est borné, cette construction termine (pour plus de détails voir [32]). Enfin, il est à remarquer que si la profondeur des marquages étendus accessibles d'un RdPR est infinie, alors le nombre des noeuds de ces marquages étendus accessibles est infini.

2.6 Conclusion

Dans ce chapitre, nous avons présenté le modèle des réseaux de Petri récursifs. Il s'agit d'un modèle puissant capable de prendre en charge des modélisations de systèmes complexes incluant divers mécanismes tels que les interruptions, tolérances aux fautes et les appels de procédures distants. Il a été prouvé que certains de ces systèmes ne peuvent être modélisés ni à l'aide des réseaux de Petri classiques ni à l'aide des algèbres de processus. Par ailleurs, les réseaux de Petri récursifs sont plus expressifs en termes de systèmes *infinis*. D'autre part, nous avons montré comment les propriétés de bornitude, d'accessibilité et de finitude des réseaux de Petri récursifs ont été prouvées décidables.
Les réseaux de Petri récursifs ont retenu notre attention car ils constituent un outil de modélisation puissant. C'est pourquoi nous les avons exploités afin de proposer des outils de modélisation aussi intéressants. Ce sont les des extensions de ces réseaux de Petri récursifs que nous proposons dans les chapitres 5 et 7.

Chapitre 3

Les Réseaux de Petri T-temporels

Dans ce chapitre, nous introduisons les réseaux de Petri T-temporels. Nous donnons d'abord les définitions de la syntaxique et de la sémantique du modèle. Nous montrons par la suite comment construire les graphes des classes d'états des réseaux de Petri T-temporels et présentons une synthèse des résultats de décidabilité de ce type de réseau. Enfin, nous donnons une mise en oeuvre concrète du calcul des classes d'états.

3.1 Presentation Informelle

Les réseaux de Petri T-temporels, ou Time Petri Nets (TPN) en anglais, sont une extension des réseaux de Petri introduite par Merlin en 1974 [53]. Ils étendent les réseaux de Petri classiques avec des contraintes temporelles sur les tirs des transitions. Les réseaux de Petri T-temporels consistent à ajouter du temps au modèle classique sous la forme d'un intervalle $[\alpha(t), \beta(t)]$ de $\mathbb{R}$ 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 $\alpha(t)$ et $\beta(t)$.
Le réseau de la partie gauche de la figure 3.1, issue de [26], montre un exemple de réseau de Petri T-temporel. Les places P_1 et P_3 sont marquées donc les transitions t_1 et t_2 sont *sensibilisées*, car elles possèdent un jeton dans leurs places amont. Cependant, seule t_1 est *tirable* car les intervalles de tir sont disjoints. Le tir de t_1 amène à un nouvel état dans lequel P_2 et P_3 sont marquées. Nous allons maintenant définir plus formellement les réseaux de Petri T-temporels.

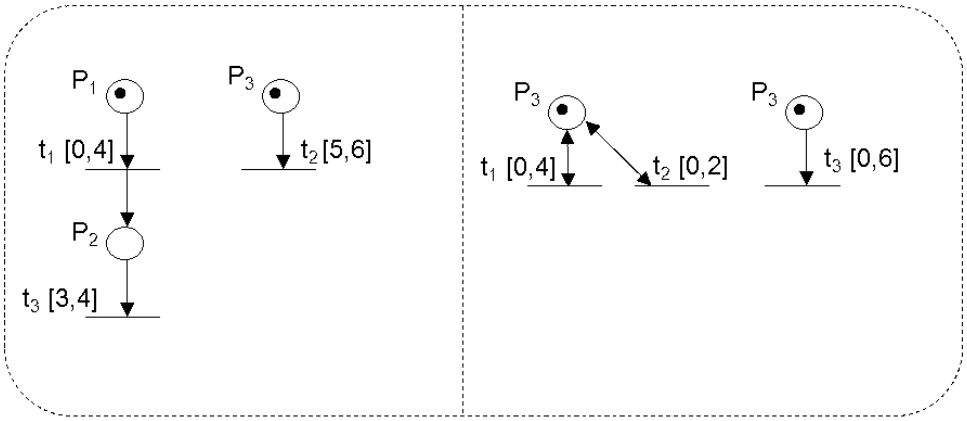

FIG. 3.1 – Deux réseaux de Petri temporels simples

3.2 Syntaxe et Sémantique des Réseaux T-temporels

3.2.1 Définitions

Définition 3.1. (Réseau de Petri T-temporel) Un réseau de Petri T-temporel est un tuple $\langle P, T, W^-, W^+, M_0, Is\rangle$, où $\langle P, T, W^-, W^+, M_0\rangle$ est un réseau de Petri ordinaire marqué avec M_0 et $Is : T \rightarrow Q^+ \mapsto (Q^+ \cup +\infty)$ est une fonction d'intervalle statique qui associe à chaque transition t du réseau un intervalle à bornes rationnelles $Is(t) = [\alpha(t), \beta(t)]$, avec $0 \leq \alpha(t) \leq \beta(t)$ ($\beta(t)$ peut être infini).
La plus petite borne de l'intervalle $Is(t)$ est appelée date de tir au plus tôt de t. La plus grande borne est appelée date de tir au plus tard de t.

Un marquage du réseau est un élément de $\mathbb{N}^P$, tel que $\forall p \in P$, $M(p)$ est le nombre de jetons dans la place p.
Une transition t est *sensibilisée* par le marquage M si $M \geq {}^\bullet t$. L'ensemble de toutes les transitions sensibilisées par le marquage M est noté $enabled(M)$. Nous notons $t \in enabled(M)$ si t est sensibilisée par le marquage M.
Une transition t est dite *nouvellement sensibilisée* par le tir de la transition t' à partir du marquage M, ce que nous noterons "$new.enabled(t, M, t')$", si t est sensibilisée par le nouveau marquage $M - {}^\bullet t' + t'^\bullet$ mais ne l'était pas par le marquage $M - {}^\bullet t'$. Formellement,
$new.enabled(t, M, t') = ({}^\bullet t \leq M - {}^\bullet t' + t'^\bullet) \wedge (({}^\bullet t > M - {}^\bullet t') \vee (t = t'))$.

Exemple 3.1. Dans le réseau de la partie droite de la figure 3.1, les transitions t_1, t_2 et t_3 sont sensibilisées par le marquage initial M_0 du réseau. Elles le sont aussi par le marquage $(M_0 - {}^\bullet t_1 + t_1^\bullet)$. Contrairement à t_2, la transition t_3 est sensibilisée par $(M_0 - {}^\bullet t_1)$. Il s'ensuit qu'après le tir de t_1 à partir du marquage initial, t_3 reste sensibilisée contrairement à t_2 qui devient *nouvellement sensibilisée*. De plus, la transition t_1 est *nouvellement sensibilisée* après son propre tir.

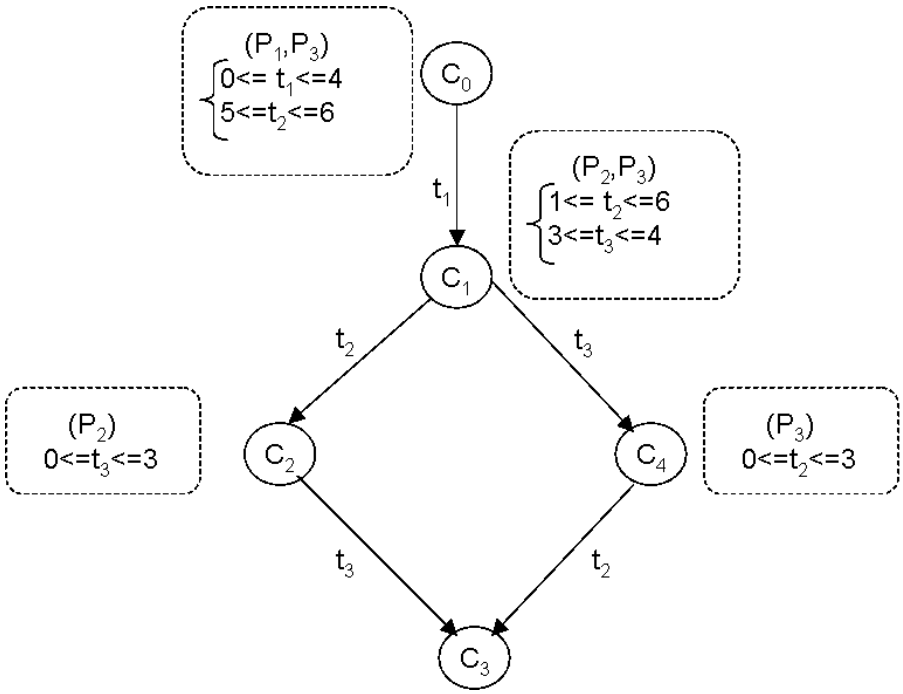

FIG. 3.2 – Exemple de graphe de classes

Définition 3.2. (Sémantique forte d'un réseau de Petri T-temporel) La sémantique forte d'un réseau de Petri T-temporel R est donnée par un système de transitions temporisé $S_R = (Q, q_0, \rightarrow)$ où :
- $Q = \mathbb{N}^P \times (\mathbb{R}^+_{\geq 0})^T$,
- $q_0 = (M_0, \overline{0})$ [1],
- $\rightarrow \in Q \times (T \cup \mathbb{R}_{\geq 0}) \times Q$ est une relation de transition incluant des transitions discrètes et des transitions continues :

1. La relation de transition discrète est définie $\forall t \in T$:

$$(M, \nu c) \xrightarrow{t} (M', \nu c') \iff \begin{cases} {}^\bullet t \leq M \\ \alpha(t) \leq \nu c(t) \leq \beta(t) \\ M' = M - {}^\bullet t + t^\bullet \\ \forall t' \in T, \nu c'(t') = \begin{cases} 0 & si\ new.enabled(t', M, t) \\ \nu c(t') & sinon. \end{cases} \end{cases}$$

2. La relation de transition continue est définie $\forall d \in \mathbb{R}_{\geq 0}$:

$$(M, \nu c) \xrightarrow{d} (M, \nu c') \iff \begin{cases} \nu c' = \nu c + d \\ \forall t \in T, M \geq {}^\bullet t \Rightarrow \nu c'(t) \leq \beta(t). \end{cases}$$

Définition 3.3. (Transition tirable) Soit $s = (M, \nu c)$ un état d'un réseau T-temporel. Une transition est tirable dans s si $M \geq {}^\bullet t$ et $\alpha(t) \leq \nu c(t) \leq \beta(t)$.

[1] $\overline{0}$ est une valuation d'horloges affectant 0 à toutes les horloges.

3.2.2 Etats et Règle de Tir

Un état d'un réseau T-temporel est un couple $E = (M, I)$ dans lequel M est un marquage et I est l'application $intervalle$ de Tir. L'état initial E_0 est constitué du marquage initial M_0 et de l'application $intervalle$ de tir I_0 qui fait correspondre à chaque transition t sensibilisée par M_0 son intervalle statique de tir $Is(t)$, et à toute autre transition non sensibilisée l'intervalle vide. Le tir d'une transition t, à une date relative θ, depuis un état $E = (M, I)$, est permis si et seulement si les conditions suivantes sont satisfaites :
(1) La transition t est sensibilisée par M, $M \geq^{\bullet} t$,
(2) θ n'est pas inférieure à la date de tir au plus tôt de t : $\theta \geq \alpha(t)$,
(3) θ n'est supérieure à la date de tir au plus tard d'aucune transition sensibilisée par M : $\forall t', M \geq^{\bullet} t' \Rightarrow \theta \leq \beta(t')$.

La première condition est classique, les deux dernières resultent de l'obligation de tirer les transitions dans leur intervalle de tir.

Définition 3.4. Deux transitions t et t' sont en conflit pour un marquage M si toutes deux sont sensibilisées par M, mais, il existe au moins une place p telle que $M(p) < W^-(p,t) + W^-(p,t')$.

Le tir d'une transition sensibilisée t, à la date θ, depuis l'état $E = (M, I)$, conduit en un état $E' = (M', I')$ déterminé comme suit :
(1) Le nouveau marquage M' est déterminé classiquement, par : $M' = M + {}^{\bullet}t + t^{\bullet}$;
(2) Le nouvel $intervalle$ de tir $I'(t')$, pour toute transition t', est défini par :
(a) Si t' est non sensibilisée par M', alors $I'(t')$ est vide,
(b) Si t' est distincte de t, sensibilisée par M, et n'est pas en conflit avec t pour M, alors
$I'(t') = [max(0, \alpha(t') - \theta), \beta(t') - \theta]$ si $\beta(t')$ est fini,
$I'(t') = [max(0, \alpha(t') - \theta), \infty[$ sinon,
(c) Sinon, $I'(t') = Is(t')$.

Notons que si t est restée sensibilisée pendant son propre tir, alors elle reçoit son intervalle statique. D'autre part, les transitions non sensibilisées par le nouveau marquage M' reçoivent des intervalles de tir vides ; les transitions distinctes de t et qui sont restées sensibilisées pendant le tir de t voient leur intervalle de tir décalé vers l'origine du temps de la valeur θ, date relative à laquelle a été tirée la transition t (et restreints si nécessaire aux valeurs de temps non négatives) ; toutes les autres transitions sensibilisées par M' reçoivent pour intervalle de tir leur intervalle statique.

Exemple 3.2. L'état initial E_0 du réseau de la partie gauche de la figure 3.1 est constitué du couple (M_0, D_0) suivant :
$M_0 : (P_1, P_3)$,
D_0 : ensemble des solutions en (t_1, t_2) du système $(0 \leq t_1 \leq 4$ et $5 \leq t_2 \leq 6)$.
Le tir de t_1 depuis l'état E_0, à une date relative θ_1 comprise dans l'intervalle [0,4], mène à l'état $E_1 = (M_1, D_1)$, avec :
$M_1 : (P_2, P_3)$,
D_1 : ensemble des solutions en (t_2, t_3) de :

$5 - \theta_1 \leq t_2 \leq 6 - \theta_1$
$3 \leq t_3 \leq 4$
Le paramètre θ_1 apparaît dans le système D_1 ci-dessus comme une constante. Comme le temps est continu, le paramètre θ_1 peut prendre toute valeur réelle dans l'intervalle [0,4]. Par conséquent, l'état E_0 admet donc une *infinité* d'états suivants par le tir de t_1 ; chaque valeur de θ_1 définit un état suivant différent des autres.
Le tir de t_3 depuis l'état E_1, à une date relative θ_2 comprise dans l'intervalle $[3, min(4, 6 - \theta_1)]$, mène à l'état $E_2 = (M_2, D_2)$, avec :
$M_1 : (P_3)$,
D_1 : ensemble des solutions en (t_2) de :
$max(0, 5 - \theta_1 - \theta_2) \leq t_2 \leq 6 - \theta_1 - \theta_2$

Un échéancier de tir est un couple (s, u) constitué d'une séquence de transitions s et d'une séquence u de dates relatives de tir. Un échéancier est *réalisable*, depuis un état E, si et seulement si toutes les transitions dans la séquence s sont successivement tirables depuis l'état E, aux dates relatives de tir qui leur correspondent dans la séquenve u [7]. Reprenons le réseau de Petri T-temporel de la partie gauche de la figure 3.1. Un exemple d'échéancier réalisable depuis l'état initial est $(t_1.t_3, 0.3)$. L'ensemble de tous les échéanciers de support t_1t_3 réalisable depuis l'état E_0 peut être décrit dans cet exemple par l'ensemble des échanciers de la forme $(t_1.t_2, \theta_1.\theta_2)$ avec $\theta_1 \in [0, 4]$ et $\theta_2 \in [3, 4]$

3.3 La Méthode des Classes d'États

La règle de tir ci-dessus définit une relation d'accessibilité sur l'ensemble des états du réseau T-temporel. Représenter le fonctionnement d'un réseau T-temporel par le graphe d'accessibilité de ses états (comme le fonctionnement d'un réseau de Petri classique est représenté par le graphe d'accessibilité de ses marquages) est en général impossible : le temps étant *continu* et les transitions pouvant être tirées à tout instant dans leur intervalle de tir. De plus, les états ainsi définis ont en général une infinité de successeurs par la règle de tir. Les *Classes d'États* définies ont pour but de fournir une *représentation finie* de cet ensemble infini d'états. MENASCHE, BERTHOMIEU et DIAZ ont été les premiers à s'intéresser à la vérification du modèle de MERLIN. Ils ont proposé, dans [22][5][52], une contraction de l'espace d'états qui produit un graphe des classes fini si et seulement si le modèle est borné.

Une classe d'états sera vue comme un couple $C = (M, D)$ dans lequel M est un marquage et D est un domaine de tir. Les inéquations de D sont de deux types [5] :

$$\begin{cases} \alpha_i \leq \theta_i \leq \beta_i \ \forall t_i \text{ tel que } t_i \in enabled(M), \\ \gamma'_{j,k} \leq \theta_j - \theta_k \leq \gamma_{j,k} \ \forall t_j, t_k \text{ tels que } t_j \in enabled(M) \text{ et } t_k \in enabled(M). \end{cases}$$

où θ_i est la date de tir de la transition t_i sensibilisée relativement à l'instant où l'on est entré dans la classe. Ces domaines admettent des formes canoniques de la même forme :
$a_i^* \leq t_i \leq b_i^*$, pour tout t_i,

$t_j - t_k \leq c^*_{j,k}$, pour tout t_i, t_k avec $j \neq k$
où a^*_i, b^*_i et $c^*_{j,k}$ désignent respectivement la plus petite valeur possible de la variable t_i, la plus grande valeur de la variable de t_i et la plus grande différence possible entre les valeurs de t_j et t_k.
Ces formes canoniques peuvent être calculées en temps polynomial. Pour celà, une technique possible est d'associer au système d'inéquations un Graphe de Contraintes[4] [56]. L'obtention de la forme canonique se réduit alors à un calcul de plus courts chemins entre toutes paires de sommets du graphe.
Soit s le système à mettre sous forme canonique ; s a la forme générale indiquée précédemment. Le graphe de contraintes a autant de sommets que le système s a de variables, plus un sommet supplémentaire src.
Le graphe est complet, les arcs sont valués par la fonction V définie comme suit :
$\forall x, V(x,x) = 0$,
si $k \leq y \in s$, alors $V(src, y) = -k$
si $x \leq k \in s$, alors $V(x, src) = k$
si $x - y \leq k \in s$, alors $V(x,y) = k$
sinon $V(x,y) = \infty$
Soit $D(x,y)$ la longueur du plus court chemin de x à y dans le graphe de contraintes valué par V ; on a alors pour tout i, j, k ($j \neq k$) :
$a^*_i = -D(t_i, src)$
$b^*_i = D(src, t_i)$
$c^*_{j,k} = D(t_j, t_k)$.
Pour le calcul des plus courts chemins, on peut par exemple utiliser l'algorithme de Floyd-Warshall [17], de complexité $O(n^3)$ en temps et $O(n^2)$ en espace (n est le nombre de sommets du graphe). Cet algorithme permet de plus de vérifier la consistance du système d'inéquations : le système s est consistant si le graphe de contraintes associé ne contient pas de cycle de poids négatif, c'est à dire si, pour tout sommet x, on a $D(x,x) \geq 0$.

Définition 3.5. (Transition tirable depuis une classe d'états) Une transition t sensibilisée est tirable depuis une classe d'états $C = (M, D)$ si et seulement si la transition t est tirée dans son intervalle de tir et qu'elle est tirée la première parmi toutes les transitions sensibilisées, en respectant les *dates de tir au plus tard* de toutes les autres. Plus formellement, on a :
Une transition t_i est tirable depuis C si t_i est sensibilisée par M et il existe une solution $(\theta^*_1, ..., \theta^*_n)$ de D telle que $\forall j \in 1...n$, $\theta^*_i \leq \theta^*_j$.

Définition 3.6. (Tir d'une transition depuis une classe d'états) Soit une classe $C = (M, D)$ et une transition tirable t_i. La classe $C' = (M', D')$, successeur de C par t_i est donnée par :

1. $M' = M - {}^\bullet t_i + t_i^\bullet$
2. Le nouveau domaine de tir, D', est calculé selon les étapes suivantes :

   (a) changements de variables $\forall j, \theta_j = \theta_i + \theta'_j$;
   (b) $\forall j \neq i$, ajout de la contrainte $\theta_j \geq \theta_i$;
   (c) élimination des variables correspondant à des transitions désensibilisées par le tir de t_i (ce qui inclut donc t_i), par exemple en utilisant la méthode de Fourier-Motzkin [21] ;
   (d) ajout des inéquations relatives aux transitions nouvellement sensibilisées par le tir de t_i : $\forall t_k$

telle que $New.enabled(t_k, M, t_i)$, alors $\alpha(t_k) \leq \theta'_k \leq \beta(t_k)$.

(e) détermination de la forme canonique du nouveau domaine de tir.

Les contraintes $\theta_j \geq \theta_i$ exprime le fait que t_i sera tirée en premier, et par conséquent, toutes les autres transitions seront tirées plus tard. En revanche, les changements de variables modélisent l'écoulement du temps pour les transitions sensibilisées, par un changement d'origine du temps : la nouvelle origine devient l'instant de tir de t_i.

3.3.1 Graphes des Classes d'États

Le calcul du graphe des classes d'états d'un réseau T-temporel commence par la détermination d'une classe initiale C_0. Cette classe se définit par $C_0 = (M_0, D_0)$, où M_0 est le marquage initial du réseau et $D_0 = \{\alpha(t_i) \leq \theta_i \leq \beta(t_i) |\ t_i \in enabled(M_0)\}$.
Pour calculer le graphe des classes d'états du réseau, il suffit de calculer itérativement tous les successeurs de la classe initiale.

Définition 3.7. (Egalité de classes d'états) Deux classes d'états $C = (M, D)$ et $C' = (M', D')$ sont dites égales si $M = M'$ et $D = D'$. Nous notons $C \equiv C'$.

Exemple 3.3. L'application de cette approche au modèle de la partie gauche de la Figure 3.1 produit le graphe des classes donné dans la figure 3.2.

3.3.2 Finitude

Le problème d'accessibilité pour un réseau T-temporel se réduit à démontrer cette propriété pour les réseaux à arcs inhibiteurs et est donc indécidable. Comme pour l'accessibilté d'un marquage, démontrer la propriété borné pour un réseau T-temporel se réduit à démontrer cette propriété pour les réseaux à arcs inhibiteurs et est donc indécidable [37].

Le problème de la finitude de l'ensemble des classes d'états d'un réseau T-temporel est donc indécidable puisque chaque classe est un couple (marquage, domaine de tir). Pour résoudre ce problème, Berthomieu et Diaz ont démontré la condition nécessaire et suffisante suivante [5].

Théorème 3.1. Le nombre de classes du graphe des classes d'états d'un réseau de Petri T-temporel T est fini si et seulement si T est borné.
Par conséquent, ce problème est indécidable car le problème de bornitude de Petri T-temporel est indécidable. Berthomieu et Diaz proposent donc également d'utiliser les conditions nécessaires de non bornitude suivantes pendant l'exécution de l'algorithme de calcul du graphe des classes [5] :

Theorème 3.2. (Condition suffisante 1 pour la propriété borné) Un réseau Temporel est borné s'il n'admet pas de paire de classes d'états $C = (M, D)$ et $C' = (M', D')$ telles que :
1. C' est accessible depuis C ;
2. $M' > M$.
Nous remarquons que ces deux conditions constituent une condition nécessaire de non bornitude. Cette condition suffisante permet d'analyser une famille importante de réseaux T-temporels mais échoue pour certains réseaux. La condition suivante est plus faible que la précédente :

Theorème 3.3. (Condition suffisante 2 pour la propriété borné) Un réseau T-temporel est borné s'il n'admet pas de paire de classes d'états $C = (M, D)$ et $C' = (M', D')$, accessibles depuis la classe initiale telles que :
1. C' est accessible depuis C ;
2. $M' > M$.
3. $D' = D$

De la même manière que précedemment, le théorème 3.3. échoue pour démontrer que certains réseaux bornés le sont. Nous énonçons une dernière condition suffisante plus faible que le théorème 3.3.

Theorème 3.4. (Condition suffisante 3 pour la propriété borné) Un réseau T-temporel est borné s'il n'admet pas de paire de classes d'états $C = (M, D)$ et $C' = (M', D')$, accessibles depuis la classe initiale telles que :
1. C' est accessible depuis C ;
2. $M' > M$;
3.) $D' = D$;
4. $\forall p \in \{p \in P, M'(p) > M(p)\}$, $M(p) > max_{t \in T}\ {}^\bullet t(p)$.

3.4 Mise en Oeuvre du Calcul des Classes

Une mise en oeuvre d'une approche de calcul des classes a été proposée dans [22]. Elle est basée sur l'utilisation de l'algorithme de FLOYD-WARSHALL [17]. Dans cette approche, la complexité de calcul d'une classe est $O(n^3)$ en temps, où n est le nombre de transitions sensibilisées par la classe (n est inférieur ou égal au nombre de transitions dans le réseau). Dans [8], une autre approche de calcul des classes, caractérisée par une complexité de $O(n^2)$, a été proposée. Dans cette approche, les domaines de tir des classes d'états sont exprimés sous la forme de matrices de différences (DBM).

3.4.1 La Matrice de Bornes de Différences

Une matrice de bornes de différences ou Difference Bounds Matrix (DBM) en anglais permet de représenter des inéquations de la forme $x_i - x_j \leq b_k$ et $x_i \leq b_k$, avec $i, j \in 1..n$ et $k \in 1..m$. Soit x_0 une variable telle que $x_0 = 0$, alors toutes les inéquations peuvent être exprimées sous la forme $x_i - x_j \leq b_k$, avec $i, j \in 0..n$.
Dans le cas où certaines contraintes sont strictes, les éléments de la DBM sont des couples $(b_k, \sim), \sim \in \{\leq, <\}$.
Soient n le nombre de variables du système et m le nombre des contantes b_k. Le calcul de la forme canonique a une complexité au pire cas polynomiale en n et m. Pour les DBM, l'égalité, l'inclusion et l'intersection ont des complexités quadratiques en n ($O(n^2)$) [45].

3.4.2 Forme Canonique de la Classe Initiale

La forme canonique de la classe d'états initiale $C_0 = (M_0, F_0)$ est le couple (M_0, D_0) composé du marquage initial M_0 et de la matrice D_0 définie comme suit :
- $\forall t \in enabled(M_0)$, $D_0[t, x_0] = -\alpha(t)$, $D_0[x_0, t] = \beta(t)$ et $D_0[t, t] = 0$;
- $\forall t, t' \in enabled(M_0) \times enabled(M_0)$ (avec $t' \neq t$), $D_0[t, t'] = D_0[t, x_0] + D_0[x_0, t']$;
- $D_0[x_0, x_0] = 0$.

3.4.3 Calcul des Transitions Franchissables

Partant d'une classe $C = (M, D)$ (sous forme canonique), le calcul des transitions franchissables peut être réalisé sans utiliser l'algorithme de FLOYD-WARSHALL ou tout autre de résolution de systèmes d'inéquations.

Proposition 3.1. La transition t_f est franchissable à partir de la classe C si et seulement si : $t_f \in enabled(M)$ et ($\forall t \in enabled(M),\ \ D[t_f, t] \geq 0$).

La preuve de cette proposition est donnée dans [8].

Calcul des classes accessibles

Soient $C = (M, F)$ une classe d'états, et $C = (M, D)$ sa forme canonique et t_f une transition franchissable à partir de la classe d'états C. La forme canonique $C' = (M', D')$ de la classe accessible à

partir de C par la transition t_f peut être calculée comme suit :

Algorithm 1 Calcul d'une classe successeur $C' = (M', D')$ suite au tir de t_f à partir de $C = (M, D)$

1: $M' = M - {}^\bullet t_f + t_f^\bullet$;
2: $D'[x_0, x_0] = 0$;
3: **for** $t \in Enabled(M')$ **do**
4: $D'[t, t] = 0$;
5: **if** $new.enabled(t, M, t_f)$ **then**
6: $D'[t, x_0] = -\alpha(t)$;
7: $D'[x_0, t] = \beta(t)$;
8: **else**
9: $D'[x_0, t] = D[t_f, t]$;
10: $D'[t, x_0] = 0$;
11: **for** $t' \in Enabled(M)$ **do**
12: $D'[t, x_0] = Min(D'[t, x_0], D[t, t'])$;
13: **end for**
14: **end if**
15: **end for**
16: **for** $t \in Enabled(M')$ **do**
17: **for** $t' \in enabled(M')$ telle que $t' \neq t$ **do**
18: **if** $new.enabled(t, M, t_f)$ ou $new.enabled(t', M, t_f)$ **then**
19: $D'[t, t'] = D'[t, x_0] + D'[x_0, t']$;
20: **else**
21: $D'[t, t'] = Min(D[t, t'], D'[t, x_0] + D'[x_0, t'])$;
22: **end if**
23: **end for**
24: **end for**
25: **END**

3.4.4 Analyse du Graphe des Classes

Les propriétés linéaires du graphe d'états (i.e. : du modèle) sont conservées dans le graphe des classes d'états. Par conséquent, pour vérifier si un modèle possède bien une propriété conservée, il suffit de vérifier si son graphe des classes possède bien cette propriété. Les propriétés concernant les marquages ou les séquences de transitions franchissables peuvent être donc vérifiées en explorant le graphe des classes de la même manière que ces propriétés sont prouvées à l'aide du graphe des marquages d'un réseau de Petri. En ce qui concerne les propriétés temps réel (temps de réponse borné, respect des contraintes temporelles imposées) qui s'expriment généralement en temps d'exécution de séquences de franchissement, elles ne peuvent être déduites, de façon *immédiate*, du graphe des classes. De ce fait, une technique de calcul des temps d'exécution de séquences de franchissement s'impose. Dans [8], une technique permettant de calculer, par un simple parcours d'un chemin du graphe des classes, les temps *minimal* et *maximal* d'exécution

de la séquence de transitions du chemin (temps de chemin ou de cycle). Dans ce qui suit, nous montrons comment se fait ce calcul.

L'algorithme de calcul des temps minimal et maximal d'exécution de la séquence d'un chemin peut être décrit par l'algorithme 2. Cet algorithme utilise les structures de données "$Chemin$" et "$Bornes$" : $Chemin$ est une liste composée des arcs du chemin ; $Bornes$ est un couple de rationnels. Les tableaux de rationnels A et B servent à mémoriser, durant le parcours du chemin, les distances qui séparent les transitions sensibilisées pour la classe courante de l'origine x_o de la classe de départ.

Algorithm 2 Bornes TempsdExecution (Chemin ch)

1: Rationnel x, $inf = 0$, $sup = 0$;
2: Rationnel $tableauA[T]$ initialisé à 0, $B[T]$ initialisé à infini ;
3: **for** chaque arc du chemin ch (du premier au dernier) **do**
4: //Soient (M, D) la classe origine et t_f la transition de l'arc.
5: $x = infini$;
6: **for** $t \in Enabled(M)$ **do**
7: **if** il s'agit du premier arc du chemin ou t est nouvellement sensibilisée **then**
8: $A[t] = inf + D[t, x_0]$;
9: $B[t] = sup + D[x_0, t]$;
10: **else**
11: $A[t] = Min(A[t], inf + D[t, x_0)]$;
12: $B[t] = Min(B[t], sup + D[x_0, t])$;
13: **end if**
14: $x = Min(x, B[t])$;
15: **end for**
16: $sup = x$;
17: $inf = A[t_f]$;
18: **end for**
19: $retouner(-inf, sup)$;
20: **END**

Il a été prouvé que la complexité de cet algorithme est $O(m \times n)$, où m est la longueur du chemin et n est le plus grand nombre de transitions sensibilisées dans les classes du chemin. De plus, il a été aussi prouvé que cet algorithme est applicable pour calculer les temps de cycle, puisqu'un cycle est un chemin fermé (la classe de départ est égale à la classe finale) [8].

3.5 Conclusion

Dans ce chapitre, nous avons présenté les réseaux de Petri T-temporels ; ces derniers permettent de combiner plusieurs notions telles que la concurrence, la synchronisation et les contraintes temporelles. Nous

avons également présenté la méthode la plus courante de calcul de l'espace d'états de ce modèle : le graphe des classes d'états. La mise en oeuvre de la méthode des classes de Merlin proposée par Boucheneb [8] est moins complexe que celle proposée dans [22]. En effet, la complexité du calcul d'une classe est réduite de $O(n^3)$ à $O(n^2)$, où n est le nombre de transitions sensiblisées pour le marquage de la classe.

Il est à noter que le graphe des classes d'états donne l'ensemble des marquages du réseau ainsi que les séquences non temporisées de transitions. Le graphe des classes obtenu permet de vérifier des propriétés de type "Linear Temporal Logic" (LTL) [55]. Cependant, nous pouvons remarquer que si la classe C' est le successeur par la transition t de la classe C, cela n'implique pas que tous les états de C ont un successeur par t dans C'. C'est pouquoi le graphe des classes ne permet pas de vérifier des propriétés de type "Computation Tree Logic (CTL)" [55].

Dans le chapitre suivant, nous présentons d'autres modèles temporisés et abordons des travaux qui ont comparé ces modèles temporisés au modèle des réseaux de Petri T-temporel.

Chapitre 4

Les Systèmes Temporisés

Dans le chapitre précédent, nous avons étudié le modèle des réseaux de Petri T-temporels à la Merlin. Dans ce chapitre, nous allons présenter d'autres modèles temporisés tels que les automates temporisés [2] et d'autres variantes des réseaux de Petri temporels. Nous allons faire quelques comparaisons *d'expressivité* entre ces modèles. Les deux principales classes d'expressivité pour les modèles temporisés sont l'expressivité *en terme de bisimulation* et celle *en terme d'acceptation de langage temporisé*.

4.1 Notations Générales

- L'ensemble $\mathbb{B}$ désignent les valeurs booléennes $true$ et $false$,
- $\mathbb{R}_{\geq 0}$ désigne l'ensemble des réels positifs ou nuls,
- $\mathbb{Q}_{\geq 0}$ désigne l'ensemble des rationnels positifs ou nuls,
-B^A désigne l'ensemble des applications de A dans B. Si A est fini et $|A| = n$, un élément de B^A est aussi un vecteur dans B^n. Les opérateurs usuels $+, <, >, \leq, \geq$ et $=$ sont étendus (élément par élément) aux vecteurs de A^n avec $A = \mathbb{N}, \mathbb{Q}, \mathbb{R}$;
- Une valuation νc sur un ensemble de variables X est un élément de $\mathbb{R}_{\geq 0}^X$. Pour tout $\nu c \in \mathbb{R}_{\geq 0}^X$ et $d \in \mathbb{R}_{\geq 0}$, $\nu c + d$ désigne la valuation $\forall x \in X$, $(\nu c + d)(x) = \nu c(x) + d$, et pour $X' \subseteq X$, $\nu c[X' \rightarrowtail 0]$ désigne la valuation, $\nu c'$ avec $\nu c'(x) = 0$ si $x \in X'$ et $\nu c'(x) = \nu c(x)$ sinon. $\overline{0}$ désigne la valuation telle que $\forall x \in X$, $\overline{0}(x) = 0$.
- Soit X un ensemble de variables, une *contrainte atomique* sur X est une formule de la forme $x \bowtie c$ avec $x \in X$, $c \in \mathbb{Q}_{\geq 0}$ et $\bowtie \in \{<, \leq, \geq, >\}$. $C(X)$ désigne l'ensemble des contraintes sur l'ensemble de variables X constitué de la conjonction des contraintes atomiques sur X.

4.2 Langages et Systèmes de Transitions Temporisés

Les modèles temporisés ont une sémantique à base de systèmes de transitions temporisés (Timed Transition System en anglais, TTS). Dans ces systèmes, il existe deux types de transitions possibles : des transitions d'action et des transitions de temps modélisant respectivement des évolutions *discrètes* et des évolutions *continues* du système.

Définition 4.1. (Système de transitions temporisé) Un système de transitions temporisé (TTS) est défini par un tuple $S = (Q, Q_0, \Sigma, \rightarrow)$ où Q est un ensemble d'états, $Q_0 \subseteq Q$ est un ensemble d'états initia ux, Σ est un ensemble fini des actions (disjoint de $\mathbb{R}_{\geq 0}$) et $\rightarrow \subseteq Q \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times Q$ est une relation de transition (ensemble d'arcs). Si $(s, e, s') \in \rightarrow$, nous notons aussi $s \xrightarrow{e} s'$.

Les transitions $\xrightarrow{a}$, $a \in \Sigma$, correspondent à des actions au sens usuel et sont considérées instantanées. Notons que le mot vide ε représente une action *non observable*. Les autres transitions $\xrightarrow{d}$, $d \in \mathbb{R}_{\geq 0}$, expriment l'écoulement d'une durée et doivent de ce fait vérifier des conditions particulières, qui traduisent la compatibilité du système par rapport aux opérations sur le temps :
- 0-DÉLAI : $s \xrightarrow{0} s'$ ssi $s = s'$,
- ADDITIVITÉ : $s \xrightarrow{d} s'$ et $s' \xrightarrow{d'} s"$ avec $d, d' \in \mathbb{R}_{\geq 0}$ alors, $s \xrightarrow{d+d'} s"$,
- CONTINUITÉ : si $s \xrightarrow{d} s'$, alors pour tout d' et $d"$ dans $\mathbb{R}_{\geq 0}$ tel que $d = d' + d"$, il existe $s"$ tel que $s \xrightarrow{d'} s" \xrightarrow{d"} s'$,
- DÉTERMINISME TEMPOREL : si $s \xrightarrow{d} s'$ et $s \xrightarrow{d} s"$, $d \in \mathbb{R}_{\geq 0}$ alors $s' = s"$.

Remarque 4.1. Si $s \xrightarrow{d} s'$, avec $d \in \mathbb{R}_{\geq 0}$, alors d exprime un délai et non un temps absolu.

Définition 4.2. (Exécution d'un TTS) Une exécution ρ d'un TTS S est une séquence finie ou infinie s'écrivant sous la forme d'une alternance de transition continue (éventuellement de durée 0) et de transition discrète :
$\rho = s_0 \xrightarrow{d_0} s'_0 \xrightarrow{a_0} s_1 \xrightarrow{d_1} s'_1 \xrightarrow{a_1} ...s_n \xrightarrow{d_n} s'_n...$
Pour une exécution de taille n, nous notons $Untimed(\rho) = a_0 a_1 ... a_n$ et $Duration(\rho) = \sum_{i=0}^{n} d_i$.

Définition 4.3. (Trace) La trace d'une exécution $\rho = s_0 \xrightarrow{d_0} s'_0 \xrightarrow{a_0} s_1 \xrightarrow{d_1} s'_1 \xrightarrow{a_1} ...s_n \xrightarrow{d_n} s'_n...$
d'un TTS est le mot temporisé $trace(\rho) = (a_0, \delta_0)(a_1, \delta_1)...(a_n, \delta_n)...$ avec $\delta_k = \Sigma_{i=0}^{k} d_i$.
Un *mot temporisé* $\omega = (a_i, d_i)_{0 \leq n}$ est *accepté* par le système de transition STT S si il existe une exécution de S à partir d'un état initial de S et de trace ω. Le langage temporisé $\mathcal{L}(S)$ est l'ensemble des mots temporisés acceptés par S.

Définition 4.4. (TTS ε-abstrait) Soit $S = (Q, Q_0, \Sigma_\varepsilon, \rightarrow)$ un TTS. Nous définissons le TTS S^ε= $S^\varepsilon = (Q, Q_0^\varepsilon, \Sigma, -$) dans lequel les actions ε ont été abstraites de S par :

- $s \xrightarrow{d}_\varepsilon s'$ avec $d \in \mathbb{R}_{\geq 0}$ ssi il existe une trace $\rho = s \rightarrow s'$ dans S avec $Untimed(\rho) = \varepsilon^*$ et $Duration(\rho) = d$.
- $s \xrightarrow{a}_\varepsilon s'$ avec $a \in \Sigma$ ssi il existe une trace $\rho = s \rightarrow s'$ dans S avec $Untimed(\rho) = \varepsilon^* a \varepsilon^*$ et

$Duration(\rho) = 0$,
– $Q_0^\varepsilon = \{s | \exists\, s' \in Q_0 |\, \rho = s' \xrightarrow{*} s \text{ et } Untimed(\rho) = \varepsilon \text{ et } Duration(\rho) = 0\}$

4.3 Bisimulations et Expressivité de Modèles Temporisés

Les deux principales classes d'expressivité pour les modèles temporisés sont l'expressivité en terme de bisimulation et celle en terme d'acceptation de langage temporisé (l'expressivité en terme de bisimulation implique l'expressivité en terme de langage). D'autre part, cette recherche s'exprime souvent en terme de traduction d'un modèle vers un autre permettant de faire hériter au premier les résultats de décidabilité du deuxième en rapport avec la classe d'expressivité considérée. Par exemple si l'on prouve que la classe de modèle A est plus expressive que la classe de modèle B en terme de bisimulation par le biais d'une traduction de B vers A préservant la bisimulation et si un problème tel que la vérification de TCTL est décidable sur la classe A alors, on obtient la décidabilité de ce même problème pour la classe B. De plus, si la comparaison conduit à une égalité d'expressivité par une double traduction alors on peut aussi déduire des résultats de complexité algorithmique [59]. Dans ce qui suit, nous formalisons la notion de bisimulation temporelle et celle de langage temporisé.

4.3.1 Bisimulation Temporelle [59]

Définition 4.5. (Bisimulation temporelle forte) Soient $S_1 = (Q_1, Q_0^1, \Sigma, \rightarrow_1)$ et $S_2 = (Q_2, Q_0^2, \Sigma, \rightarrow_2)$ deux systèmes de transitions temporisés et $\approx_{\mathcal{S}}$ une relation binaire $\subseteq Q_1 \times Q_2$. $\approx_{\mathcal{S}}$ est une *relation de bisimulation temporelle forte* entre S_1 et S_2 si et seulement si $\forall a \in \Sigma \cup \mathbb{R}_{\geq 0}$:

– si $s_1 \in Q_0^1$ alors il existe $s_2 \in Q_0^2$ tel que $s_1 \approx_{\mathcal{S}} s_2$,
– si $s_1 \xrightarrow{a}_1 s_1'$ et $s_1 \approx_{\mathcal{S}} s_2$ alors $\exists\, s_2 \xrightarrow{a}_2 s_2'$ tel que $s_1' \approx_{\mathcal{S}} s_2'$;
– inversement, si $s_2 \xrightarrow{a}_2 s_2'$ et $s_1 \approx_{\mathcal{S}} s_2$ alors $\exists\, s_1 \xrightarrow{a}_1 s_1'$ tel que $s_1' \approx_{\mathcal{S}} s_2'$.

Deux Systèmes de transitions temporisés S_1 et S_2 sont sont en relation de bisimulation temporelle forte si il existe une bisimulation temporelle forte entre S_1 et S_2. Nous notons alors $S_1 \approx_{\mathcal{S}} S_2$.

Définition 4.6. (Bisimulation Temporelle Faible) Soient $S_1 = (Q_1, Q_0^1, \Sigma \cup \{\varepsilon\}, \rightarrow_1)$ et $S_1 = (Q_2, Q_0^2, \Sigma \cup \{\varepsilon\}, \rightarrow_2)$ deux systèmes de transitions étiquetés et une relation binaire $\approx_{\mathcal{W}} \subseteq Q_1 \times Q_2$. $\approx_{\mathcal{W}}$ est une relation de *bisimulation temporelle faible* entre S_1 et S_2 si elle est une relation de bisimulation forte entre S_1^ε et S_2^ε.

Remarques :
- si $S_1 \approx_{\mathcal{S}} S_2$ alors $S_1 \approx_{\mathcal{W}} S_2$;
- si $S_1 \approx_{\mathcal{W}} S_2$ alors $\mathcal{L}(S_1) = \mathcal{L}(S_2)$;

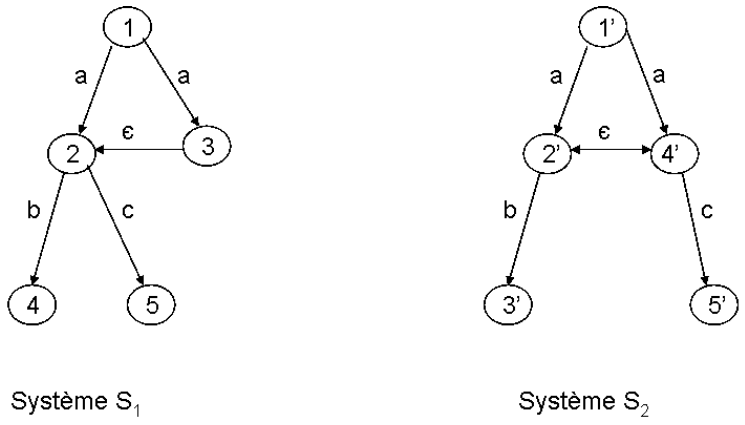

FIG. 4.1 – Deux systèmes à comparer

Dans ce qui suit, nous considérons que la bisimulation faible et on notera $\approx$ pour $\approx_{\mathcal{W}}$.

4.3.2 Expressivité des Modèles Temporisés

Soient C et C' deux classes de modèles temporisés.

Définition 4.7. (Expressivité en termes de langage temporisé) La classe C est plus expressive qu'une classe C' en termes de langage temporisé si pour tout $B' \in C'$ il existe $B \in C$ tel que $\mathcal{L}(B) = \mathcal{L}(B')$. Nous notons alors $C' \leq_{\mathcal{L}} C$. De plus, si il existe $B \in C$ tel qu'il n'existe pas $B' \in C'$, avec $\mathcal{L}(B) = \mathcal{L}(B')$, alors C est *strictement plus expressive* que C'. Nous notons alors $C' <_{\mathcal{L}} C$. Si $C \leq_{\mathcal{L}} C'$ et $C' \leq_{\mathcal{L}} C$ alors C et C' sont *expressivement équivalentes* en termes d'acceptation de langage temporisé, et nous notons $C =_{\mathcal{L}} C'$.

Définition 4.8. (Expressivité en termes de bisimulation temporelle) La classe C est plus expressive qu'une classe C' en termes de bisimulation temporelle si pour tout $B' \in C'$ il existe $B \in C$ tel que $B \approx B'$. Nous notons alors $C' \subseteq_{\approx} C$. De plus, si il existe $B \in C$ tel qu'il n'existe pas $B' \in C'$, avec $B \approx B'$, alors C est *strictement plus expressive* que C'. Nous notons alors $C' \subset_{\approx} C$. Si $C \subseteq_{\approx} C'$ et $C' \subseteq_{\approx} C$ alors C et C' sont *expressivement équivalentes* en termes de bisimulation temporelle et nous notons $C =_{\approx} C'$.

Exemple 4.1. Considérons les deux systèmes S_1 et S_2 de la figure 4.1. Il est clair que le langage non temporisé de S_1 et celui de S_2 sont identiques ($\mathcal{L}(S_1) = \mathcal{L}(S_2) = \{ab, ac\}$). En revanche, ces deux systèmes ne sont pas fortement bisimulaires ; en effet, l'état 2 de S_1 n'est bisimulaire à aucun état de S_2 (de même pour l'état 3). Par contre, les deux systèmes sont faiblement bisimulaires.

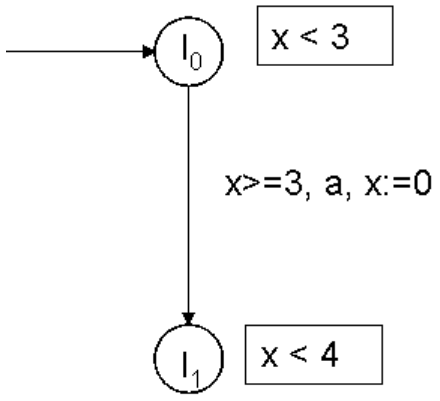

FIG. 4.2 – Un automate temporisé simple

4.4 Automates Temporisés

Les automates automatisés, ou timed automata en anglais, ont été introduits par Alur et Dill en 1994 [3]. Henzinger et al ont étendu ce modèle à l'aide de contraintes sur les horloges, appelées invariants associés aux localités [34]. Le temps est ajouté au modèle classique des automates sous la forme d'horloges et de prédicats sur ces horloges. Ces prédicats sont de deux types : les *gardes*, associées aux transitions discrètes, donnent des contraintes sur les horloges à respecter pour pouvoir effectuer ces transitions discrètes. Les *invariants*, associés aux localités, donnent des contraintes sur les horloges qui doivent être respectées dans les localités.

Définition 4.9. (Automate temporisé) Un automate temporisé est un 6-uplet $\langle L, l_0, X, \Sigma, E, F, Inv \rangle$ où :
- L est un ensemble fini de localités,
- l_0 est la localité initial,
- X est un ensemble fini d'horloges à valeurs réelles positives,
- Σ est un ensemble fini d'actions,
- $E \subseteq L \times C(X) \times \Sigma \times 2^X \times L$ est un ensemble fini d'arcs. Soit $e = (l, \gamma, a, X', l') \in E$. e est un arc reliant la localité l à la localité l', avec la garde γ, l'action a et l'ensemble des horloges à remettre à zéro X',
- $Inv \in C(X)$ associe un invariant à chaque localité.

La figure 4.2 donne un exemple très simple d'automate temporisé. La localité initiale est l_0. L'automate dispose d'une seule horloge : x. x est nul à l'instant initial et l'invariant de l_0 indique donc que l'on pourra rester dans l_0 strictement moins de 3 unités de temps. Dès que 3 unités de temps se sont écoulées, la garde de la transition entre l_0 et l_1 est vérifiée (et l'invariant de l_1 est vérifié après franchissement de cette transition) et la transition peut donc être franchie. Si cette transition est franchie, l'horloge x est remise à zéro et l_1

devient la localité active.

Définition 4.10. (Sémantique d'un automate temporisé) La sémantique d'un automate temporisé A= $\langle L, l_0, X, \Sigma, E, I$ est un système de transition temporisé $S_A = (Q, q_0, \rightarrow)$ où
- $Q = L \times (\mathbb{R}_{\geq 0})^X$,
$- \ q_0 = (l_0, \overline{0})$ est la configuration initiale
- $\rightarrow \in Q \times (\Sigma \cup \mathbb{R}) \times Q$ est définie par :
i) la relation de transition discrete
$(l, \nu c) \rightarrow^a (l', \nu c')$ ssi $\exists (l, \gamma, a, X', l') \in E$ tel que

$$\begin{cases} \gamma(\nu c) = true, \\ \nu c' = \nu c[X' \rightarrowtail 0], \\ Inv(l')(\nu c') = true; \end{cases}$$

ii) la relation de transition continue $(l, \nu c) \rightarrow^d (l, \nu c')$ ssi

$$\begin{cases} \nu c' = \nu c + d, \\ \forall 0 \leq d' \leq d, Inv(l)(\nu c + d') = true; \end{cases}$$

En général, on décrit un système comme une composition d'automates temporisés. Dans ce but, on utilise la notion classique de composition basée sur une fonction de synchronisation à la Arnold-Nivat [3]. Par ailleurs la composition parallèle de n automates temporisés est basée sur un ensemble commun d'actions possibles Σ et une fonction de synchronisation définie de $\Sigma_\bullet^n$ dans Σ, où $\Sigma_\bullet = \Sigma \cup \{\bullet\}$ et $\bullet$ est un symbole spécial utilisé quand un des automates n'est pas concerné par un pas du système globale. Soient $A_1, ..., A_n$ n automates avec $A_i = \langle L_i, l_{i,0}, X_i, \Sigma, E_i, Inv_i, \rangle$ tels que les différents X_i (avec, $i = 1..n$) soient disjoints. Nous notons $A = (A_1|...|A_n)_f$ la composition parallèle des A_i synchronisés par f. Les états de $(A_1|...|A_n)_f$ sont appelés configurations et sont des paires $(\bar{l}, \nu c)$ où $\bar{l} = (l_1, ..., l_n)$ et $\nu c = (\nu c_1, ..., \nu c_n)$ avec $\nu c_i \in (\mathbb{R}_{\geq 0})^{X_i}$. La sémantique d'un produit synchronisé d'automates temporisés est un système de transitions temporisé. Cela est formalisé comme suit :

Définition 4.11. (Sémantique d'un produit d'automates temporisés) Soient $A_1, ..., A_n$ n automates avec $A_i = \langle L_i, l_{i,0}, X_i, \Sigma, E_i, Inv_i, \rangle$ et f une fonction de synchronisation de $\Sigma_\bullet^n \rightarrow \Sigma$, la sémantique de $(A_1|...|A_n)_f$ est un système de transitions temporisé $S = (Q, q_0, \rightarrow)$ défini par :
$Q = L_1 \times ... \times L_n \times (\mathbb{R}_{\geq 0})^X$ (avec $X = \cup_{i=1..n} X_i$) est l'ensemble des configurations, $q_0 = (\overline{l_0}, \overline{0})$ est la configuration initiale et la relation de transition $\rightarrow$ définie par :
1. $(\bar{l}, \nu) \rightarrow^b (\overline{l'}, \nu')$ ssi il existe $(a_1, ..., a_n) \in A_\bullet^n$ tel que $f(a_1, ..., a_n) = b$ et pour tout $i \in 1..n$ nous avons :
- si $a_i = \bullet$, alors $l'_i = l_i$ et $\nu'_i = \nu_i$,
- si $a_i \in A$, alors $(l_i, \nu_i) \rightarrow^{a_i} (l'_i, \nu'_i)$.
2. $(\bar{l}, \nu) \rightarrow^d (\overline{l'}, \nu')$ ssi $\forall i \in [1..n]$, nous avons $(l_i, \nu_i) \rightarrow^d (l'_i, \nu'_i)$.

4.5 Réseaux de Petri P-temporels

4.5.1 Le Modèle

Les P-temporels, introduits dans [39], associent des intervalles temporels aux places des réseaux ; ces intervalles représentent des durées de séjour admissibles des jetons dans les places. Par ailleurs, la violation de cette spécification se traduira par la mort des jetons. Une transition est franchissable si et seulement si tous les jetons utilisés pour le tir de la transition respectent leurs intervalles résiduels. Soient p_i une place caractérisée par l'intervalle temporel $[a_i, b_i]$ et un jeton de cette place. Ce jeton participe à la validation d'une de ses transitions de sortie que s'il a séjourné au moins la durée a_i dans cette place. De plus, dans la sémantique forte, le jeton doit quitter la place p_i quand d'une part, sa durée de séjour devient égale à b_i et d'autre part, il existe au moins une transition franchissable qui peut le consommer. Après le temps b_i, si le jeton n'a pas quitté la place, il devient mort et ne participera plus au franchissement des transitions. Quand un jeton est généré dans une place, son âge est nul.

Définition 4.12. (Réseau de Petri P-temporel) Un réseau de Petri P-temporel est un tuple (R, Is) où R est un réseu de Petri classique et Is est la fonction d'intervalle statique, qui à chaque place p associe son intervalle de contraintes temporelles $[\alpha(p), \beta(p)]$.
Dans ce qui suit, nous allons nous intéresser à des P-temporels *saufs* (1-bornés).

4.5.2 Sémantique d'un P-temporel

L'état d'un P-temporel est un tuple $(M, dead, \nu c)$ où M est un marquage ordinaire, $dead \in \{0,1\}^P$ est l'application de jetons morts et $\nu c \in \mathbb{R}^P_{\geq 0}$ est l'application "âge" des jetons des places.
Soit $dead : P \rightarrow \{0,1\}^P$ une application qui associe à chaque place le nombre de jetons morts dans la place p $(\forall p \in P : dead(p) \leq M(p))$. Notons que $p \in M - dead$ pour $M(p) - dead(p) \geq 1$.
Pour les P-temporels, les notions de *enabled* et *firable* sont définies comme suit :
$t \in enabled(M - dead)$ ssi $M - dead \geq {}^\bullet t$
$t \in firable(M, dead, \nu c)$ ssi $t \in enabled(M - dead) \wedge \forall p \in {}^\bullet t : \nu c(p) \in [\alpha(p), \beta(p)]$

Définition 4.13. (Sémantique forte d'un réseau de Petri P-temporel) La sémantique d'un P-temporel R est donnée par un système de transition temporisé $S_R = (Q, q_0, \rightarrow)$ où : $Q = \{0,1\}^P \times \{0,1\}^P \times (\mathbb{R}_{\geq 0})^P$, $q_0 = (M_0, \overline{0}, \overline{0})$, $\rightarrow \in Q \times ((\Sigma \cup \{\varepsilon\}) \cup \mathbb{R}_{\geq 0}) \times Q$ est une relation de transition qui se décompose en une relation de transition discrète et une relation de transition continue :

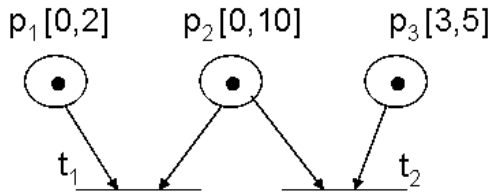

FIG. 4.3 – Un réseau P-temporel simple

1. La relation de transition discrète est définie $\forall t \in T$:

$$(M, dead, \nu c) \xrightarrow{t} (M', dead', \nu c') \iff \begin{cases} t \in firable(M, dead, \nu c) \\ M' = M - {}^\bullet t + t^\bullet \\ \nu c'(p) = \begin{cases} 0 & si\ (dead(p) = 0) \\ & \wedge (p \in t^\bullet) \\ \nu c(p) & sinon \end{cases} \end{cases}$$

2. La relation de transition discrète est définie $\forall d \in \mathbb{R}_{\geq 0}$:

$$(M, dead, \nu c) \xrightarrow{d} (M, dead', \nu c') \iff$$

$$\begin{cases} \nu c' = \nu c + d. \\ \forall t \in T : \\ t \notin firable(M, dead, \nu c + d) \Rightarrow (\forall d' \in [0, d] : t \notin firable(M, dead, \nu c + d') \\ dead'(p) = \begin{cases} 1 & si\ (p \in (M - dead)) \wedge (\nu c'(p) > \beta(p)). \\ dead(p) & sinon. \end{cases} \end{cases}$$

Définition 4.14. (Sémantique faible d'un réseau de Petri P-temporel) La sémantique faible d'un P-temporel est similaire à la forte sauf que la relation de transition continue ne comporte pas la relation suivante :

$$\forall t \in T : \\ t \notin firable(M, dead, \nu c + d) \Rightarrow (\forall d' \in [0, d] : t \notin firable(M, dead, \nu c + d').$$

Exemple 4.2. Considérons le P-temporel de la figure 4.3. En considérant la sémantique forte, la transition t_1 doit être franchie avant t_2 et par conséquent, t_2 ne le sera jamais. En effet, le jeton de la place p_1 atteint sa borne temporelle supérieure avant que t_2 ne soit franchissable (à cet instant, le jeton dans p_3 n'est pas encore visible). t_1 étant la seule transition de sortie associée à p_1, elle doit être tirée. Dans la sémantique faible, nous pouvons tirer soit t_1 soit t_2.

4.6 Réseaux de Petri A-temporels

4.6.1 Le modèle

Ce sont des réseaux temporels, comme ceux de Merlin, car ils considèrent un délai et non une durée. Ils associent des intervalles temporels aux arcs [33] [1] [25]. Comme pour les P-temporels, un âge est

associé à chaque jeton. Une transition t peut être tirée que s'il existe, pour chaque place en entrée de la transition, un jeton dont l'age satisafait la contrainte sur l'arc liant la place à la transition. Aussi, comme pour les P-temporels, il y a création de jetons morts (i.e. ceux dont l'âge dépasse les bornes temporelles supérieures de tous les arcs sortants de la place contenant le jeton). De plus, chaque jeton est généré avec un âge nul. Par ailleurs, dans la sémantique forte quand un jeton atteint sa borne temporelle supérieure et il existe une transition pouvant le consommer, la transition doit être tirée.

Définition 4.15. (Réseau de Petri A-temporel) Un réseau de Petri A-temporel est un tuple (R, Is) où R est un réseau de Petri classique et Is est la fonction d'intervalle statique, qui associe, à chaque arc e, un intervalle $[\alpha(e), \beta(e)]$ de contraintes temporelles.
Dans ce qui suit, nous allons nous intéresser à des A-temporels saufs (1-bornés).
Pour les A-temporels, les notions de *enabled* et *firable* sont définies comme suit :
$t \in enabled(M - dead)$ ssi $M - dead \geq^{\bullet} t$
$t \in firable(M, dead, \nu c)$ ssi $t \in enabled(M - dead) \wedge \forall p \in^{\bullet} t : \nu c(p) \in [\alpha(p,t), \beta(p,t)]$.

4.6.2 Sémantique d'un A-temporel

Définition 4.16. (Sémantique forte d'un réseau de Petri A-temporel) La sémantique d'un A-temporel R est donnée par un système de transition temporisé $S_R = (Q, q_0, \rightarrow)$ où : $Q = \{0,1\}^P \times \{0,1\}^P \times (\mathbb{R}_{\geq 0})^P$, $q_0 = (M_0, \overline{0}, \overline{0})$, $\rightarrow \in Q \times ((\Sigma \cup \{\varepsilon\}) \cup \mathbb{R}_{\geq 0}) \times Q$ est une relation de transition qui se décompose en une relation de transition discrète et une relation de transition continue :

1. La relation de transition discrète est définie $\forall t \in T$:

$$(M, dead, \nu c) \xrightarrow{t} (M', dead', \nu c') \iff \begin{cases} t \in firable(M, dead, \nu c) \\ M' = M - {}^{\bullet}t + t^{\bullet} \\ \nu c'(p) = \begin{cases} 0 & si\ (dead(p) = 0) \\ & \wedge (p \in t^{\bullet}) \\ \nu c(p) & sinon \end{cases} \end{cases}$$

2. La relation de transition discrète est définie $\forall d \in \mathbb{R}_{\geq 0}$:

$$(M, dead, \nu c) \xrightarrow{d} (M, dead', \nu c') \iff$$
$$\begin{cases} \nu c' = \nu c + d \\ \forall t \in T : \\ t \notin firable(M, dead, \nu c + d) \Rightarrow (\forall d' \in [0, d] : t \notin firable(M, dead, \nu c + d')) \\ dead'(p) = \begin{cases} 1 & si\ (p \in (M - dead)) \wedge (\forall t \in p^{\bullet}, \nu c'(p) > \beta(p,t)) \\ dead(p) & sinon \end{cases} \end{cases}$$

Exemple 4.3. Considérons le réseau A.temporel de la figure 4.4. En considérant la sémantique forte, la transition t_1 doit être franchie avant t_2 et par conséquent, t_2 ne le sera jamais. En effet, le jeton de la place p_1 atteint la borne temporelle supérieure de son arc de sortie (p_1, t_1)avant que t_2 ne soit franchissable (à cet

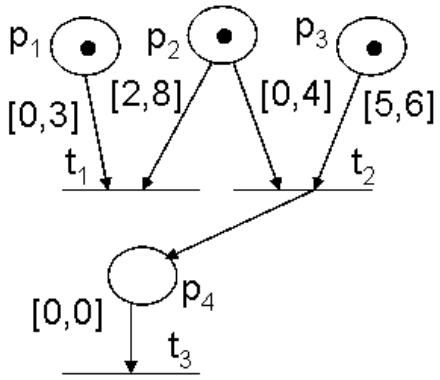

FIG. 4.4 – Un réseau A-temporel simple

instant, le jeton de p_3 ne respecte pas encore la contrainte de l'arc (p_3, t_2). Il s'ensuit que t_2 et t_3 ne seront jamais tirées. Dans la sémantique faible, nous pouvons tirer soit t_1 soit t_2 suivi du tir de t_3.

Notations. Nous notons $\overline{X.RdPT}$ (resp. X.RdPT) les X-temporels (avec $X \in \{A, P, T\}$) avec la sémantique forte (resp. avec la sémantique faible).

4.7 Réseaux de Petri T-temporels et Automates Temporisés

Parmi les travaux les plus recents comparant l'expressivité des $\overline{T.RdPT}$ et des automates temporisés (AT), nous pouvons citer les travaux de Olvier H. R oux, Franck Cassez, Béatrice Bérard, Serge Haddad et Didier Lime [13][11][12][15] [14]. Ces travaux ont permis de prouver que les $\overline{T.RdPT}$ et les AT ont la *même expressivité* en termes d'acceptation de *langage temporisé* [59]. D'autre part, ils ont aussi prouvé que les ATs sont *strictement plus expressifs* que les $\overline{T.RdPT}$ bornés en termes de bisimulation temporelle. Pour cela il a été prouvé que tout $\overline{T.RdPT}$ peut être traduit en un AT avec *variables* qui lui est temporellement bisimilaire. Nous ne pouvons pas donner tous les résultats de ces travaux car cela dépasse le cadre de cette thèse. Nous allons montrer comment traduire un $\overline{T.RdPT}$ en un AT et conclure sur la comparaison de l'expressivité des $\overline{T.RdPT}$ et AT en termes de bisimulation.

4.7.1 Technique de Traduction

Cette technique consiste à traduire un $\overline{T.RdPT}$ en un produit synchronisé d'automates temporisés. Pour cela, un AT est associé à chaque transition du $\overline{T.RdPT}$. Chaque automate temporisé possède une horloge représentant le temps de sensibilisation de la transition correspondante du $\overline{T.RdPT}$. Les états de l'automate donnent l'état de cette transition : *sensibilisée*, *non-sensibilisée* ou *en train d'être tirée*. Il a été prouvé que la traduction proposée préserve le comportement du $\overline{T.RdPT}$ initial dans le sens où la sémantique du $\overline{T.RdPT}$ et sa traduction sont temporellement bisimilaires. Nous donnons la procédure de

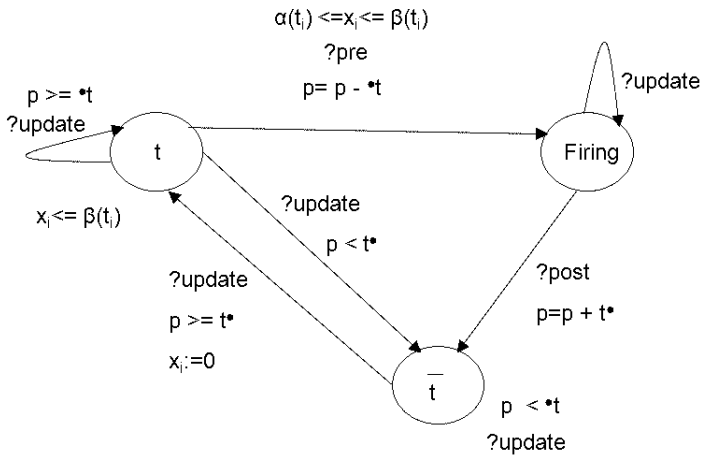

FIG. 4.5 – Un automate temporisé A_t associé à la transition t

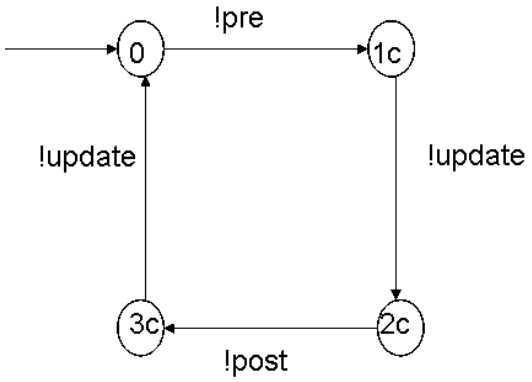

FIG. 4.6 – L' automate temporisé SU associé au superviseur

traduction comme elle a été définie dans [15][14] :

Automate temporisé associé à une transition du $T.RdPT$

Soit R un $\overline{T.RdPT}$ à traduire en un AT. Nous définissons un automate temporisé A_i pour chaque transition t_i du $\overline{T.RdPT}$ (voir figure 4.5). Cet automate temporisé possède une horloge x_i. Les états de l'automate A_i donnent l'état de la transition t_i : dans l'état t, la transition est sensibilisée ; dans l'état $\overline{t}$, elle n'est pas sensibilisée ; dans l'état F, elle est en train d'être tirée. L'état initial de chaque A_i dépend du marquage initial M_0 du $\overline{R.RdPT}$ R. Si $M_0 \geq {}^\bullet t_i$ alors l'état initial est t et, dans le cas contraire, c'est $\overline{t}$. Cet automate met à jour un tableau d'entiers p qui est le vecteur de marquage et qui est partagé par tous les automates A_i.

Le superviseur

Le superviseur SU est décrit dans la figure 4.6. Les localités 1 à 3 indexées par un c sont supposées urgentes ; ce qui signifie que le temps ne peut pas s'écouler lorsque l'on est dans ces localités. L'état initial du superviseur est 0. Nous définissons la fonction de synchronisation f à $n+1$ paramètres (n est le nombre de transitions du $\overline{T.RdPT}$ à traduire) par :

1. $f(!pre, \bullet, ..., ?pre, \bullet, ...) = pre_i$ si $?pre$ est le $(i+1)$ème argument et tous les autres arguments sont $\bullet$,
2. $f(!post, \bullet, ..., ?post, \bullet, ...) = post_i$ si $?post$ est le $(i+1)$ème argument et tous les autres arguments sont $\bullet$,
3. $f(!update, ?update, ..., ?update) = update$.

Nous notons $\Delta(R) = (SU \times A_1 \times ... \times A_n)_f$ l'automate temporisé associé au $\overline{T.RdPT}$ R. Il a été prouvé que la sémantique de $\Delta(R)$ est *équivalente* à celle de R [15][14].

4.7.2 Comparaison entre un Réseau de Petri T-Temporel et un Automate Temporisé

Nous pouvons remarquer que les variables utilisées dans les automates codent le marquage des places du réseau. Etant donné qu'une variable bornée peut être encodée par un automate classique, on peut conclure qu'un $T.RdPT$ borné peut être traduit en un *produit synchronisé d'automates temporisés classiques*. Cela a permis d'aboutir au théorème 4.1 (sa preuve est donnée dans [15][14]) :

Théorème 4.1. Pour tout $\overline{T.RdTP}$ *borné* R, il existe un automate temporisé A tel que $A \approx R$.

Grâce à ce théorème, nous pouvons affirmer que les $\overline{T.RdTP}$ bornés héritent des résultats de décidabilité des *automates temporisés classiques* tels que le *problème du vide* (notons que le test du langage vide permet de décider de l'accessibilité d'un état) d'un $\overline{T.RdPT}$) et ***TCTL***.

Théorème 4.2. Il n'existe pas de $\overline{T.RdPT}$ temporellement bisimilaire (même faiblement) à A_0 de la figure 4.7a (voir [13]).

Ceci est dû au fait que l'écoulement de temps ne peut pas désensibiliser une action (transition) dans la sémantique forte d'un réseau $T.RdPT$, alors que cela est possible pour un automate temporisé. Donc si il existait un $\overline{T.RdPT}$ bisimilaire à A_0, cela impliquerait que l'on peut séjourner indéfiniment dans l'état initial $\overline{T.RdPT}$ et de ce fait désensibiliser la transition a. Ceci n'est pas possible à cause de la sémantique forte des $\overline{T.RdPT}$. Grâce à ces deux théorèmes, il a été prouvé que la classe des $\overline{T.RdPT}$ bornés est d'*expressivité strictement inférieure* à celle de la classe des AT.

4.8 Expressivité des Différentes Variantes des Réseaux Temporels

Très peu d'études [40][10][16] comparent l'expressivité des differentes classes des réseaux de Petri temporels ($\overline{T.RdPT}$, $\underline{T.RdPT}$, $\overline{A.RdPT}$, $\underline{A.RdPT}$, $\overline{P.RdPT}$ et $\underline{P.RdPT}$) en considérant les deux types de sémantiques possibles (faible et forte). Les travaux les plus récents faisant de telles comparaisons sont ceux de M. Boyer et O. H. Roux [9]. Nous allons donner une synthèse des résultats de ces travaux.

Théorème 4.3. Le RdPT $R_{T0} \in \underline{T.RdPT}$ (voir figure 4.7b) est temporellement bisimilaire (faiblement) à $A_0 \in \mathit{AT}$.

Soit $(l, \nu c)$ un état de $A_0 \in \mathit{AT}$ où $l \in (l_0, l_1)$ et $\nu c(x) \in \mathbb{R}_{\geq 0}$ est la valuation de l'horloge x. Nous définissons une relation $\approx \subseteq (\{l_0, l_1\} \times \mathbb{R}_{\geq 0}) \times (\{0, 1\} \times \mathbb{R}_{\geq 0})$ par :

$$(l, \nu c) \approx (M, \upsilon c) \iff \begin{cases} (1)\ l = l_0 \Leftrightarrow M(p_1) = 1 \\ \quad\ l = l_1 \Leftrightarrow M(p_1) = 0 \\ (2)\ \nu c(x) = \upsilon c(x) \end{cases}$$

Il est clair que $\approx$ est une relation de bisimulation temporelle faible.
De la même manière, le théorème 4.4 a été prouvé.

Théorème 4.4. Le RdPT $R_{P0} \in \underline{P.RdPT}$ (figure 4.7c) est temporellement bisimilaire (faiblement) à $A_0 \in \mathit{AT}$.

Théorème 4.5. (La sémantique faible ne peut pas simuler la sémantique forte) $\overline{T.RdPT} \not\subseteq_{\approx} \underline{T.RdPT}$ $\overline{P.RdPT} \not\subseteq_{\approx} \underline{P.RdPT}$ $\overline{A.RdPT} \not\subseteq_{\approx} \underline{A.RdPT}$

Supposons qu'il existe un $R_{T2} \in \underline{T.RdPT}$ temporellement bisimilaire (faiblement) à $R_{T1} \in \overline{T.RdPT}$ (figure 4.7d). A partir de l'état initial de R_{T2}, il est possible d'y séjourner pour une durée $d > 1$ et donc de désensibiliser la transition a (dans la sémantique faible, un délai est toujours possible). Par l'hypothèse de bisimulation, ceci serait aussi possible à partir de l'état initial de R_{T1} (par conséquent ceci impliquerait la désensibilisation de la transition a dans R_{T1}). Or ceci n'est pas possible car l'écoulement de temps ne peut pas désensibiliser les transitions en considérant la sématique forte. Ceci contredit notre hypothèse. Il s'ensuit que $\overline{T.RdPT} \not\subseteq_{\approx} \underline{T.RdPT}$. Nous pouvons tenir le même raisonnement pour les réseaux P-temporels et A-temporels.

Théorème 4.6. $\underline{T.RdPT} \not\subseteq_{\approx} \overline{T.RdPT}$.

Il a été prouvé qu'il existe $R_{T0} \in \underline{T.RdPT}$ (Figure 4.7b) temporellement bisimilaire (faiblement) à $A_0 \in TA$ (voir théorème 4.3) et il n'existe de $RdPT \in \overline{T.RdPT}$ temporellement bisimilaire (faiblement) à A_0 (voir théorème 4.2). Donc il n'existe pas de $RdPT \in \overline{T.RdPT}$ temporellement bisimilaire (faiblement) à R_{T0}. Ceci démontre le théorème 4.6.

Théorème 4.7. $\overline{P.RdPT} \not\subseteq_{\approx} \overline{T.RdPT}$.

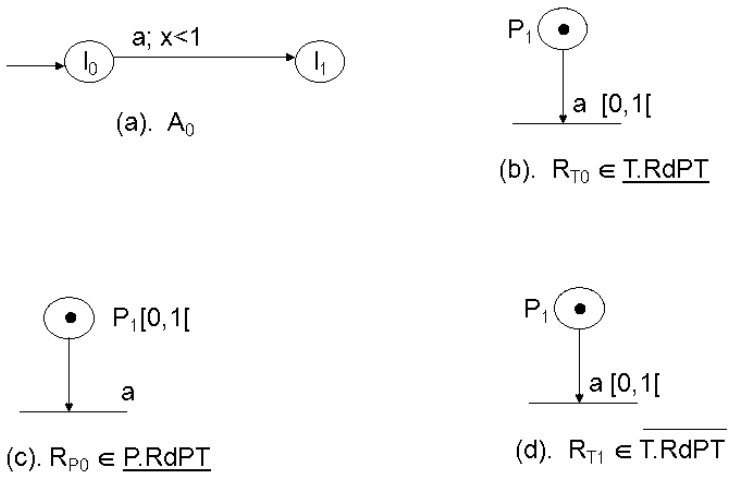

FIG. 4.7 – Des modèles temporisés

Il a été prouvé que $R_{P1} \in \overline{P.RdPT}$ est temporellement bisimilaire (faiblement) à $R_{P0} \in \underline{P.RdPT}$. Ce dernier est temporellement bisimilaire (faiblement) à A_0 (théorème 4.4), il s'ensuit que R_{P1} est aussi temporellement bisimilaire (faiblement) à A_0. Comme il n'existe pas de $\overline{T.RdPT}$ temporellement bisimilaire (faiblement) à A_0 (théorème 4.2), alors il n'existe pas de $\overline{T.RdPT}$ temporellement bisimilaire (faiblement) à $\overline{P.RdPT}$.

Théorème 4.8. $\overline{T.RdPT} \subset_{\approx} \overline{A.RdPT}$.
La preuve consiste à traduire un $\overline{T.RdPT}$ en un $\overline{A.RdPT}$.

4.9 Conclusion

Dans ce chapitre, nous avons, dans un premier temps, étudié l'expressivité des réseaux de Petri T-temporels et des automates temporisés. Nous avons souligné que les automates temporisés sont strictement plus expressifs que les réseaux de Petri T-temporels bornés en termes de bisimulation temporelle. Ceci permet de faire hériter aux réseaux T-temporels bornés les résultats de décidabilité et les méthodes de vérification et de contrôle des automates temporisés. Dans un deuxième temps, nous nous sommes interessés aux travaux comparant les différentes variantes des réseaux de Petri temprels, c'est-à-dire les T-temporels, P-temporels et A-temporels et nous avons présenté des relations d'inclusion strictes ou larges entre ces différentes variantes. D'après cette étude, les T-temporels ne sont pas les plus expressifs. Toutefois, contrairement aux autres modèles temporels, ils ont des outils d'analyse performants et fiables et ont fait l'objet de nombreuses modélisations de systèmes embarqués. Ils présentent donc un bon compromis entre outils d'analyse et puissance d'expression.

Chapitre 5

Extension Temporelle des Réseaux de Petri Récursifs

5.1 Introduction

Dans ce chapitre, nous exposons la démarche que nous avons adoptée pour étendre les RdPR avec le temps. La conception d'un tel modèle nous a poussés à nous poser un certain nombre de questions. D'abord, comment ajouter le temps ? Faut-il supposer une seule référence de temps globale à toutes les opérations possédant des contraintes de temps et ceci pour l'ensemble des processus du réseau ? Ou bien au contraire supposer que chaque composant du système a sa propore propre référence de temps locale ? Pour cette première question, notre choix s'est porté sur une seule référence de temps afin de rester fidèle à la sémantique du modèle temporisé choisi, i.e. celui des réseaux T-temporels. D'autre part, il s'agit de déterminer si le temps, à prendre en compte dans la spécification, doit être discret (correspondant à des instants réguliers) ou continu (pris dans l'ensemble des réels positifs). Comme pour les réseaux T-temporels, le temps pris en compte, dans notre cas, est continu. Une autre question concerne la manière d'attacher les intervalles de tir aux éléments d'un réseau de Petri récursif. Notons que, dans un tel réseau, un tir correspond soit à une transition élémentaire ou abstraite, soit à une coupure. Ainsi, pour obtenir un modèle uniforme, nous proposons d'attacher des intervalles temporels non seulement aux transitions (élémentaires ou abstraites) mais aussi aux coupures. Finalement, comment interpréter ces intervalles de tir ? Pour une transition élémentaire, l'intervalle temporel de tir a la même sémantique que pour une transition d'un réseau T-temporel. Qu'en est-t-il pour une transition abstraite ? Nous avons deux possibilités : soit cet intervalle conditionne les instants de tir et de fermeture d'une transition abstraite, soit il conditionne uniquement l'instant de tir de la transition. Nous avons opté pour la deuxième solution afin de rester fidèle à la sémantique d'un réseau T-temporel. Par la suite, nous montrerons qu'il est possible, avec cette solution, de contraindre l'instant de fermeture d'une transition abstraite. Ainsi, l'intervalle temporel $[a_t, b_t]$ associé à une transition ou une coupure t représente les instants de tirs, ces instants sont relatifs à la date de la dernière sensibilisation de la transition ou coupure t. Si cette référence a lieu à l'instant θ, alors t ne peut pas être tirée avant $\theta + a_t$ et doit l'être avant ou au plus à l'instant $\theta + b_t$, à moins que t a été désensibilisée avant son tir par le tir d'une autre transition ou une coupure.

5.2 Le Modèle des Réseaux de Petri Récursifs Temporels

Un Réseau de Petri Récursif Temporel (RdPRT) est une extension des réseaux de Petri récursifs avec le temps [19] [20]. D'une manière générale, les RdPRT sont basés sur la sémantique des RdPR et celle des réseaux de Petri T-temporels (T.RdPT).

Définition 5.1. (Définition syntaxique d'un RdPRT) Un Réseau de Petri Récursif Temporel est un tuple $\langle R, Is \rangle$ où :

1. R est un RdPR,
2. Is, une *fonction d'intervalle statique*, définie de $T \cup \{\tau_i \mid i \in I_C\} \longrightarrow \mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \{\infty\})$ (où $\mathbb{Q}^+$ est l'ensemble des nombre positifs rationnels). La fonction Is associe à chaque transition ou chaque coupure t un intervalle temporel statique. La plus petite borne de cet intervalle est notée sEFT(t) et est appelée date de tir au plus tôt de t. La plus grande borne est notée par sLFT(t) et est appelée date de tir au plus tard de la transition ou coupure t.

Un marquage étendu d'un RdPRT est défini comme pour un RdPR. Les conditions de tirs des transitions et des coupures restent valables mais sont enrichies par des contraintes temporelles ; ceci est fait non seulement pour les transitions mais aussi pour les coupures. Par conséquent, une transition ou une coupure sensibilisée dans un noeud d'un marquage étendu n'est pas forcément franchissable dans le RdPRT. Quand les conditions de sensibilisation et temporelles sont satisfaites pour une transition ou une coupure, son tir conduit à un marquage étendu construit comme pour les RdPR.

Exemple 5.1 : : Le réseau de la figure 5.1 illustre les principales caractéristiques d'un RdPRT. Les informations temporelles associées à un RdPRT sont représentées de la manière suivante : le nom d'une transition (élémentaire ou abstraite) est suivi par son intervalle de tir statique. Par ailleurs, la notation $\Upsilon_i : Is(\tau_i)$ est utilisée pour représenter l'intervalle de tir statique de la coupure τ_i. Par souci de clarté de la représentation graphique, l'intervalle de tir statique $[0, \infty[$ est omis.

Le réseau de la figure 5.1 montre la modélisation d'une transaction effectuée par un serveur distant. Le statut du serveur est décrit par les places ON et OFF. Une marque dans la place ON indique qu'il est opérationnel. Dans ce cas, le serveur traite deux types de requêtes (voir les transitions t_1 and t_3) permettant d'exécuter une transaction (voir les transitions abstraites t_2 and t_4 avec leurs maquages de départ respectifs, incluant chacun un jeton dans la place P_{init}). Il est à remarquer que les deux transactions correspondent au *même composant* (voir les places P_{init} et P_{end}), sauf qu'elles s'exécutent avec des *conditions temporelles différentes*. Ainsi une transaction associée à une requête de type 1 (la transition t_1) doit finir au plus tard 4 unité de temps depuis son début (voir la transition t_7) alors que la requête de type 2 (la transition t_3) termine au plus tard 7 unités de temps depuis son début (voir la transition t_8).

Une transaction en exécution peut soit terminer normalement (représenté par un jeton dans la place P_{end}

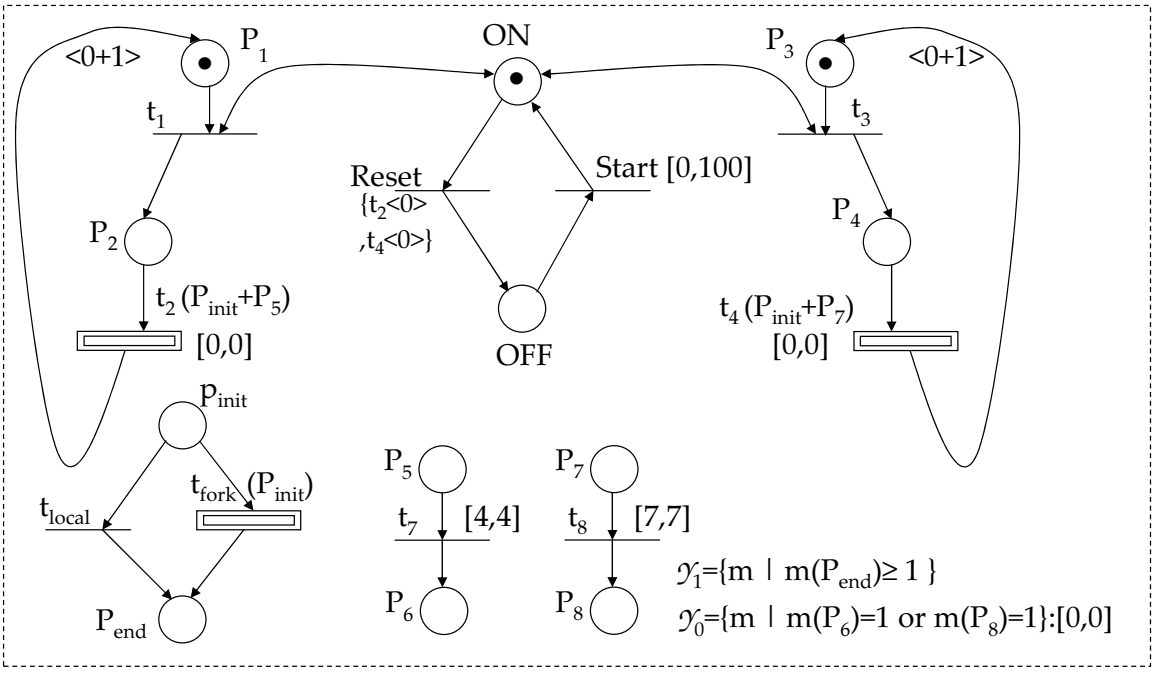

FIG. 5.1 – Une application RdPRT

obtenu par un tir de la transition t_{local}), soit terminer de manière "forcée" quand le temps qui lui est alloué expire. Dans ce dernier cas, ceci est possible par une présence d'un jeton dans la place P_6 ou la place P_8. La premier type de terminaison admet l'indice 1, alors que le deuxième l'indice 0. Les marquages finaux correspondant à ces deux types de terminaison sont données respectivement par les ensembles Υ_1 et Υ_0.

Quel que que soit le type de terminaison d'une transaction, une nouvelle requête de n'importe quel type est possible. Par conséquent, les arcs sortants de t_2 (resp. t_4) sont étiquetés par les deux indices de terminaison possibles (i.e. 0 et 1). Il est à noter qu'une terminaison "forcée", c'est à dire impliquée par un marquage appartenant à Υ_0 est immédiate (voir la contrainte temporelle attachée à l'ensemble Υ_0). Alors qu'une terminaison normale peut avoir lieu à n'importe quel moment (l'intervalle temporel $[0, \infty[$ est attaché à l'ensemble Υ_1).

Par ailleurs, un serveur peut être réinitialisé à n'importe quel moment (dans ce cas, la place OFF devient marquée suite au tir de la transition $Reset$). Dans ce cas, toute transaction en exécution correspondant soit à t_2, soit à t_4 est abortée immédiatement (dans tous les cas, l'indice de la termination est 0, tel que ceci est spécifié et attaché à la transition $Reset$). En effet, le tir de la transition $Reset$ permet de préempter (préemption externe) tout processus chargé d'exécuter une transaction de type 1 ou 2. Le serveur sera de nouveau opérationnel dans moins ou au plus tard dans 100 unités de temps (voir la transition $Start$). Ainsi, de nouvelles requêtes peuvent être prises en charge.

Cet exemple démontre les capacités du modèle RdPRT . En effet, il permet à la fois la préemption externe et interne. Par exemple, le tir de $Reset$ modélise le concept de préemption externe, alors que la coupure τ_1 est une modélisation d'une préemption interne. L'exemple montre aussi que le même composant peut être exécutée avec des contraintes temporelles différentes (les transactions de type 1 et 2 correspondent au même

composant mais ont des contraintes temporelles différentes). Par ailleurs, aucune contrainte temporelle n'est imposée à l'instant de tir de la transition t_2 (resp. t_4) (voir l'intervalle temporel attaché à t_2 (resp. t_4)), en revanche sa durée doit être inférieure ou égale à 4 unités de temps (resp. 7 unité de temps). C'est aussi une manière de définir des contraintes temporelles sur les instants de fermeture des transitions abstraites.

L'exemple montre également l'usage d'un appel récursif au sein d'une transaction (voir la transition abstraite t_{fork} et son marquage de départ). Par conséquent, le tir d'une telle transition induit à des séquences de tir (temporisés) infinies. En l'occurrence : (t_1,t_2,t_{fork},t_{fork},t_{fork},...). Ainsi, la profondeur des marquages étendus accessibles et le nombre de noeuds de ces marquages étendus sont infinis.

Notations
Les deux notations suivantes seront utilisées pour faciliter la description de la sémantique d'un RdPRT :
- $TC = T \cup \{\tau_i \mid i \in I_C\}$. Par exemple, $TC = \{t_1, t_2, t_3, t_4, t_5, \tau_1\}$ pour le RdPRT de la figure 5.2.

- $Vall = (\bigcup_{Tr \in Reach(R,m_0)} V(Tr))$ correspond à l'ensemble de tous les noeuds des marquages étendus accessibles d'un RdPR marqué. Considérons le RdPR sous-jacent au RdPRT marqué de la figure 5.2. Ce RdPR a été présenté dans le chapitre 2 (la figure 2.1) et une partie de son graphe des marquages étendus est donné dans la figure 2.2. L'ensemble $Vall$ correspondant à ce RdPR est $\{v_0, v_1\}$.

5.3 Sémantique d'un RdPRT

La sémantique d'un RdPRT peut être donnée en termes de systèmes de transitions temporisés qui sont des systèmes de transitions classiques telles que les différentes séquences d'exécution, appelées séquences de tir temporisées, utilisent deux types de transitions possibles : des transitions d'action et des transitions de temps modélisant respectivement des évolutions discrètes et des évolutions continues du système. Le modèle est basé sur un ensemble d'horloges, utilisant la même référence de temps, servant à conditionner les tirs à partir des états. Nous allons passer en revue les principaux aspects d'un RdPRT :

1. Un ensemble d'horloges dont les valeurs sont positives et réelles est associé à chaque marquage étendu accessible Tr ; en effet, une horloge est associée à chaque couple (v, t), avec $v \in V(Tr)$ et $t \in TC$. Contrairement à un T.RdPT, dans un RdPRT une transition ou une coupure peut avoir plusieurs comportements temporels concurrents : un comportement spécifique à chaque processus (ou encore à chaque noeud).

2. Etant donnée une séquence de tir temporisée, chaque état accessible est composé d'un marquage étendu accessible Tr et d'une valuation d'horloge $\nu c \in \mathbb{R}_{\geq 0}^{TC \times V(Tr)}$ (où $\mathbb{R}_{\geq 0}$ est l'ensemble des réels positifs), tel que $\nu c(t, v)$ représente le temps écoulé depuis que t est devenue sensibiliséee pour la dernière fois dans le noeud v. Une telle sensibilisation a eu lieu suite à la génération de Tr ou dans un de ses prédecesseurs dans la séquence. L'état initial du RdPRT correspond à $(Tr_0, \nu c_0)$, tel que $\forall t \in TC, \nu c_0(t, v_0) = 0$.

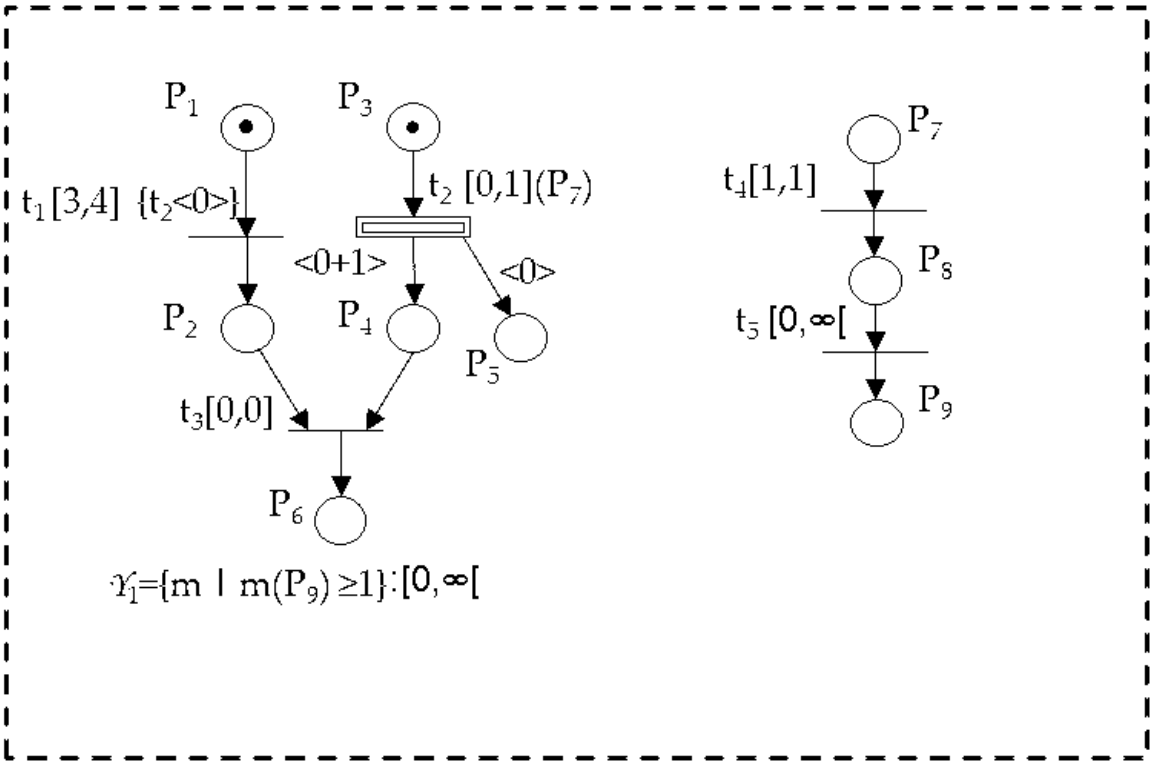

FIG. 5.2 – Un RdPRT simple

3. La durée d'un tir d'une transition ou une coupure à partir de n'importe quel état est toujours nulle. En revanche, un tel tir peut modifier l'ensemble des horloges puisque des noeuds peuvent être supprimés ou créés selon la nature de l'élément tiré. Ainsi, la suppression d'un noeud implique la suppression de toutes les horloges en relation avec le noeud. De même, des horloges sont générées et initialisées à zéro pour tout noeud fraichement créé.

Comme pour les autres modèles temporisés des réseaux de Petri, nous introduisons le concept de " transition nouvellement sensibilisée" qui définit les règles de mise à zéro des horloges associées aux transitions. Dans notre cas, ce concept doit être généralisé aux coupures et dépend des types de transitions.

Définition 5.2. (Transition ou coupure nouvellement sensibilisée) Considérons un tir $\langle V, M, E, A\rangle \xrightarrow{t,v} \langle V', M', E', A'\rangle$. On note par $new.enabled(t', v', M', M, t, v)$ pour indiquer que t' est nouvellement sensibilisée dans le noeud v' suite au tir de t à partir du noeud v (M et M' sont les fonctions de marquage respectives des noeuds v' et v). Les transitions ou coupures nouvellement sensibilisées après le tir de t à partir de v sont obtenues comme suit :

– $t \in T_{el}$: seul le noeud v peut avoir des transitions ou coupures nouvellement sensibilisées, plus précisément chaque transition ou coupure t' est nouvellement sensibilisée dans v après ce tir si et seulement si t' est sensibilisée par le marquage $M'(v)$ et, soit t' est non sensibilisée dans $(M(v) - {}^\bullet t)$ ou $(t' = t)$. Formellement,

-$\forall t' \in T \wedge v' \in V'$, $new.enabled(t', v', M', M, t, v) = (v' = v) \wedge ({}^\bullet t' \leq M'(v')) \wedge (({}^\bullet t' > M(v') -^\bullet t) \vee (t = t'))$,
-$\forall i \in I_C \wedge v' \in V'$, $new.enabled(t', v', M', M, t, v) = (v' = v) \wedge ((M(v') -^\bullet t) \notin \Upsilon_i) \wedge (M'(v') \in \Upsilon_i) \wedge (t' = \tau_i)$.

– $t \in \{\tau_i | i \in I_C\}$: seul le prédecesseur de v (i.e. $pred(v)$) peut avoir des transitions ou coupures nouvellement sensibilisées. Plus précisément, une transition ou une coupure est considérée nouvellement sensibilisée dans le noeud $pred(v)$ après ce tir ssi t' est sensibilisée par le nouveau marquage de $pred(v)$ et ne l'est pas par le marquage $M(pred(v))$. Formellement,

-$\forall t' \in T \wedge v' \in V'$, $new.enabled(t', v', M', M, t, v) = (v' = pred(v)) \wedge ({}^\bullet t' \leq M'(v')) \wedge ({}^\bullet t' > M(v'))$,
-$\forall i \in I_C \wedge v' \in V'$, $new.enabled(t', v', M', M, t, v) = (v' = pred(v)) \wedge (M(v') \notin \Upsilon_i) \wedge (M'(v') \in \Upsilon_i) \wedge (t' = \tau_i)$.

– $t \in T_{ab}$: seuls le noeud frais v' rajouté suite au tir de t et le noeud v peuvent avoir des transitions ou coupures nouvellement sensibilisées. Plus précisément, toutes les transitions et coupures sont nouvellement sensibilisées dans le noeud frais v'. De plus, t peut être nouvellement sensibilisée dans v si elle est sensibilisée par le nouveau marquage de (i.e. $M'(v)$). Formellement,

-$\forall t' \in TC \wedge v' \in V'$, $new.enabled(t', v', M', M, t, v) = (v' \in (V' \setminus V)) \vee ((v' = v) \wedge (t = t') \wedge ({}^\bullet t' \leq M'(v')))$.

Maintenant nous formalisons la sémantique d'un RdPRT marqué en terme d'un système de transitions temporisé.

Définition 5.3. (Sémantique Formelle d'un RdPRT) La sémantique formelle d'un RdPRT marqué RdPRT (R, m_0) est un système de transition S_{R,m_0} = $(\mathbb{Q}, q_0, \mapsto)$ où :

Soit $X = TC \times V(Tr)$
- $\mathbb{Q} = Reach(R, m_0) \times (\bigcup_{Tr \in Reach(R,m_0)} \mathbb{R}^X_{\geq 0})$,
- $q_0 = (Tr_0, \nu c_0)$,
- $\mapsto = \mathbb{Q} \times ((TC \times Vall) \cup \mathbb{R}_{\geq 0}) \times \mathbb{Q}$ consiste en des transitions discrètes ou continues définies comme suit :

1. La relation de transition discrète, dénotée par $(Tr, \nu c) \mapsto^{t,v} (Tr', \nu c')$, est définie, pour tout $t \in TC$ et $v \in V(Tr)$, par :
   (a) $Tr \xrightarrow{t,v} Tr'$,
   (b) $sEFT(t) \leq \nu c(t, v) \leq sLFT(t)$,
   (c) $\forall t' \in TC$ et $\forall v' \in V(Tr')$, $\nu c'(t', v') = 0$ ssi t' est nouvellement sensibilisée par $M'(v')$ (avec $M'(v')$ est un marquage ordinaire du noeud v' de Tr'), sinon $\nu c'(t', v') = \nu c(t', v')$.

2. La relation de transition continue, dénoté par $(Tr, \nu c) \mapsto^d (Tr', \nu c')$, est définie, pour n'importe quel

$d \in \mathbb{R}_{\geq 0}$, par :

(a) $Tr = Tr'$,

(b) $\forall t \in TC$ et $\forall v \in V$, $\nu c'(t, v) = \nu c(t, v) + d$,

(c) $\forall t \in TC$ et $\forall v \in V$, t est sensibilisée dans $M(v)$ (où $M(v)$ est le marquage ordinaire du noeud v de Tr) $\Rightarrow \nu c'(t, v) \leq sLFT(t)$.

Remarque : Etant donné un ensemble d'horloges associées au marquage étendu courant, une opération d'écoulement de temps concerne toutes les valuations d'horloges. Ainsi, celles-ci progressent de manière synchrone. Par ailleurs, l'écoulement du temps ne peut désensibiliser ni les transitions ni les coupures.

Exemple 5.2 : Considérons le RdPRT de la figure 5.2. et supposons qu'une valuation des horloges correspondant à un noeud v d'un marquage étendu accessible est $\nu c(v) = (x_1, x_2, x_3, x_4, x_5, r_1)$ où x_i et r_1 representent des valeurs réelles positives, associées respectivement à la transition t_i et la coupure τ_1. Les séquences de tir temporisées suivantes font abstraction des relations continues où le temps est nul :

Première séquence de tir temporisée [1] :

$$\begin{array}{lll} Tr_0 & \mapsto t_2, v_0 & Tr_1 \quad \mapsto 1 \\ \nu c(v_0) = (0,0,0,0,0,0) & & \nu c(v_0) = (0,0,0,0,0,0) \\ & & \nu c(v_1) = (0,0,0,0,0,0) \\ Tr_1 & \mapsto t_4, v_1 & Tr_2 \quad \mapsto t_5, v_1 \\ \nu c(v_0) = (1,1,1,1,1,1) & & \nu c(v_0) = (1,1,1,1,1,1) \\ \nu c(v_1) = (1,1,1,1,1,1) & & \nu c(v_1) = (1,1,1,1,0,1) \end{array}$$

$$\begin{array}{lll} Tr_3 & \mapsto \tau_1, v_1 & Tr_4 \\ \nu c(v_0) = (1,1,1,1,1,1) & & \nu c(v_0) = (1,1,1,1,1,1) \\ \nu c(v_1) = (1,1,1,1,0,0) & & \end{array}$$

Deuxième séquence de tir temporisée :

$$\begin{array}{lll} Tr_0 & \mapsto t_2, v_0 & Tr_1 \quad \mapsto 1 \\ \nu c(v_0) = (0,0,0,0,0,0) & & \nu c(v_0) = (0,0,0,0,0,0) \\ & & \nu c(v_1) = (0,0,0,0,0,0) \\ Tr_1 & \mapsto t_4, v_1 & Tr_2 \quad \mapsto 2 \\ \nu c(v_0) = (1,1,1,1,1,1) & & \nu c(v_0) = (1,1,1,1,1,1) \\ \nu c(v_1) = (1,1,1,1,1,1) & & \nu c(v_1) = (1,1,1,1,0,1) \end{array}$$

$$\begin{array}{lll} Tr_2 & \mapsto t_1, v_0 & Tr_7 \\ \nu c(v_0) = (3,3,3,3,3,3) & & \nu c(v_0) = (3,3,0,3,3,3) \\ \nu c(v_1) = (3,3,3,3,2,3) & & \end{array}$$

[1] Les marquages etendus de ce RdPRT sont donnés dans la figure 2.2 du chapitre 2.

5.4 Graphe des Classes d'États Étendu

Rappelons que les T.RdPT ont un modèle de temps dense, ainsi, leur espace d'états est potentiellement infini. Des techniques pour réduire cet espace d'états infini s'avèrent nécessaires : plusieurs techniques ont été introduites pour définir et calculer le graphe des classes d'états [22][5][52]. Ici, nous proposons une technique pour calculer un graphe des classes d'états étendu pour les RdPRT. Rappelons également qu'un état d'un T.RdPT est représenté par une paire, appelée classe, composée d'un marquage ordinaire et d'un domaine de tir définissant les instants de tir possibles.

Nous suivons une approche similaire pour les RdPRT, mais une classe doit référencer un marquage étendu, ce qui signifie une collection de marquages ordinaires. Un domaine de tir doit être associé à chacun de ces marquages ordinaires. Toutefois, en supposant une seule référence de temps pour l'ensemble de tous les processus d'un RdPRT, nous pouvons rassembler les différents domaines de tir du marquage étendu en un seul, représentant le domaine de la classe. Plus concrètement, une classe est une paire $C = (Tr; D)$ avec Tr son marquage étendu et D son domaine de tir. D'autre part, D est un système d'inéquations qui définit, pour chaque noeud v de Tr, les intervalles temporels dans lesquels peuvent être tirées les transitions ou les coupures sensibilisées au sein du noeud v. Ainsi, une variable d'un tel système sera matérialisée par une paire $\langle t, v\rangle$, avec v un noeud du marquage étendu de la classe (par abus de langage, on dira le noeud de la classe) et t est une transition ou une coupure ; une telle variable est appelée $t.v$. Il est à noter que la variable $t.v$ représente la date de tir de la transition ou la coupure t à partir du noeud v de la classe. De manière classique, une inéquation peut avoir l'une des formes suivantes :

1. $\alpha_v \leq t.v \leq \beta_v$, avec $t \in TC$ et t est sensibilisée par le marquage du noeud v.

2. $\gamma_{v,v'} \leq t.v - t'.v' \leq \gamma'_{v,v'}$, avec $t \in TC$ et $t' \in TC$, t est sensibilisée par le marquage de v et t' est sensibilisée par le marquage de v'.

Dans ces inéquations les bornes sont des constantes rationnelles, avec $\alpha_v, \gamma_{v,v'} \in \mathbb{Q}^+$ et $\beta_v, \gamma'_{v,v'} \in \mathbb{Q}^+ \cup \{\infty\}$. Rappelons que le nombre des constantes rationnelles $\alpha_v, \beta'_v, \gamma_{v,v'}$ et $\gamma'_{v,v'}$ est fini pour un T.RdPT [22]. Ce résultat est aussi valable pour un RdPRT puisque le calcul de ces constantes est similaire à celui des T.RdPT.

5.4.1 Construction des Classes Successeurs d'une Classe

La construction d'un graphe des classes d'états étendu se fait de manière classique. Elle consiste à déterminer toutes les classes accessibles à partir de la classe initiale.

La classe initiale La classe initiale d'un RdPRT est $C_0 = (Tr_0; D_0)$ avec D_0 défini comme suit :
$\forall t \in TC$, t est sensibilisée dans $v_0 \Rightarrow sEFT(t) \leq t.v_0 \leq sLFT(t)$ est une inéquation de D_0.

Egalité de deux classes Deux classes $C_1 = (Tr_1; D_1)$ et $C_2 = (Tr_2; D_2)$ sont égales ssi $Tr_1 = Tr_2$ et $D_1 = D_2$.

Définition 5.4. Une transition ou une coupure t est tirable à partir d'une classe $C = (Tr; D)$ si et seulement si les deux conditions suivantes sont vérifiées :

(i) Il existe un noeud v de Tr tel que t soit sensibilisée dans v.

(ii) D enrichi par les inéquations de l'ensemble suivant $\{t.v \leq t'.v' |\ t'.v'$ est une variable de D différente de $t.v\}$ admet une solution. Ces inéquations sont appelées les conditions de franchissabilité de t et v.

Le tir d'une transition ou une coupure t à partir d'un noeud v d'une classe $C = (\langle V, M, E, A\rangle; D)$ conduit à une autre classe $C' = (\langle V', M', E', A'\rangle; D')$. Cette action est dénotée par $C \xrightarrow{t,v} C'$. C' est calculée comme suit :
(i) $\langle V, M, E, A\rangle \xrightarrow{t,v} \langle V', M', E', A'\rangle$,

(ii) Le calcul de D' se fait en suivant les étapes suivantes :

1. $D' = D$.
2. Enrichir D' par les conditions de franchissabilité de t et v.
3. Retirer de D' les inéquations qui sont en relation avec les noeuds qui ont été supprimés suite au tir de t à partir de v.
4. Si t est une transition, alors éliminer de D' chaque variable associée à une transition t' (sauf t) tel que t et t' soit en conflit pour la marquage de v (i.e. t et t' sont sensibilisées par le marquage $M(v)$ et $\exists p \in {}^\bullet t \cap {}^\bullet t', M(v)(p) < W^-(p,t) + W^-(p,t')$).
5. Si t est une transition, alors éliminer de D' chaque variable associée à une coupure τ_i, avec $i \in I_C$, tel que t et τ_i soit en conflit pour le marquage de v (i.e. t et τ_i sont sensibilisées par le marquage $M(v)$ et $(M(v) - {}^\bullet t) \notin \Upsilon_i$).
6. Finalement, pour chaque noeud $v' \in V'$ et chaque transition ou coupure t' tel que t' est nouvellement sensiblisée dans v' après le tir de t de v (voir définition 5.2), nous introduisons une nouvelle inéquation $sEFT(t') \leq t'.v' \leq sLFT(t')$ dans D'.

Exemple 5.3 : Considérons le réseau de la figure 5.2 et supposons maintenant que le marquage initial est m_0=($2P_1, P_3$). une séquence du graphe des classes d'états étendu est définie comme suit :

$C_0 \xrightarrow{t_2,v_0} C_1 \xrightarrow{t_4,v_1} C_2 \xrightarrow{t_5,v_1} C_3 \xrightarrow{\tau_1,v_1} C_4 \xrightarrow{t_1,v_0} C_5 \xrightarrow{t_3,v_0} C_6 \xrightarrow{t_1,v_0} C_7$. Ces classes sont décrites dans la table 5.1.

TAB. 5.1 – Graphe de classes d'états étendu

Nom de la classe	Marquage étendu	Domaine de tir	Nom de la classe	Marquage étendu	Domaine de tir
C_0	$\bullet v_0\ (2P_1, P_3)$	$3 \leq t_1.v_0 \leq 4$ $0 \leq t_2.v_0 \leq 1$	C_1	$\bullet v_0\ (2P_1)$ $\downarrow t_2$ $\bullet v_1(P_7)$	$2 \leq t_1.v_0 \leq 4$ $1 \leq t_4.v_1 \leq 1$
C_2	$\bullet v_0\ (2P_1)$ $\downarrow t_2$ $\bullet v_1\ (P_8)$	$1 \leq t_1.v_0 \leq 3$ $0 \leq t_5.v_1 \leq \infty$	C_3	$\bullet v_0\ (2P_1)$ $\downarrow t_2$ $\bullet v_1\ (P_9)$	$0 \leq t_1.v_0 \leq 3$ $0 \leq \tau_1.v_1 \leq \infty$
C_4	$\bullet v_0\ (2P_1, P_4)$	$0 \leq t_1.v_0 \leq 3$	C_5	$\bullet v_0\ (P_1, P_2, P_4)$	$3 \leq t_1.v_0 \leq 4$ $0 \leq t_3.v_0 \leq 0$
C_6	$\bullet v_0\ (P_1, P_6)$	$3 \leq t_1.v_0 \leq 4$	C_7	$\bullet v_0\ (P_2, P_6)$	

5.5 Propriétés Théoriques des RdPRT

Dans cette section, nous nous focalisons sur la bornitude et la finitude des RdPRT marqués. Rappelons que, dans un RdPR, la propriété de bornitude assure qu'il existe une valeur entière n telle que, au sein de n'importe quel noeud, le nombre de jeton(s) de chaque place est inférieur ou égal à n. La propriété de finitude assure que le nombre de marquages étendus accessibles est fini. Ces deux propriétés sont décidables mais non équivalentes pour un RdPR.

Théorème 5.1. Les problèmes d'accessibilté et de bornitude sont indécidables pour les RdPRT.

Preuve : la preuve est directe puisqu'un T.RdPT est un cas particulier d'un RdPRT (sans transition abstraite) et ces problèmes sont indécidables pour un T.RdPT.
Dans ce qui suit, nous définissons les conditions de finitude d'un RdPRT.

Lemme 5.1. Soit un RdPRT marqué, l'ensemble de tous les domaines de tirs du réseau est fini ssi la profondeur des marquages étendus des classes d'états accessibles est finie.

Preuve : Nous procédons par contradiction. ($\Leftarrow$) Supposons que l'ensemble des domaines de tir d'un RdPRT est infini. Ceci implique que le nombre de variables utilisées pour décrire les différents domaines de tir est infini puisque comme pour un T.RdPT, le nombre des constantes α_v, β_v, $\gamma_{v,v'}$ et $\gamma'_{v,v'}$ apparaissant dans les domaines de tir est fini [22]. De plus, les variables $\langle t.v \rangle$ utilisées pour décrire les domaines de tir des classes d'états accessibles dépendent d'une part des transitions ou coupures, et d'autre part des noeuds des marquages étendus de ces classes. Par conséquent, nous déduisons qu'un nombre infini de variable implique un nombre infini des noeuds des marquages étendus accessibles (puisque le nombre de transitions et coupures est fini). Par conséquent, la profondeur des marquages étendus accessibles est infinie. ($\Leftarrow$) Supposons que la profondeur des marquages étendus accessibles est infini. Il s'ensuit que le nombre de noeuds de ces marquages étendus est infini. Il est évident que le nombre de variables $\langle t.v \rangle$ définissant les domaines de tir est infini.
Dans ce qui suit, nous établissons des critères pour caractériser la finitude d'un RdPRT. Pour ce faire, nous

avons besoin de caractériser la non bornitude de leurs réseaux sous-jacents (i.e. les RdPR). Nous allons montrer comment nous avons adapté les critères de non bornitude des RdP classiques aux RdPR.

5.5.1 Critère de non Bornitude des RdPR

Dans cette section, nous allons définir des critères afin de caractériser la non bornitude d'un RdPRT. Dans un premier temps, nous écartons le concept de préemption ; en d'autres termes, les transitions élémentaires sont classiques. Nous reprenons d'abord une proposition caractérisant la non bornitude des RdP ordinaires.

Proposition 5.1. (Caractérisation d'un réseau de Petri ordinaire borné) Si le RdP marqué (R, m_0) admet une séquence de franchissement $m_0 \xrightarrow{\sigma_1} m_1 \xrightarrow{\sigma_2} m_2$, avec $m_1 < m_2$, alors (R, m_0) est non borné.

Cette caractérisation de non bornitude des RdP ne s'adapte pas aux marquages ordinaires des processus d'un RdPR. En effet, si un réseau récursif marqué (R, m_0) admet une séquence de franchissement $Tr_0 \xrightarrow{\sigma_1} Tr_1 \xrightarrow{\sigma_2} Tr_2$, tel qu'il existe un noeud commun $v \in V(Tr_1) \cap V(Tr_2)$ et le marquage de v dans Tr_2 soit strictement supérieur à son marquage dans Tr_1, alors ceci ne prouve pas la non bornitude de (R, m_0). Pour prouver cela, il suffit de prendre σ_2 égale à une coupure tirée à partir d'un successeur direct de v. Dans ce cas, le marquage de v dans Tr_2 grossit par rapport à son marquage dans Tr_1 (voir règle de tir d'une coupure) même quand le réseau récursif est borné. Considérons la séquence $Tr_3 \xrightarrow{\tau_1, v_1} Tr_4$ du graphe des marquages étendus de la figure 2.2. Le marquage de v_0 dans Tr_4 est strictement supérieur à son marquage dans Tr_3 sans pour autant que le RdPR en question soit non borné.

Proposition 5.2. (Caractérisation d'un réseau récursif borné sans préemption externe) Un réseau de Petri récursif marqué (R, Tr_0) est non borné, si il existe une séquence de franchissement $Tr_0 \xrightarrow{\sigma_1} Tr_1 \xrightarrow{\sigma_2} Tr_2$, tel que :

1. Il existe un noeud $v \in V(Tr_1) \cap V(Tr_2)$ et le marquage de v dans Tr_2 soit strictement supérieur à son marquage dans Tr_1.

2. Soit σ_2 ne comporte aucun tir d'une coupure à partir d'un successeur direct du noeud v (cas 1), soit elle en contient mais, dans ce cas de figure, la condition suivante doit être vérifiée (cas 2) : pour chaque tir d'une coupure, à partir d'un successeur direct de v et visant à fermer une transition abstraite donnée, celle-ci est franchie, dans σ_2, à partir de v. De plus, dans la séquence σ_2, un tir d'une transition abstraite t à partir de v est combiné avec au plus un tir d'une coupure à partir d'un successeur direct de v.

Le Premier cas : tous les tirs à partir du noeud v concernent des tirs de transitions élémentaires ou abstraites. Puisque le marquage de v dans Tr_2 est strictement supérieur à son marquage dans Tr_1, les transitions de σ_2 tirées à partir de v peuvent être franchies de nouveau à partir du noeud v de Tr_2. On obtient alors un marquage étendu Tr_3 tel que le marquage de v dans Tr_3 soit strictement supérieur à son marquage dans Tr_2. En itérant ce procédé, on obtient la séquence infinie $\sigma_1.\sigma_2^{\infty}$. D'autre part, il existe forcément une place p dont le marquage, au sein du noeud v, croît indéfiniment.

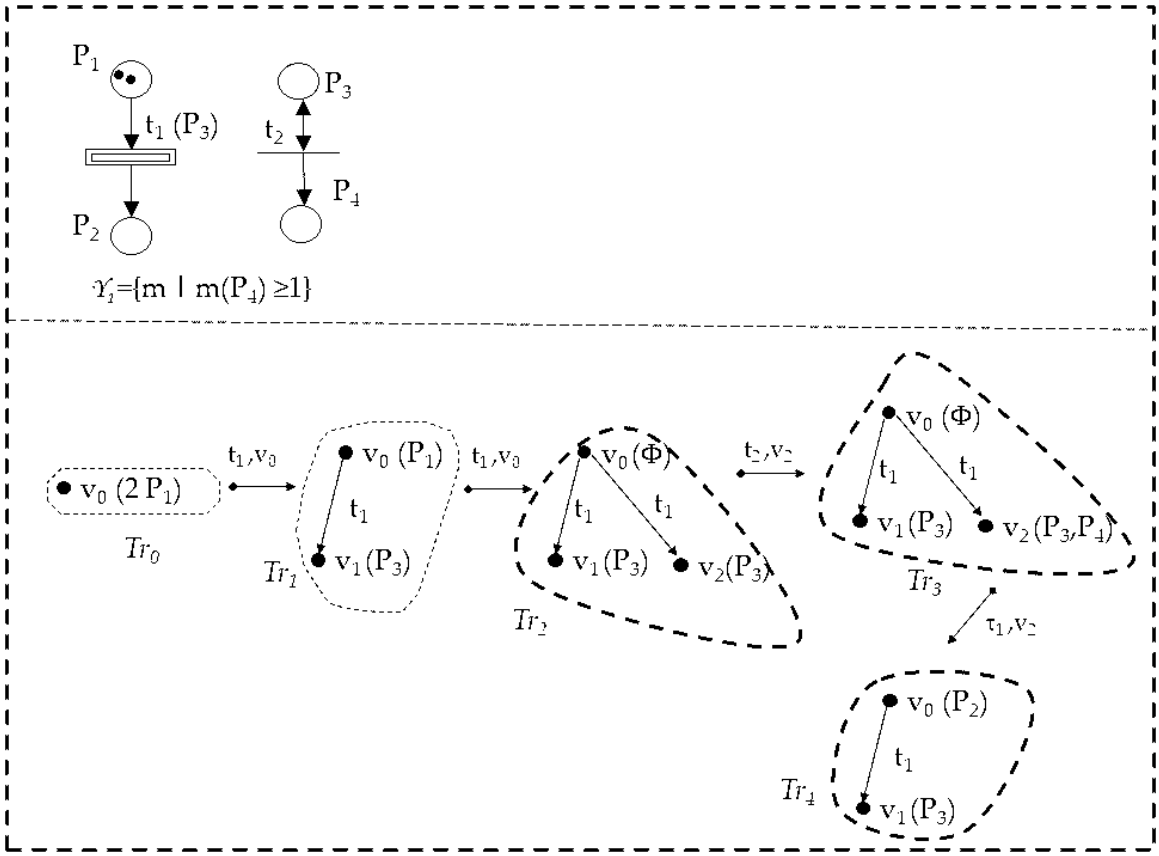

FIG. 5.3 – Un RdPR simple

Le Deuxième cas : nous transformons la séquence σ_2 en une séquence *équivalente* en éliminant les coupures tirées à partir des successeurs directs de v. Il suffit de combiner le tir de chaque coupure d'un succcesseur direct de v avec un tir, à partir de v, de la transition abstraite concernée par la coupure. Les deux tirs combinés sont simulés à l'aide d'une transition élémentaire qui a les mêmes pré-conditions et post-conditions que la transition abstraite (la sémantique des RdPR permet une telle simulation). Ainsi, la séquence σ_2 est transformée en une séquence sans aucun tir de coupure à partir d'un successeur direct de v. Par conséquent, on se ramène au premier cas.

Proposition 5.3. (Noeud révélateur de non bornitude d'un RdPR) Soient la séquence $Tr_1 \xrightarrow{\sigma_2} Tr_2$ d'un RdPR marqué et v un noeud commun à Tr_1 et Tr_2 répondant aux deux critères suivants : (1) son marquage dans Tr_2 est strictement supérieur à son marquage dans Tr_1. (2) De plus, il n'existe pas une transition abstraite t, tel que le nombre des arcs sortants de v étiquetés par t dans Tr_2 soit strictement inférieur au nombre de ses arcs sortants étiquetés par la même transition dans Tr_1. On dira qu'un tel noeud est révélateur de non bornitude du RdPR.

Exemple 5.4 : Considérons le RdPR marqué de la figure 5.3 et le tir $Tr_2 \xrightarrow{t_2,v_2} Tr_3$ (voir la partie basse de la figure 5.3), le noeud v_2 commun à Tr_2 et Tr_3 est révélateur de non bornitude du RdPR ; en effet, v_2 répond aux deux critères d'un noeud révélateur de non bornitude d'un RdPR. En revanche, en considérant la

séquence $Tr_3 \xrightarrow{\tau_1,v_2} Tr_4$ du même RdPR, le noeud v_0 répond au premier critère et non au deuxième critère. Bien que son marquage croît suite au tir de la séquence, v_0 n'est pas révélateur de non bornitude du RdPR. En considérant la séquence $Tr_2 \xrightarrow{t_2,v_1} Tr_3 \xrightarrow{\tau_1,v_2} Tr_4$, la même remarque peut être faite pour le noeud v_0 de Tr_2 et Tr_4.

Remarque : Les caractérisations de non bornitude des RdPR sans préemption externe restent valables pour les RdPR avec préemption externe. En effet, le tir d'une transition élémentaire, à partir d'un noeud donné, contrôlant une transition abstraite peut être transformé en un tir, à partir du même noeud, d'une transition élémentaire classique suivi éventuellement, par le(s) tir(s) de coupure(s) visant à fermer la transition contrôlée.

Théorème 5.2. Un RdPRT est fini si les deux points suivants sont vérifiés :

1. Il n'existe aucune classe d'états accessible dont le marquage étendu contient un chemin contenant au moins deux arcs étiquetés par la même transition abstraite.

2. Il n'existe aucune classe $C' = (Tr'; D')$ accessible à partir d'une autre classe $C = (Tr; D)$ tel qu'un noeud commun v de Tr et Tr' soit révélateur de non bornitude du RdPR sous-jacent.

Preuve : Supposons qu'un RdPRT est infini, donnant lieu à un nombre infini de classes d'états distinctes. Ici, nous avons deux cas possibles :

(i) Il y a un nombre infini de domaines de tir. Nous déduisons, à partir du lemme 5.1, que la profondeur des marquages étendus apparaissant dans les classes d'états accessibles est infini. Selon [29], une telle profondeur infinie est due à une répétition d'étiquettes sur les arcs d'un même chemin d'un marquage étendu accessible. Par conséquent, il existe une classe d'états violant le premier point du théorème.

(ii) Contrairement au nombre des marquages étendus contenus dans les classes accessibles, le nombre de domaines de tir est fini. Dans ce cas, d'après le lemme 5.1, le nombre de noeuds des variables définissant les domaines de tir est fini. Il s'ensuit que la non finitude des marquages étendus des classes d'états accessibles est due forcément à un noeud dont le marquage croît indéfiniment (i.e. il s'agit d'un noeud révélateur de non bornitude). C'est le deuxième point du théorème.

5.6 Graphe des Classes d'États Modulaire

Dans cette section, nous nous intéressons à des RdPRT sans indice et sans préemption externe. Nous allons proposer une seconde approche pour la construction d'un espace d'états d'un RdPRT *borné* et *fini*. Dans cette nouvelle approche, un graphe des classes d'états *classique* est généré pour **chaque transition abstraite**, y compris pour une **transition abstraite virtuelle** chargée de créer le processus initial de tout RdPRT.

5.6.1 Informations Temporelles d'un Graphe des Classes d'États

D'abord, nous allons définir une procédure qui permet d'associer des informations temporelles à un graphe des classes d'états *classique*, *borné* et *fini*. Ces informations donnent les intervalles d'exécution des séquences de transitions du graphe. Dans ce qui suit, nous nous intéressons uniquement à des séquences de transitions qui mènent, à partir de la classe initiale du graphe, à des *classes finales* (i.e. marquées par des marquages finaux). De plus, une transition élémentaire virtuelle, que l'on nomme τ, est rajoutée à la fin de chaque séquence. Cette dernière simule une coupure. De ce fait, elle est sensibilisée que pour des marquages finaux et est soumise aux mêmes contraintes temporelles qu'une coupure. Nous allons considérer les séquences de tir du graphe. Il est à noter que si le nombre de celles-ci est infini, cela est du forcément à la présence d'au moins un circuit dans le graphe (puisque le graphe est borné et fini). Deux cas sont à considérer :

1. Le graphe à analyser ne contient aucun circuit. Dans ce cas de figure, le nombre de séquences de tir du graphe est fini. Par ailleurs, pour chaque séquence, nous calculons l'intervalle $[min, max]$ où min (resp. max) est la durée minimale (resp. maximale) de la séquence (pour cela, nous utilisons les techniques de calcul des durées des séquences présentées dans le chapitre 3). Nous faisons l'union disjointe de ces intervalles temporels.

2. Le graphe à analyser contient au moins un circuit. Dans ce cas, si aucune classe finale n'est accessible à partir d'une autre classe qui d'une part apparaît dans un circuit du graphe, et d'autre part est accessible à partir de la classe initiale alors on se ramène au cas précédent. Ainsi, les classes des circuits sont exclues des séquences à considérer puisqu'elles ne mènent pas à des marquages finaux. Par conséquent, le nombre de ces séquences est fini. Il faut souligner que l'on se ramène à un problème d'accessibilité, donc à un problème décidable. En revanche, dans le cas contraire, un seul intervalle temporel dont la borne maximale est infinie sera associé au graphe (on peut tirer des transitions du graphe un nombre quelconque de fois avant d'atteindre une classe finale). Pour la borne inférieure de cet intervalle, on va considérer le plus petit temps d'exécution des séquences de tir du graphe mais sans boucler sur les transitions des circuits .

5.6.2 Principe de Construction

Partant d'un RdPRT marqué (R, m_0), nous allons montrer comment construire la collection des graphes des classes d'états associés aux transitions abstraites de R. La procédure commence par générer l'espace d'états associé à la transition abstraite virtuelle. Pour cela, le RdPRT R est transformé en un T.RdPT R_{TPN} en substituant chaque transition abstraite t par une transition élémentaire fraiche t^s, simulant le tir de t, qui a les pré-conditions et l'intervalle temporelle statique de t. Le réseau temporel R_{TPN} sera enrichi par d'autres transitions élémentaires au cours de la construction des espaces des transitions abstraites. Ainsi, chaque transition abstraite sera substituée par un réseau ordinaire, que l'on nomme le réseau d'interface de la transition. Ce processus de substitution d'une transition abstraite permet de reproduire le comportement de la transition. En d'autres termes, il reproduit d'une part le tir de la transition dans son intervalle temporel, et d'autre part la fermeture de la transition en tenant compte des intervalles d'exécution

des séquences de transitions induites par le tir de la transition. Par ailleurs, le réseau ordinaire simulant une transition abstraite fait abstraction au comportement local de la transition.

La génération du graphe des classes d'états associé à la transition virtuelle se fait de manière classique, sauf que lorsqu'une transition t^s simulant le tir d'une transition abstraite est choisie pour être tirée , la construction du graphe en cours est interrompue. Et on commence la génération du graphe des classes d'états associé à la transition abstraite t, i.e. l'espace d'états du réseau T.RdPT marqué $(R_{TPN}, \Omega(t))$. Ainsi, la collection des espaces d'états se fait en cascade. Un mécanisme de détection des transitions récursives est effectué pour éviter que la procédure proposée boucle infiniment. Quand la construction d'un espace d'états associé à une transition abstraite t est achevée, alors, si celle-ci est fermable, on calcule l'union disjointe des intervalles d'exécution des séquences du graphe des classes d'états associés à t. Ici, nous considérons deux cas :

- L'union calculée se réduit à un seul intervalle temporel, dénoté par I : une transition élémentaire fraiche simulant la fermeture de t est rajoutée à R_{TPN}. Cette transition a les post-conditions de t et a I comme intervalle temporel statique. Par ailleurs, elle est reliée à t^s par une place fraiche non marquée.

- L'union calculée comporte n intervalles temporels $I_1 \cup ... \cup I_n$, avec $n > 1$: une transition élémentaire fraiche simulant la fermeture de t est rajoutée à R_{TPN}. Cette transition a les post-conditions de t et a I_1 comme intervalle temporel statique. Par ailleurs, elle est reliée à t^s par une place fraiche non marquée. Pour chaque I_i , avec $i > 1$, on rajoute à R_{TPN}, deux transitions élémentaires fraiches simulant respectivement le tir et la fermeture de t. La première transition rajoutée a les pré-conditions et l'intervalle statique de t. Alors que la deuxième a les post-conditions de t et I_i comme intervalle statique. Les deux transitions sont reliées par une place fraiche.
De plus, les places rajoutées sont dites des places de liaison. La procédure *récursive* $class\text{-}graph$ permet de construire la collection des graphes des classes d'états d'un RdPRT marqué (R, m_0). L'algorithme 3 utilise, pour chaque espace associé à une transition abstraite t, un ensemble $Waiting(t)$ de noeuds (classes) déjà créés mais non encore traités. La fonction $Node\text{-}class(C, t)$ crée un noeud C dans l'espace associé au graphe associé à t et le rajoute à $Waiting(t)$, si il n'existe pas déjà. La fonction $Arc\text{-}class(C, t', C', t)$ crée un nouveau arc ($C \xrightarrow{t'} C'$) au graphe associé à la transition abstraite t. $Compute$, appliqué au graphe des classes d'états associé à la transition abstraite t, donne les intervalles d'exécution des séquences de transitions du graphe associé à t. La fonction $convert$, appliquée à un RdPRT, retourne un T.RdPT tel que chaque transition abstraite t est substituée par une transition élémentaire t^s ayant les pré-conditions et l'intervalle temporel statique de t. La fontion $Link$ permet de rajouter à un T.RdPT des transitions élémentaires fraiches et des places fraiches afin de compléter la construction du réseau d'interface d'une certaine transition abstraite. La phase d'initialisation de la procédure de construction correspond aux instructions suivantes :

1. $R_{TPN} = Convert(R)$;
2. **Stack** $stack$;
3. $stack.empty()$;
4. $class\text{-}graph(t_{virtuelle}, m_0)$;

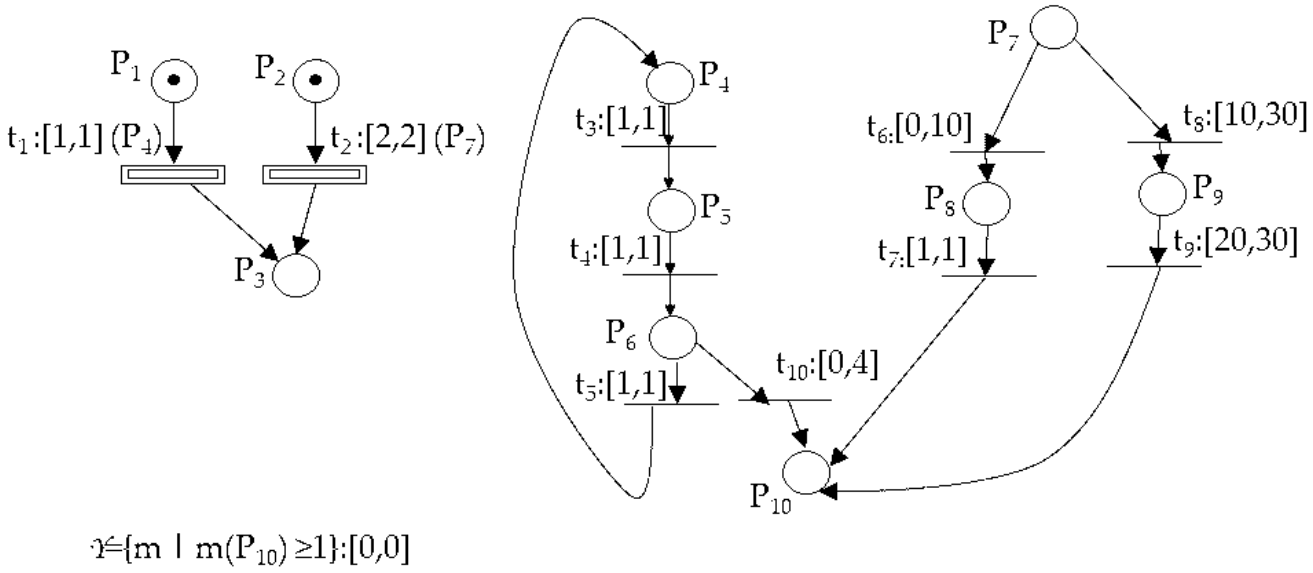

FIG. 5.4 – Un RdPRT simple R_1

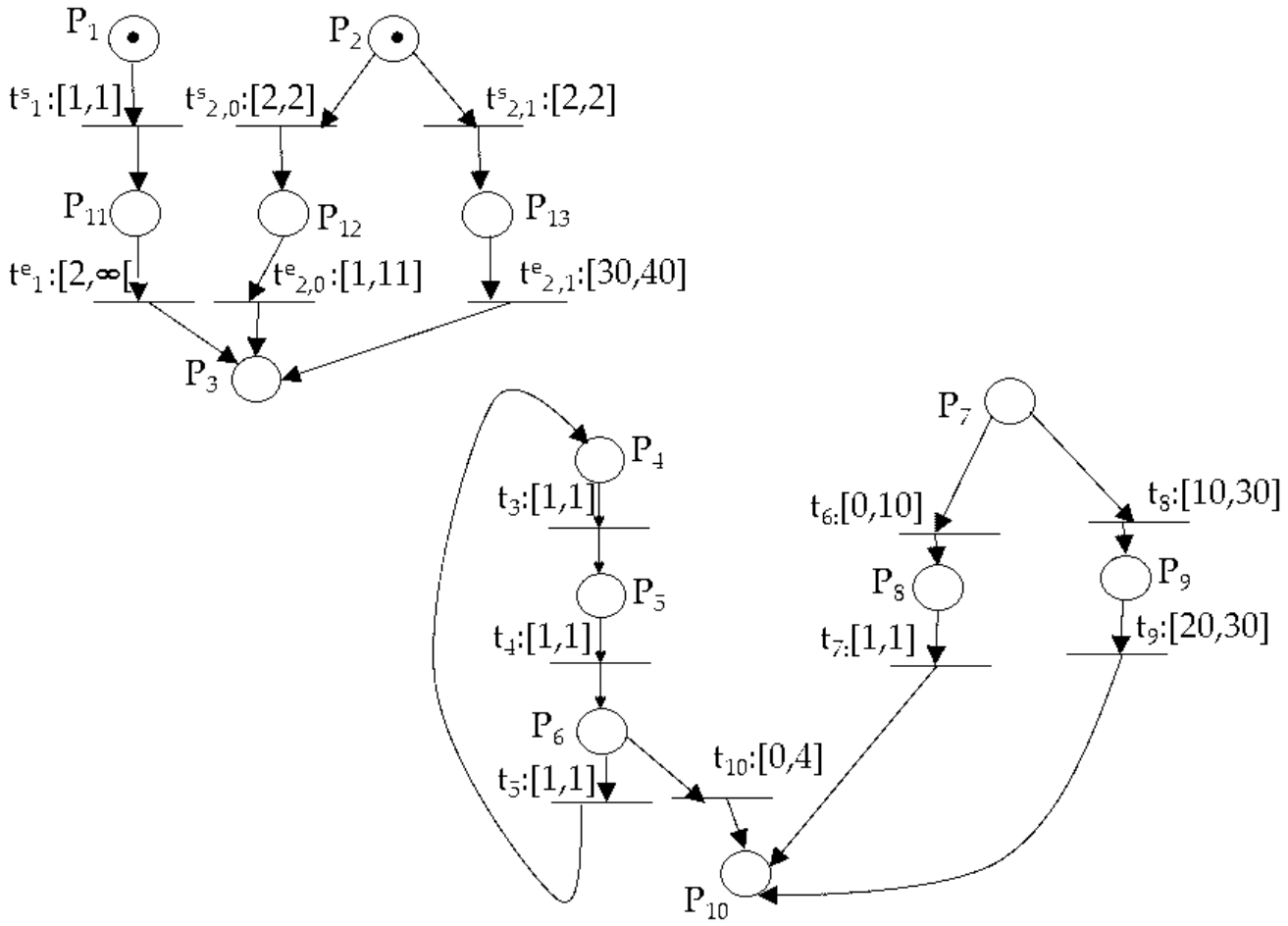

FIG. 5.5 – Le T.RdPRT associé à R_1

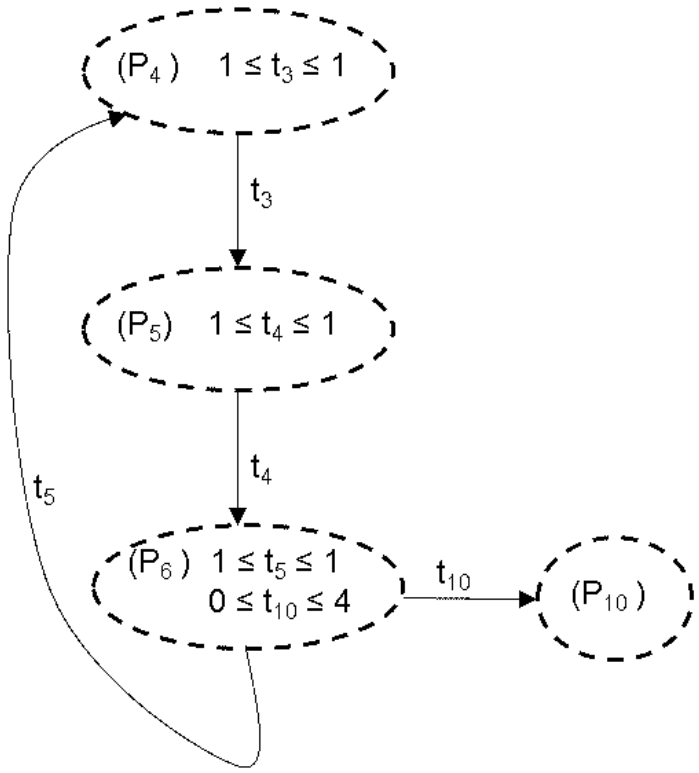

FIG. 5.6 – Le graphe des classes d'états associé à t_1

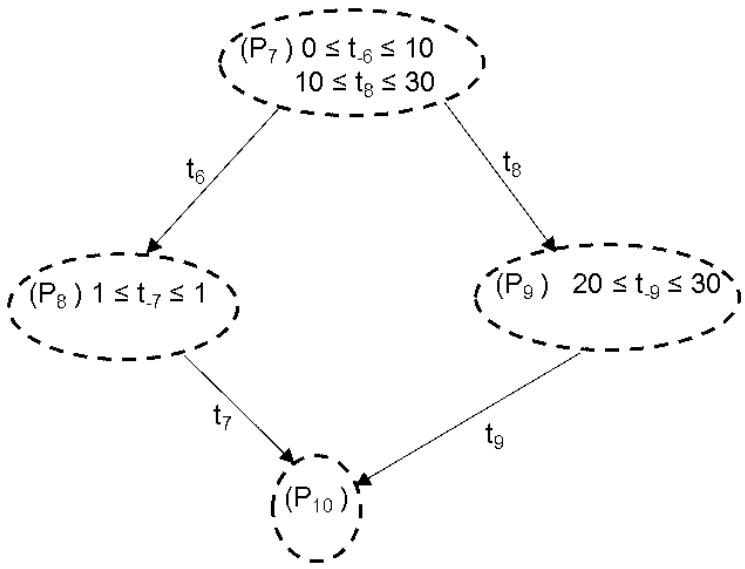

FIG. 5.7 – Le graphe des classes d'états associé à t_2

Algorithm 3 $class\text{-}graph$ (Transition abstraite : t_{ab}, marquage ordinaire : m_0)

1: $Node\text{-}class(C_0, t_{ab})$ /*C_0 est la classe initiale du T.RdPT (R_{TPN}, m_0)*/
2: WHILE $waiting(t_{ab}) \neq \emptyset$
3: **Choisir** $C \in waiting(t_{ab})$;
4: **for** chaque transition élémentaire t de R_{TPN} **do**
5: **if** $C \xrightarrow{t} C'$ **then**
6: **if** t est une transition élémentaire rajoutée à R_{TPN} simulant le tir d'une transition abstraite t'_{ab} **then**
7: **if** t'_{ab} existe dans $stack$ **then**
8: GOTO End-etiq /* Cas d'un graphe de classes d'états infini*/
9: **end if**
10: stack.push(t'_{ab}) ;
11: $class\text{-}graph(t'_{ab}, \Omega(t'_{ab}))$;
12: $Compute(t'_{ab}, I_1, ..., I_n)$;/* $I_1 \cup ... \cup I_n$ est l'union disjointe des intervalles d'exécution des séquences du graphe associé à t'_{ab}*/
13: $Link(R_{TPN}, t'_{ab}, I_1, ..., I_n)$;
14: $stack.pop()$;
15: **end if**
16: $Node\text{-}class(C', t_{ab})$;
17: $class\text{-}arc(C, t, C', t_{ab})$;
18: **end if**
19: **end for**
20: End-etiq : **END**

Exemple 5.5 : Considérons le RdPRT de la figure 5.4. La construction du graphe des classes d'états associé à la transition abstraite t_1 s'achève en premier lieu (voir figure 5.6). L'analyse de ce graphe révèle l'existence d'un circuit ; de plus, la classe finale (marquée par un marquage final) du graphe est accessible à partir de n'importe quelle classe du circuit. Par conséquent, un seul intervalle temporel dont la borne maximale est infinie est associé à ce graphe. En revanche, la borne inférieure de cet intervalle est calculée en considérant les séquences d'exécution du graphe sans boucler sur les transitions du circuit (ici cet ensemble de séquences se réduit à deux séquences $t_3t_4t_{10}\tau$ et $t_3t_4t_5\tau$). La plus petite durée de ces séquences correspond à 2 unités de temps. Ainsi, la transition abstraite sera simulée par deux transitions élémentaires : la première t_1^s simule le tir de t_1 et la deuxième t_1^e simule la fermeture de t_1 (voir le T.TdRPT de la figure 5.5). La construction du graphe des classes d'états associé à la transition abstraite t_2 s'achève en deuxième lieu (voir figure 5.7). L'analyse de ce graphe révèle l'existence de deux séquences d'exécution : $t_6t_7\tau$, $\theta_6 \in [0, 10]$, $\theta_7 \in [1, 1]$ et $\theta_\tau \in [0, 0]$ et $t_8t_9\tau$, $\theta_8 \in [10, 10]$, $\theta_9 \in [20, 30]$ et $\theta_\tau \in [0, 0]$ (les θ_i et θ_τ sont des dates relatives de tir). Les durées respectives de la première séquence et de la deuxième séquence sont comprises dans les intervalles $[1, 11]$ et $[30, 40]$. L'union temporelle de ces deux intervalles est $[1, 11] \cup [30, 40]$. La transition t_2 est simulée soit par $t^s_{2,0}$ et $t^e_{2,0}$, soit par $t^s_{2,1}$ et $t^e_{2,1}$ [2](voir le T.RdPT présenté dans la figure 5.5). Le graphe des classes d'états associé à la transition virtuelle est donné dans la figure 5.8.

L'exemple que nous venons de présenter illustre la deuxième méthode de construction d'un espace

[2]La transition t_2^s est renommée : $t^s_{2,0}$.

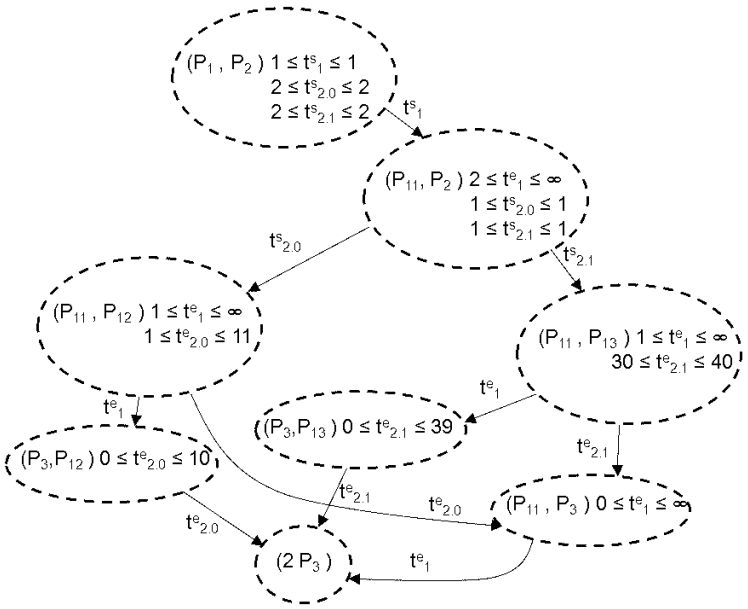

FIG. 5.8 – Le graphe des classes d'états associé à la transition abstraite virtuelle

d'états d'un RdPRT. Cet espace est une collection de graphes des classes d'états classiques. Un graphe sera associé à chaque processus créé donnant son *comportement temporel*. Lors de la phase d'analyse, ces graphes sont utilisés *simultanément*. Par exemple, le nombre de jetons dans une place de liaison donne le nombre de processus en cours créés suite au tir d'une certaine transition. Cette deuxième méthode d'analyse présente les avantages suivants :

1. Les classes d'états manipulent des marquages ordinaires et non des marquages étendus. Ainsi, la comparaison des classes implique la comparaison de marquages ordinaires et non de marquages étendus. Il est évident que comparer des vecteurs est une opération plus rapide que celle visant à comparer des arbres.

2. Possibilité d'utiliser les outils d'analyse des T.RdPT tels que Tina ou Roméo puisqu'on manipule des graphes des classes d'états classiques.

3. Au lieu de traiter un *domaine de tir global* pour l'ensemble des processus, on traite, pour chaque processus, son *domaine de tir de manière séparée*.

5.7 Conclusion

Dans ce chapitre, nous avons étendu les réseaux de Petri récursifs avec le paramètre temps. Le modèle obtenu (i.e. les RdPRT) hérite, par construction, de tous les concepts de modélisation des RdPR [29]. En particulier, la même description d'un composant connexe peut être utilisée par des contextes d'exécution différents (marquages différents) et avec des contraintes temporelles différentes, alors qu'une modélisation équivalente par des T.RdPT nécessite une représentation *explicite* de chaque contexte d'exécution et donc

une duplication du composant. Par ailleurs, les RdPRT ont la capacité d'interrompre un composant sous certaines conditions temporelles en tirant *uniquement* une seule transition élémentaire (préemption externe). Alors que la modélisation d'une telle interruption à l'aide d'un T.RdPT (ou encore à l'aide d'un RdP) est une tâche complexe puisqu'elle nécessite de bloquer chaque transition du composant. Par conséquent, les RdPRT sont plus compacts et lisibles que les T.RdPT. De plus, les RdPRT ont une plus grande expressivité que les T.RdPT en termes de systèmes infinis. Par ailleurs, nous avons proposé une première méthode de calcul de l'espace d'états des RdPRT. L'étude des propriétés théoriques du modèle proposé nous a amenés à définir des critères de non bornitude du modèle sous-jacent (i.e. les RdPR). Nous avons proposé une deuxième méthode de calcul de l'espace d'états des RdPRT. Cette deuxième technique est basé sur les travaux de Boucheneb concernant les calculs des durées des séquences d'exécution. Nous avons comparé les deux techniques. Dans le chapitre suivant, nous utilisons les RdPRT pour faire un contrôle de la cohérence temporelle des documents multimédias.

Chapitre 6

Modélisation de Documents SMIL à L'aide des Réseaux de Petri Récursifs Temporels

6.1 Introduction

Les documents multimédias intéractifs sont utilisés dans divers domaines et sont accessibles via le web. Plusieurs langages et outils tels que MADEUS [38] et SMIL [65] sont utilisés pour décrire les comportements spatiaux et temporels de ces documents. Par ailleurs, le langage SMIL (Synchronized Multimedia Integration Language) a été proposé par le consortium W3C (World Wide Web Consortium) [65] et est considéré jusqu'à présent comme l'approche la plus classique pour la description des présentations multimédias qui peuvent être intégrées dans le web. Grâce au modèle temporel de SMIL, un auteur peut spécifier des relations de synchronisation entre les objets multimédias qu'il manipule. De telles relations constituent un ensemble de contraintes temporelles qui doivent être satisfaites lors de la présentation du document. Malheureusement des situations d'incohérences peuvent surgir ("interblocage", "média jamais joué", "référence à un média inexistant", ect...). Il s'ensuit que des méthodes de détection de telles situations doivent être utilisées. Parmi ces méthodes, nous pouvons citer la méthode utilisée par Madeus [38]. Dans cette approche, un document est traduit en un graphe où les noeuds correspondent à des évènements tels que le début ou la fin d'un objet média et les arcs correspondent à des relations entre ces évènements. Ces relations prennent en charge des paramètres métriques des objets tels que les durées des objets multimédias. Des algorithmes tels que la détection de cycle sont appliqués à ces graphes afin de contrôler la cohérence temporelle et d'autres propriétés du document à analyser. Il faut toutefois noter que cette méthode ne prend pas en charge certains aspects dynaniques des documents multimédias tels que les liens et les durées non contrôlables des objets.
Dans [18], une autre méthodologie, basée sur le système RT-LOTOS, pour l'analyse et le contrôle de la cohérence temporelle des documents SMIL est proposée. Dans cette seconde approche, des opérateurs de composition sont utilisés pour combiner des objets multimédias afin de décrire le comportement temporel des documents. Il faut noter que, contrairement à la première approche, l'aspect dynamique des documents multimédia est bien pris en charge.

Dans ce chapitre, nous montrons que notre modèle des réseaux de Petri récursifs temporels constitue une excellente approche pour l'analyse temporelle des documents multimédias (en particulier, les documents SMIL).

6.2 Brève Description du Langage SMIL

SMIL est un langage de balisage basé sur XML (eXtended Markup Language). Sa syntaxe et sa présentation sont similaires à celles du langage HTML. Les objets multimédias manipulés par SMIL peuvent être :

1. Un objet de base tel qu'une vidéo, une image, une audio, un texte ou un lien (un lien peut correspondre à l'élément $\langle a \rangle$ ou l'élément $\langle area \rangle$). Chaque objet de base Obj est caractérisé par son instant de début (noté par $begin(Obj)$), son instant de fin (noté par $end(Obj)$) et sa durée (noté par $dur(Obj)$). Les attributs $begin(Obj)$ et $end(Obj)$ peuvent correspondre à des valeurs ou des évènements de synchronisation (par exemple, $begin(E) = end(C)$ signifiant que la fin de l'objet C implique le début de l'objet E).

2. Un objet composite (i.e. défini avec des opérateurs temporels). Les principaux opérateurs de synchronisation sont l'opérateur $\langle par \rangle$ qui permet de faire jouer plusieurs éléments simultanément et l'opérateur $\langle seq \rangle$ qui fait jouer un élément après l'autre. Ces deux opérateurs ont, en plus des attributs définis pour les objets de base, des attributs spécifiques. En l'occurrence, l'attribut $endsync$ de l'opérateur $\langle par \rangle$ permet de spécifier sa fin. Par exemple, si $endsync = "first"$, la terminaison de $\langle par \rangle$ a lieu dès que le premier élément de l'opérateur $\langle par \rangle$ termine. Par contre, quand $endsync = "ID"$ alors $\langle par \rangle$ termine aussitôt que l'élément ID termine. Par ailleurs, les instants de début et de fin d'un objet multimédia lié à un opérateur temporel donné sont relatifs à ce dernier. En d'autres termes, un opérateur temporel est une référence de temps pour les objets qu'il comporte et il est considéré comme leur parent.

Il est à noter que les spécifications temporelles d'un document SMIL sont données dans la partie $body$ du document. Par ailleurs, il existe un opérateur, dédié au début de la spécification temporelle, dit $body$. Il a un comportement similaire à celui de l'opérateur $\langle seq \rangle$. De plus, un auteur peut définir des relations de synchronisation entre des objets ayant des références de temps différentes. On parle alors de relations temporelles "extra parent". Aussi, un auteur peut définir des contraintes temporelles de manière incrémentale : il peut rajouter des contraintes ou en supprimer d'autres. Il peut aussi modifier les attributs des objets, ect.. Ces manipulations peuvent rendre le document incohérent.
Nous allons montrer, à travers deux exemples, comment se font la traduction et l'analyse d'un document multimédia en utilisant les réseaux de Petri récursifs temporels.

6.3 Modélisation d'un Document Multimédia Cohérent

Supposons qu' un auteur souhaite présenter trois médias appelés respectivement img_1 (une image), aud (une audio) et img_2 (une seconde image). Les durées de présentation de ces médias sont définies respectivement par les valeurs suivantes : 5s, 20s et 35s. L'auteur exprime en outre les contraintes de synchronisation globales suivantes :

1. Inclure, dans cet ordre, img_1 et aud sous le contrôle d'un même opérateur $\langle seq \rangle$. De plus, 3 secondes après la fin de img_1, aud commence.
2. Inclure cette présentation séquentielle $\langle seq \rangle$ et img_2 sous le contrôle d'un même opérateur $\langle par \rangle$. Le début de $\langle seq \rangle$ est décalé de 1 seconde par rapport à celui de img_2.
3. Quand la séquence $\langle seq \rangle$ termine, deux secondes après, la présentation de img_2 est arrêtée.

Le document spécifié peut être décrit en SMIL de la manière suivante :

$\langle par\ id =' par' \rangle$
$\langle seq\ id =' seq'\ begin = 1s \rangle$
$\langle img\ id =' img_1'\ dur =' 5s' \rangle$
$\langle audio\ id =' aud'\ begin =' 3s'\ dur =' 20s' / \rangle$
$\langle / seq \rangle$
$\langle img\ id =' img_2'\ dur =' 35s'\ end = end(seq) + 2 / \rangle$
$\langle / par \rangle$

Le modèle décrit en termes de réseaux de Petri récursifs temporels associé à cet exemple est donné dans la figure 6.1. Dans ce qui suit, nous décrivons ce modèle.

6.3.1 Description du Modèle

Transitions abstraites du modèle :

1. ts_{smil} permet de modéliser le document SMIL.
2. ts_{par} permet de créer un processus chargé de simuler l'opérateur $\langle par \rangle$,
3. ts_{seq} permet de créer un processus chargé de simuler l'opérateur $\langle seq \rangle$,
4. ts_{img_2} permet de créer un processus chargé de la présentation de l'image img_2,
5. ts_{img_1} permet de créer un processus chargé de la présentation de l'image img_1,
6. ts_{aud} permet de créer un processus chargé de la présentation de l'audio aud.

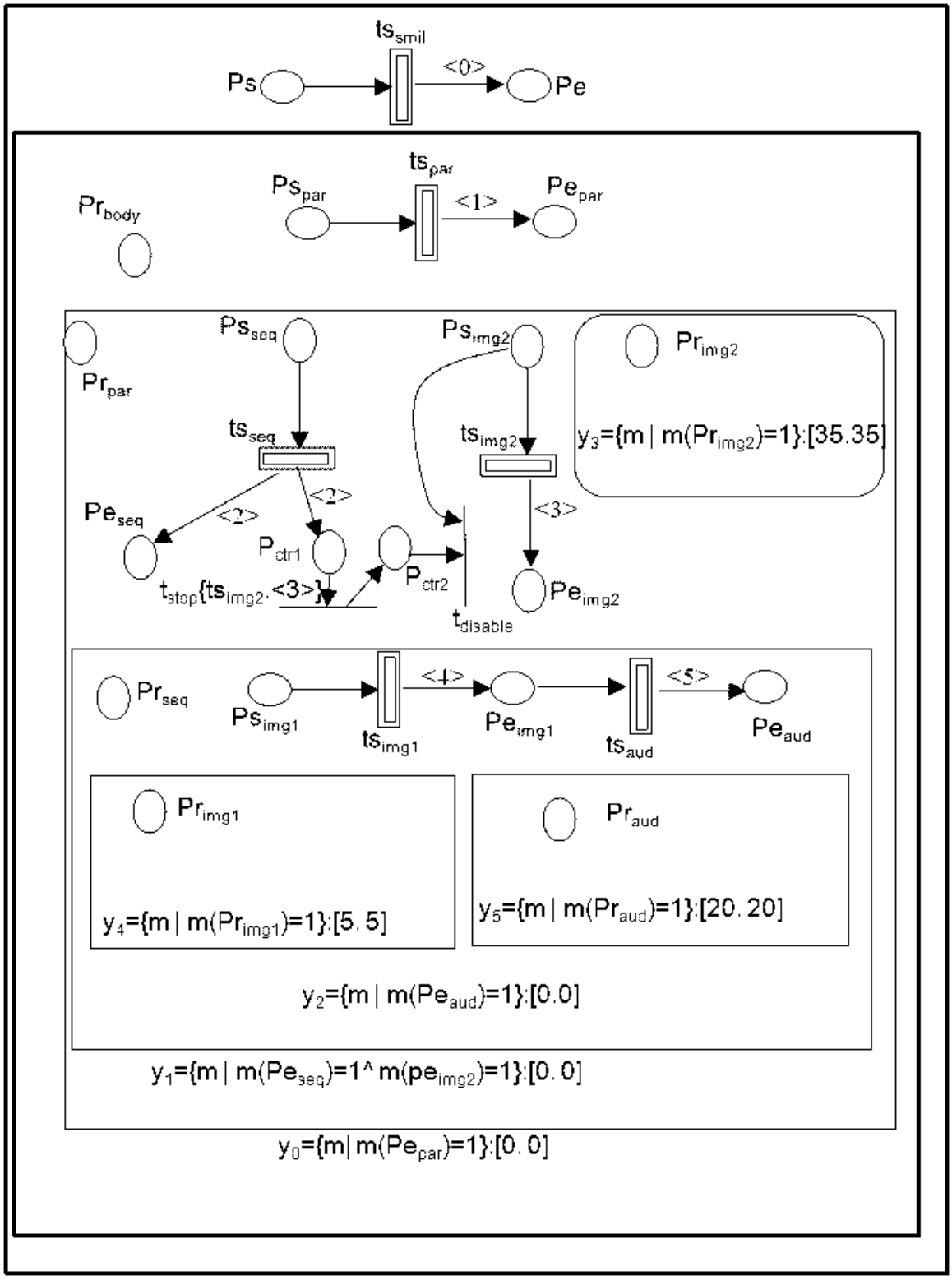

FIG. 6.1 – Modélisation d'un document multimédia cohérent

Les places du modèle :

1. Ps et Pe correspondent respectivement à la balise d'entrée et la balise de sortie du document multimédia,
2. Les places Pr_{img_2}, Pr_{img_1} et Pr_{aud}, appelés places régions, sont dédiées respectivement aux objets img_2, img_1 et aud. Une marque dans un telle place signifie que l'objet associé est en cours d'exécution. ‘
3. La place Pr_{body} est une place région associé à la présentation SMIL . Une marque dans une telle place signifie que le document SMIL est en cours.
4. La place Pr_{par} (resp. Pr_{seq}) est une place région associé à l'opérateur par (resp. seq). Une marque dans une telle place signifie que la présentation parallèle (resp. séquentielle) est en cours.
5. Les places Ps_{par} et Pe_{par} (resp. Ps_{seq} et Pe_{seq}) sont les places en entrée et en sortie de la transition t_{par} (resp. t_{seq}). Une marque dans la place Ps_{par} (resp. Ps_{seq}) est à l'origine de l'activation de la présentation parallèle (resp. séquentielle), alors qu'une marque dans Pe_{par} (resp. Pe_{seq}) signifie que la présentation parallèle (resp. Séquentielle) s'est achevée. La même explication peut être donnée pour les place Ps_{img_1}, Pe_{img_1} et Pe_{aud}. Il est à noter que Pe_{img_1} est à l'origine de l'activation du média aud (ceci explique l'absence de la place Ps_{aud}).
6. La place P_{ctr1} (resp. P_{ctr2}) est une place de contrôle qui permet de sensibiliser la transition élémentaire t_{stop} (resp. $t_{disable}$) que nous décrivons ultérieurement.

Marquages de départ des transitions abstraites du modèle :

Le marquage de départ d'une transition abstraite t est constitué d'une marque dans la place région (Pr_X, avec $X \in \{body, par, seq, img_1, img_2, aud\}$) de l'objet multimédia ou l'opérateur temporel modélisé par t et éventuellement de marque(s) dans les (la) place(s) Ps_X, avec $X \in \{par, seq, img_1, img_2, aud\}$ des transitions abstraites qui seront activées suite au tir de la transition abstraite t. Par exemple, le marquage de départ de la transition ts_{par} est (Pr_{par}, Ps_{seq}, Ps_{img_2}) [1] alors que celui de ts_{img_1} est (Pr_{img_1}). De même, le marquage de départ de la transition t_{seq} est (Pr_{seq}, Ps_{img_1}). Rappelons que le tir de t_{seq} permet d'activer en premier lieu t_{img_1} (i.e. le premier élément de l'opérateur t_{seq}).

Les intervalles temporels attachés aux transitions abstraites du modèle :

1. L'intervalle temporel $[0, 0]$ est attaché à la transition ts_{smil} signifiant qu'elle déclenche le début de la présentation du document multimédia,
2. De même [0 , 0] est attaché à la transition ts_{par} signifiant que la présentation parallèle doit commencer aussitôt que la présentation du document multimédia commence,

[1] En effet, $\langle seq \rangle$ et img_2 sont les éléments de $\langle par \rangle$ et les transitions abstraites t_{seq} et ts_{img_2} seront activées suite au tir de ts_{par}

3. $[0, 0]$ (resp. $[1, 1]$) est attaché à la transition ts_{img2} (resp. ts_{seq}) signifiant que la présentation img_2 commence immédiatement après le début de la présentation parallèle (resp. 1 seconde après le début de la présentation parallèle),
4. $[0, 0]$ est attaché à la transition ts_{img_1} signifiant que la présentation de img_1 débute immédiatement après le début de la présentation séquentielle,
5. $[3, 3]$ est attaché à la transition ts_{aud} signifiant que la présentation de aud débute 3 secondes après la fin de la présentation de img_1.

Transitions élémentaires du modèle et leurs intervalles temporels respectifs :

Le modèle comporte deux transitions élémentaires de contrôle :

1. t_{stop} est sensibilisée aussitôt que la présentation séquentielle se termine, son rôle est de préempter, 2 secondes après sa sensibilisation, t_{img_2} si celle-ci est en cours d'exécution. Ainsi l'intervalle $[2, 2]$ est associé à t_{stop}.

2. $t_{disable}$ est sensibilisée aussitôt que t_{stop} est tirée. Son rôle est de voler la marque de Ps_{img_2}, si elle existe, empêchant dans ce cas de figure le tir de ts_{img_2}. En effet, si 2 secondes après la fin de la présentation séquentielle, la présentation de img_2 n'a pas encore commencé, elle ne débutera jamais. Ainsi, l'intervalle $[0, 0]$ est associé à $t_{disable}$.

Ensemble de marquages finaux :

La famille indexée des ensembles de marquages finaux correspond à $\Upsilon_i, i \in 0..5$, avec

1. Υ_0 signifie qu'une fois la place Pe_{par} est marquée au sein du processus créé par ts_{smil}, il doit *immédiatement* terminer ($[0, 0]$ est attaché à la coupure τ_0),
2. Υ_1 signifie qu'une fois les places Pe_{seq} et Pe_{img_2} deviennent marquées au sein du processus créé par ts_{par}, il doit *immédiatement* terminer ($[0, 0]$ est attaché à la coupure τ_1),
3. Υ_2 signifie qu'une fois la place Pe_{aud} devient marquée au sein du processus créé par ts_{seq}, il doit *immédiatement* terminer ($[0, 0]$ est attaché à la coupure τ_2),
4. Υ_3 signifie qu'une fois la place Pr_{img_2} est marquée au sein du processus créé par ts_{img_2}, il doit terminer dans 35 secondes ($[35, 35]$ est attaché à la coupure τ_3),
5. Υ_4 signifie qu'une fois la place Pr_{img_1} est marquée au sein du processus créé par ts_{img_1}, il doit terminer dans 5 secondes ($[5, 5]$ est attaché à la coupure τ_4),
6. Υ_5 signifie qu'une fois la place Pr_{aud} est marquée au sein du processus créé par ts_{aud}, il doit terminer dans 20 secondes ($[20, 20]$ est attaché à la coupure τ_5),

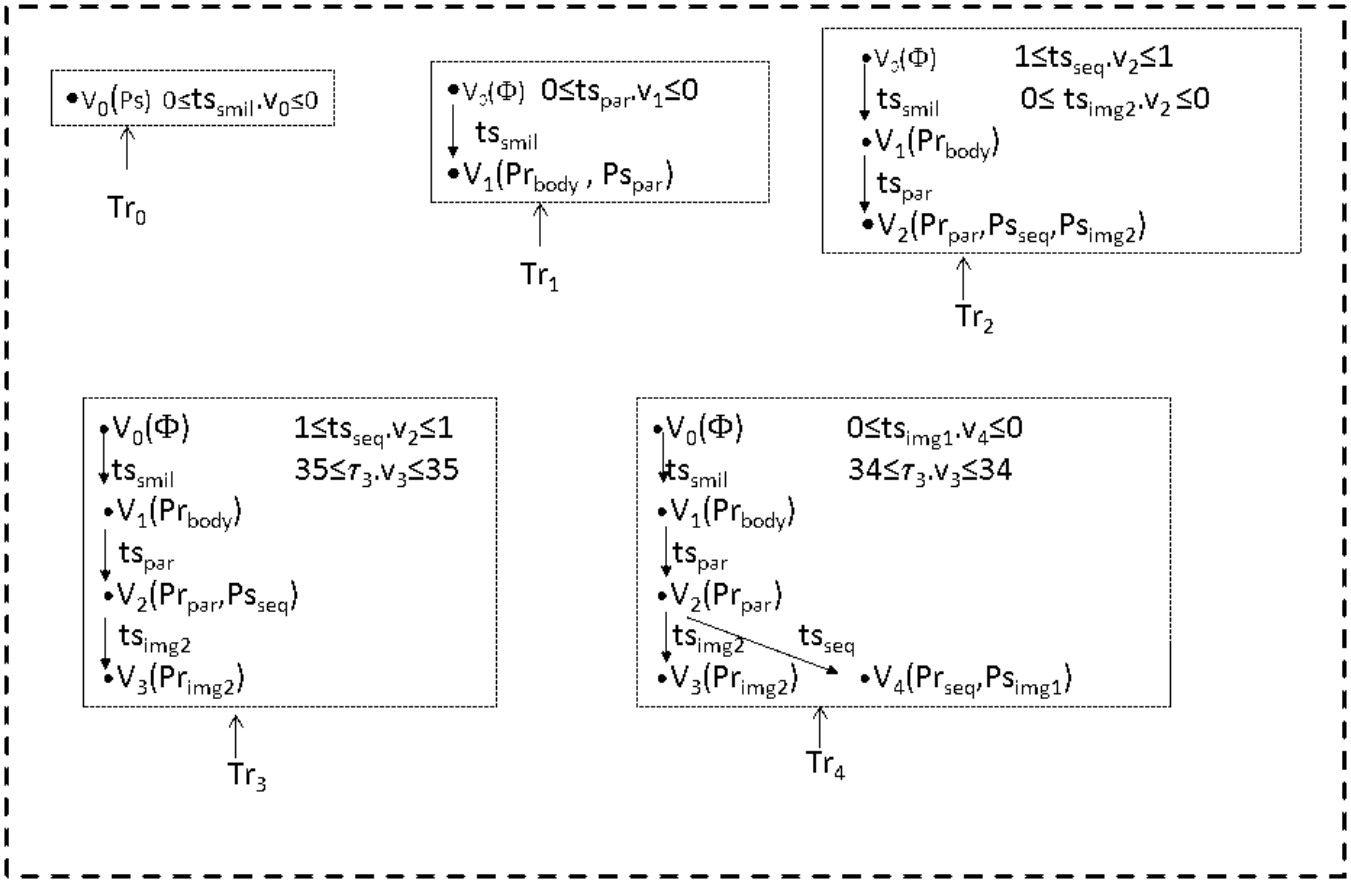

FIG. 6.2 – Classes d'états (première partie)

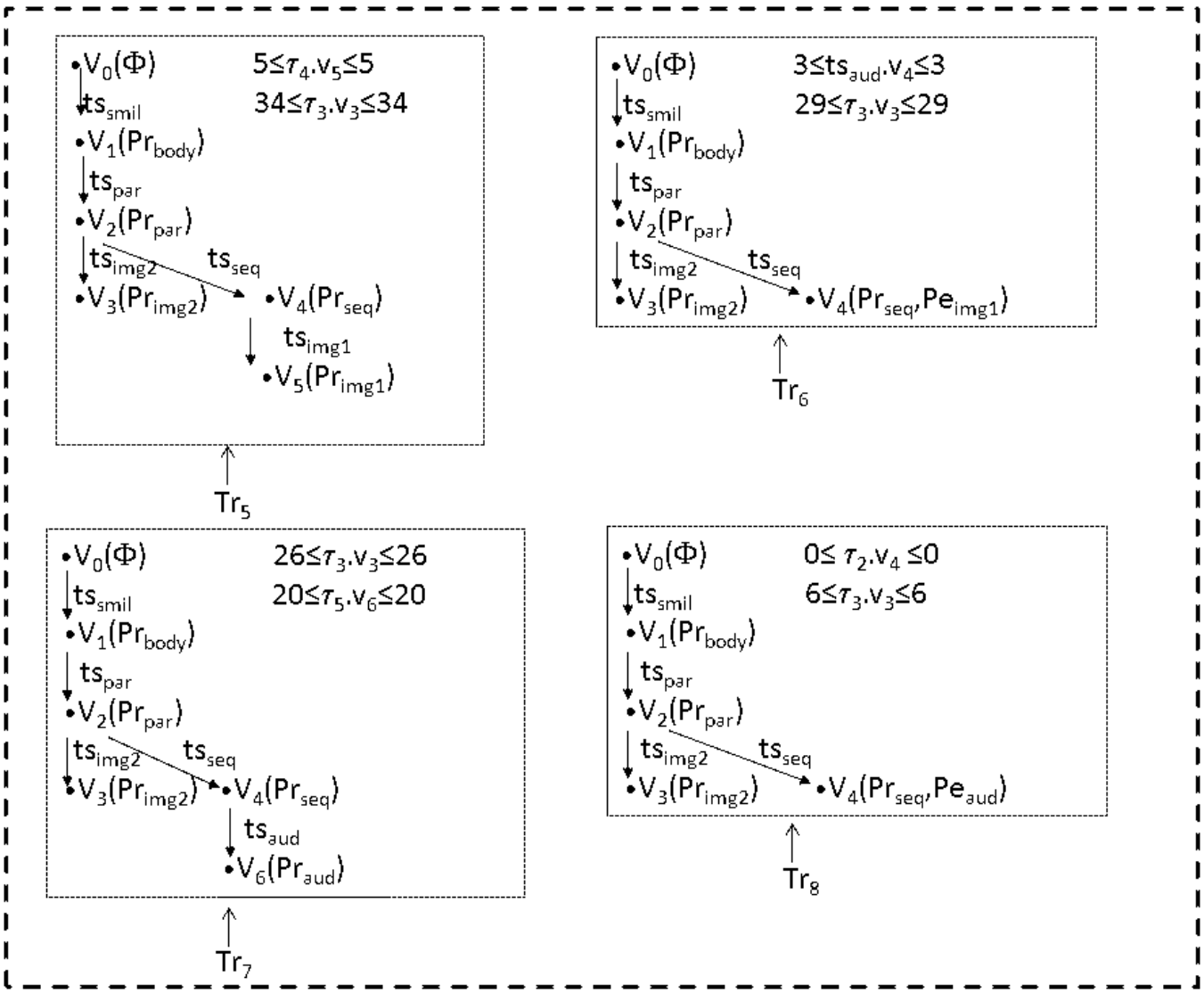

FIG. 6.3 – Classes d'états (deuxième partie)

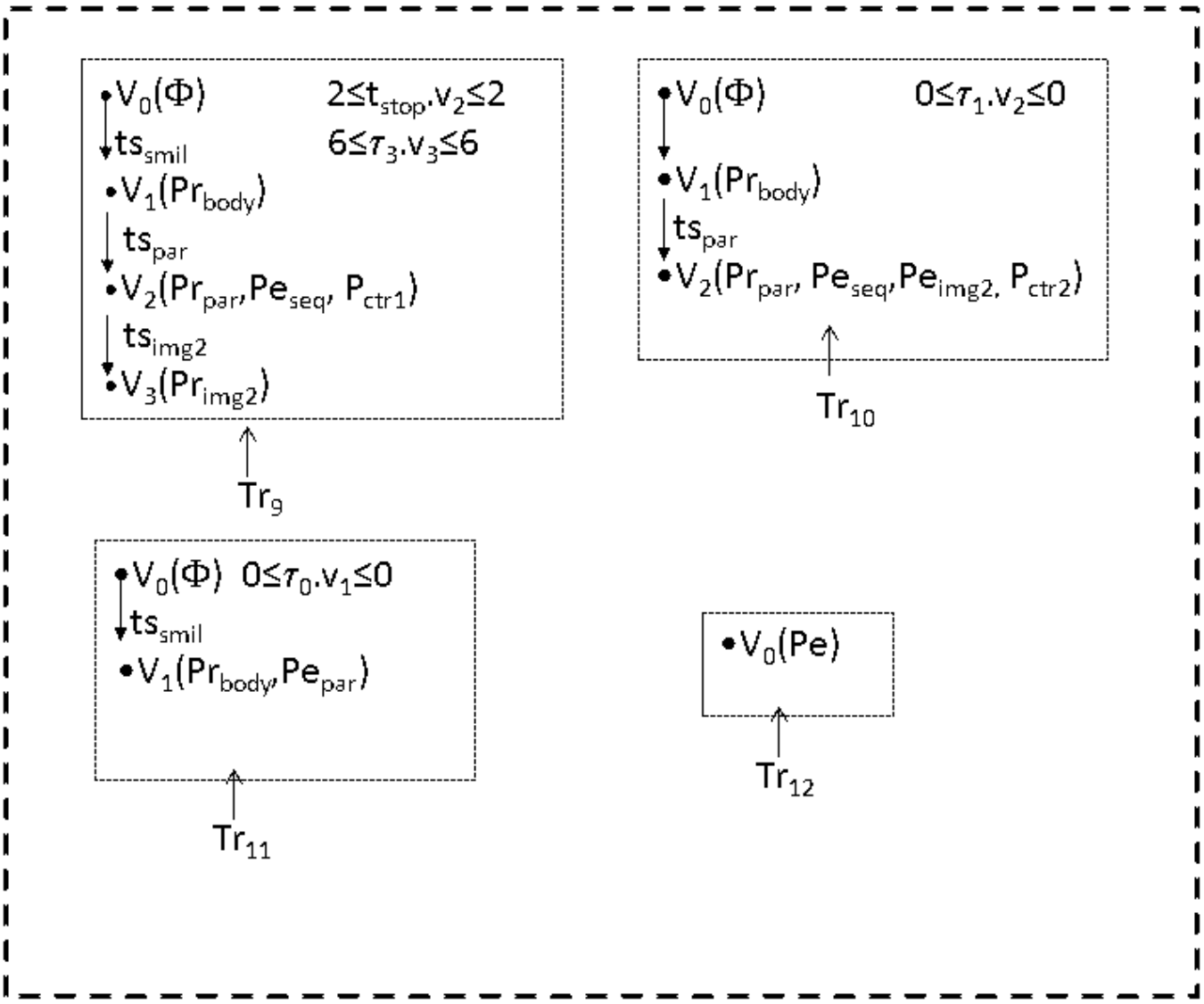

FIG. 6.4 – Classes d'états (troisième partie)

6.3.2 Analyse du Document Multimédia

Dans l'approche que nous proposons dans [47], un document multimédia est cohérent si il existe, dans le modèle RdPRT qui lui est associé, une séquence de tir qui mène de la classe initiale (celle-ci est marquée par un marquage étendu réduit à un seul noeud marqué par (Ps)) à une autre classe marquée par un marquage étendu tel que la racine contienne au moins une marque dans (Pe) [47]. Pour l'exemple que nous venons de traiter, une telle séquence exite (voir figures 6.2, 6.3 et 6.4) et correspond à :

$Tr_0 \xrightarrow{ts_{smil},v_0} Tr_1 \xrightarrow{ts_{par},v_1} Tr_2 \xrightarrow{ts_{img_2},v_2} Tr_3 \xrightarrow{ts_{seq},v_2} Tr_4 \xrightarrow{ts_{img_1},v_4} Tr_5 \xrightarrow{\tau_4,v_5} Tr_6 \xrightarrow{ts_{aud},v_4} Tr_7 \xrightarrow{\tau_5,v_6}$

$Tr_8 \xrightarrow{\tau_2,v_4} Tr_9 \xrightarrow{t_{stop},v_2} Tr_{10} \xrightarrow{\tau_1,v_2} Tr_{11} \xrightarrow{\tau_0,v_1} Tr_{12}.$

6.4 Exemple d'un Document Multimédia Incohérent

Dans le paragraphe précédent, nous avons fait l'analyse d'un document multimédia cohérent sur le plan temporel. Ici, nous considérons un deuxième document multimédia incohérent et montrons comment prouver son incohérence temporelle. Nous reprenons l'exemple précédent, nous maintenons la même description sauf que le début de la présentation séquentielle se fait immédiatement après la fin de img_2. De même, le début de img_2 se fait immédiatement après la fin de la présentation séquentielle. Par ailleurs, le fin de img_2 ne dépend plus de celle de la présentation séquentielle. Le document en question est décrit en SMIL comme suit :

$\langle par\ id =' par' \rangle$
$\langle seq\ id =' seq'\ begin = end(img_2) \rangle$
$\langle img\ id =' img_1'\ dur =' 5s' \rangle$
$\langle audio\ id =' aud'\ begin =' 3s'\ dur =' 20s' / \rangle$
$\langle / seq \rangle$
$\langle img\ id =' img_2'\ begin = end(seq)\ dur =' 35s' \rangle$
$\langle / par \rangle$

La figure 6.5 donne le RdPRT associé à ce deuxième document multimédia. Il est similaire au modèle associé au premier scénario, sauf que les deux transitions élémentaires t_{stop} et $t_{disable}$ qui permettent de contrôler la fin de img_2 ont été supprimées (puisque cette contrainte temporelle n'existe pas dans le nouveau scénario). Par ailleurs, deux places de contrôle (P_{ctr1} et P_{ctr2}) sont rajoutées ; elles permettent de mettre en oeuvre les contraintes temporelles définies sur les instants de début de la présentation séquentielle et img_2. P_{ctr1} est marquée aussitôt que la transition abstraite ts_{seq} est fermée ; elle est en entrée de la transition abstraite ts_{img_2}. De même, la place P_{ctr2} est une sortie différée de la transition ts_{img_2} et en entrée de la transition ts_{seq}. Par ailleurs, l'intervalle temporel $[0, 0]$ est attaché à t_{seq} et aussi à ts_{img_2}. Le graphe des classes d'états étendu associé à ce modèle, est décrit par la séquence suivante :

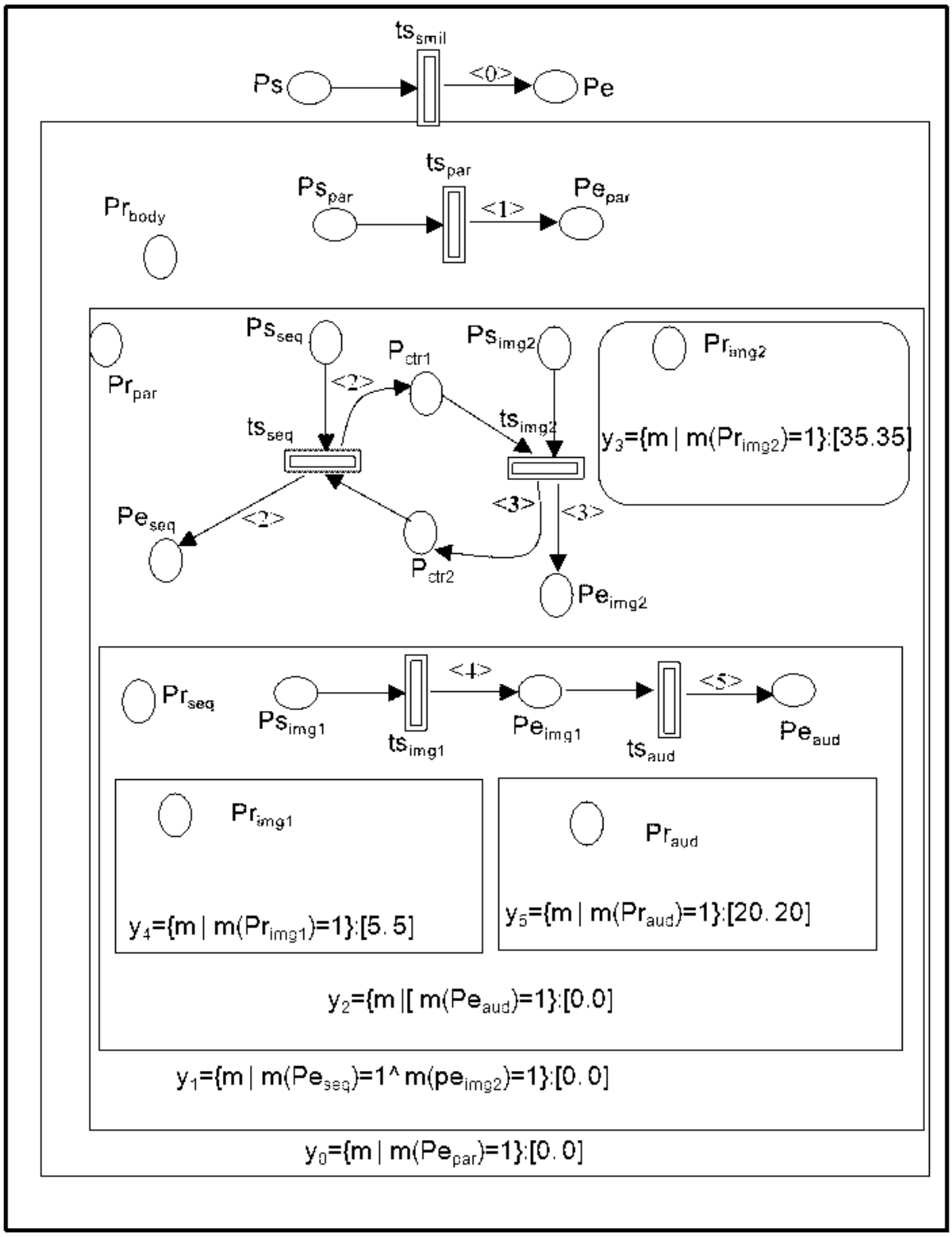

FIG. 6.5 – Modélisation d'un document multimédia incohérent

$Tr_0 \xrightarrow{ts_{smil},v_0} Tr_1 \xrightarrow{ts_{par},v_1} Tr_2$

À partir de Tr_2, aucune transition n'est tirable, il s'agit d'un état mort. Il s'ensuit qu'on ne peut pas atteindre une classe où la place P_e soit marquée au sein du noeud racine du marquage étendu de la classe. Par conséquent, l'exemple que nous avons présenté est incohérent et ne pourra pas être joué.

6.5 Un Document Multimédia avec des Relations de Synchronisation Extra Parent

Les modèles RdPRT que nous nous avons présentés sont *compacts* et *modulaires*. Ils ont des structures hiérarchiques et sont très lisibles. En effet, dans la modélisation de documents multimédias à l'aide des RdPRT que nous avons proposée dans [47][49], chaque objet de base ou composite est décrit par un *module à part*. Les modules décrivant ces objets sont donnés en cascade reproduisant exactement la structure du document multimédia traduit. Maintenant considérons un troisième document SMIL.

$\langle par\ id =' par'_0 \rangle$
$\langle par\ id =' par'_1 \rangle$
$\langle video\ id =' vid'\ dur =' 36s' \rangle$
$\langle audio\ id =' aud'\ begin = begin(img_1) + 2\ dur =' 20s' / \rangle$
$\langle / par \rangle$
$\langle seq\ id =' seq'_0 \rangle$
$\langle img\ id =' img'_1\ dur =' 5s' \rangle$
$\langle img\ id =' img'_2\ begin = end(aud) + 3\ dur =' 25s' / \rangle$
$\langle / seq \rangle$
$\langle / par \rangle$

Dans cet exemple, une première relation de synchronisation est définie entre les objets img_1 et aud et une deuxième relation de synchronisation est définie entre les objets img_2 et aud. Il s'agit de relation de synchronisation "extra parent". En effet, $\langle seq_0 \rangle$ est le parent de img_1 et $\langle par_1 \rangle$ est celui de aud (i.e. img_1 et aud ont des parents différents). De même, aud et img_2 ont des parents différents. Dans cet troisième exemple, nous ne pouvons pas dédier un module à chaque média de base ou composite, comme pour les deux exemples précédents, ceci est du aux relations de synchronisation. En effet, $\langle seq_0 \rangle$ et $\langle par_1 \rangle$ doivent être décrits dans le même module pour mettre en oeuvre les relations de synchronisation entre les médias à synchroniser. Ceci est fait au détriment de la structure hiérarchique et la lisibilité du modèle. En effet, plusieurs relations de synchronisation "extra parent" au sein d'un même document SMIL rendent les modèles RdPRT générés peu lisibles. Dans ce cas de figure, les RdPRT ressemblent aux modèles T.RdPT que nous avons proposé pour les documents SMIL dans [50] [51].

6.6 Conclusion

Dans ce chapitre, nous avons montré les possibilités de modélisation de notre modèle RdPRT. Nous avons montré, à travers deux exemples multimédias, que les modèles RdPRT sont très lisibles, compacts et élégants. Par ailleurs, l'aspect dynamique est pris en charge de manière naturelle grâce à la gestion dynamique des processus. Nous avons expliqué comment se fait le contrôle de la cohérence temporelle d'un document multimédia en exploitant son modèle RdPRT et en construisant son graphe des classes d'états étendu.
Toutefois, la présence de relations de synchronisation "extra parent" entre les objets médias fait perdre la lisibilité de leurs modèles RdPRT. Ceci est du au fait que les processus d'un RdPRT (ou encore d'un RdPR) n'ont pas la faculté de communiquer entre eux. Ceci nous amène à penser à enrichir les RdPRT, plus particulièrement leur modèle sous-jacent les RdPR, par cette faculté. C'est justement l'objet de la deuxième partie de notre travail de thèse.

Chapitre 7

Réseaux de Petri Récursifs avec Places Partagées

7.1 Introduction

La description des systèmes parallèles et complexes en termes de procesus est meilleure qu'une description mono-processus. Cependant, les processus décrivant les sytèmes parallèles ont besoin de communiquer entre eux. Le modèle des réseaux de Petri récursifs, étudié dans le cadre de cette thèse, permet une forme de communication entre les processus *très restreinte*. En effet, quand un processus crée un autre processus fils, ce dernier opère sur un marquage des places complètement *local* et *invisible* aux autres processus. La seule forme d'intéraction possible entre les processus réside dans les relations de naissance et mort de processus.
Dans ce chapitre, nous proposons d'étendre le modèle des réseaux de Petri récursifs afin d'élargir les possibilités de communication entre les processus. Une telle extension a un impact négatif sur les résultats de décidabilité des RdPR. Ainsi, nous avons été amenés à imposer des contraintes sémantiques au modèle étendu.

7.2 Description des Réseaux de Petri Récursifs avec Places Partagées

Le modèle étendu que nous proposons est intitulé "Réseau de Petri Récursif avec places partagées (RdPRp)". C'est une extension des RdPR caractérisée par les deux points suivants :

1. La possibilité, pour un processus donné, de contrôler les naissances de ses processus fils avec un nouveau type de sorties associées aux transitions abstraites, dites *sorties immédiates* ; de telles sorties

sont produites *aussitôt* qu'une transition abstraite est tirée[1].

2. D'autre part, la sémantique opérationnelle de ce modèle étendu permet la manipulation des places *locales* et *partagées*. Contrairement à une place locale, toute modification du marquage d'une place partagée, par un processus donné, est *visible* par l'ensemble des processus existants.

Il est à noter qu'un RdPR est un cas particulier d'un RdPRp. En d'autres termes, un RdPR est un RdPRp manipulant uniquement des places locales et admettant que des sorties différées.

Définition 7.1 (*Réseau de Petri Récursif avec Places Partagées*) Un Réseau de Petri Récursif avec Places Partagées (RdPRp) est défini par un tuple $Rp = \langle R, W_{im}, Co\rangle$ avec :

1. $R = \langle P, T, W^-, W^+, \Omega, \Upsilon\rangle$ est un RdPR tel que défini dans [29][2],
2. W_{im} est la fonction d'incidence avant de $P \times T_{ab}$ dans $\mathbb{N}$ permettant la génération des sorties immédiates d'une transition abstraite,
3. Co est une fonction de P dans $\{P, L\}$ donnant la nature de chaque place (i.e. locale ou partagée).

Notations et Conventions :
1. L'ensemble P est l'union disjointe des ensembles des places locales et partagées qui sont notés respectivement par P_{loc} et P_{par}.
2. Le marquage de départ, les pré-conditions et les post-conditions des transitions abstraites portent uniquement sur les places *locales*. De la même manière, les marquages finaux portent uniquement sur les places *locales*.
3. Par souci de clarté, certaines places sont dupliquées dans les représentations graphiques des modèles.

Comme pour un RdPR, dans un RdPRp, nous avons deux types de marquages : marquages étendus et marquages ordinaires. Un marquage étendu donne l'état d'un RdPRp. Un marquage ordinaire représente un contexte d'exécution d'un processus.

Définition 7.2 (*Marquage étendu*) Un marquage étendu d'un RdPRp $Rp = \langle\langle P, T, W^-, W^+, \Omega, \Upsilon\rangle, W_{im}, Co\rangle$ est un arbre $Tr_p = \langle Tr, M_{par}\rangle$ où :

1. $Tr = \langle V, E, A, M\rangle$ est un marquage étendu défini de la même manière que pour un RdPR,
2. M_{par} est une application de P_{par} dans $\mathbb{N}$ associant un marquage à chaque place partagée.

Les conditions de sensibilisation des transitions et coupures sont similaires à celles des RdPR sauf que, pour les transitions élémentaires, la condition suivante doit être vérifiée : $\forall p \in P_{par}$ et $\forall t \in T_{el}$, $M_{par}(p) \geq W^-(p,t)$ [3].

Définition 7.3 (*Tir d'une coupure*) Le tir d'une une coupure τ à partir d'un noeud v d'un marquage étendu $Tr_p = \langle\langle V, M, E, A\rangle, M_{par}\rangle$ conduit au marquage étendu $Tr'_p = \langle\langle V', M', E', A'\rangle, M'_{par}\rangle$ avec :
a. V', M', E', A' définis comme pour un RdPR,

[1] Les autres sorties d'une transition abstraite sont appelées différées.
[2] Cette version des RdPR ne prend pas en charge les sorties indicées des transitions abstraites et le concept de préemption.
[3] Seules les transitions élémentaires manipulent les places partagées.

b. $M_{par} = M'_{par}$.

Définition 7.4 (*Tir d'une transition abstraite*) Le tir d'une transition abstraite t_f à partir d'un noeud v d'un marquage étendu $Tr_p = \langle\langle V, M, E, A\rangle, M_{par}\rangle$ conduit au marquage étendu $Tr'_p = \langle\langle V', M', E', A'\rangle, M'_{par}\rangle$ avec :
a. V', E', A' définis comme pour un RdPR,
b. $\forall p \in P_{loc}, M'(v)(p) = M(v)(p) - W^-(t_f, p) + W_{im}(t_f, p)$,
b. $M_{par} = M'_{par}$.

Définition 7.5 (*Tir d'une transition élémentaire*) Le tir d'une transition élémentaire t_f à partir d'un noeud v d'un marquage étendu $Tr_p = \langle\langle V, M, E, A\rangle, M_{par}\rangle$ conduit au marquage étendu $Tr'_p = \langle\langle V', M', E', A'\rangle, M'_{par}\rangle$ avec :
a. V', M', E', A' définis comme pour un RdPR
b. $\forall p \in P_{par}$, $M_{par}(p) \geq W^-(p, t_f)$ et $\forall p \in P_{par}$, $M'_{par}(p) = M_{par}(p) - W^-(p, t_f) + W^+(p, t_f)$.

Exemple 7.1. Le réseau de la figure 7.1 représente un RdPRp . Nous avons représenté une place locale par un cercle et une place partagée par un double cercle. Les marquages étendus accessibles d'un RdPRp sont calculés de la même manière que pour un RdPR, sauf qu'un marquage étendu est enrichi par le marquage des places partagées (que l'on spécifie au niveau de la racine du marquage étendu). Une séquence de tir du RdPRp Rp_1 est présentée dans la figure 7.2. Aucune place partagée n'est marquée dans Tr_0. Dans Tr_1, Pg_1 est marquée, alors que dans Tr_5, Pg_1 contient deux marques. Cette séquence est infinie et la place Pg_1 n'est pas bornée. Il est à remarquer que le tir de la transition t_4 peut impliquer une infinité de tirs de la même transition. Une telle transition est dite *récursive*.

7.3 Décidabilité du Problème d'Accessibilité

Le problème d' accessibilité consiste à déterminer si un état donné est accessible à partir d'un autre état. Ce problème a été prouvé *décidable* pour les RdP (voir [48] [43][44]) et aussi pour les RdPR (voir [32][29]).
Le problème d'accessibilité pour un RdPR se ramène à un problème d'*accessibilité classique* en utilisant un RdP ordinaire déduit à partir du RdPR. Rappelons qu'un tel réseau est obtenu en subsituant chaque transition abstraite par une transition élémentaire. Cette dernière a les mêmes pré-conditions que la transition abstraite. De plus, si la transition abstraite est "fermable", ses post-conditions sont associées à la transition élémentaire. Ce processus de substitutions permet de faire abstraction des comportements locaux des transitions abstraites. La procédure d'accessibilité que nous proposons pour les RdPRp est basée sur le même principe, sauf qu'une transition abstraite sera substituée par un sous-réseau ordinaire, que l'on appelera *réseau d'interface* de la transition. Un tel réseau modélise la partie significative du *comportement* du sous-système, associé à la transition abstraite, par rapport au marquage des places *partagées*. Par ailleurs, il fait *abstraction au comportement local* de la transition abstraite.

Définition 7.6 (*Transition d'interface*) Une transition d'interface t est une transition élémentaire telle

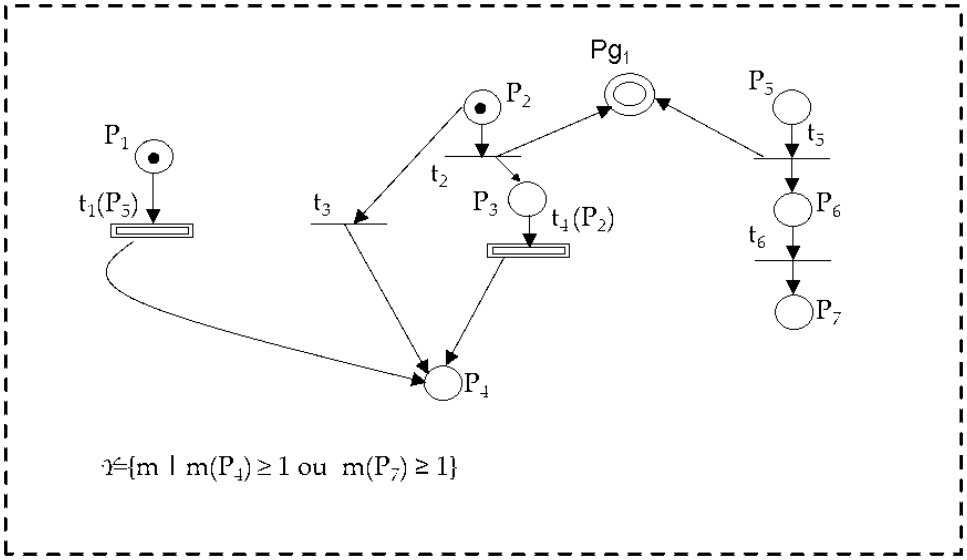

FIG. 7.1 – Un RdPRp (Rp_1) avec une "transition récursive"

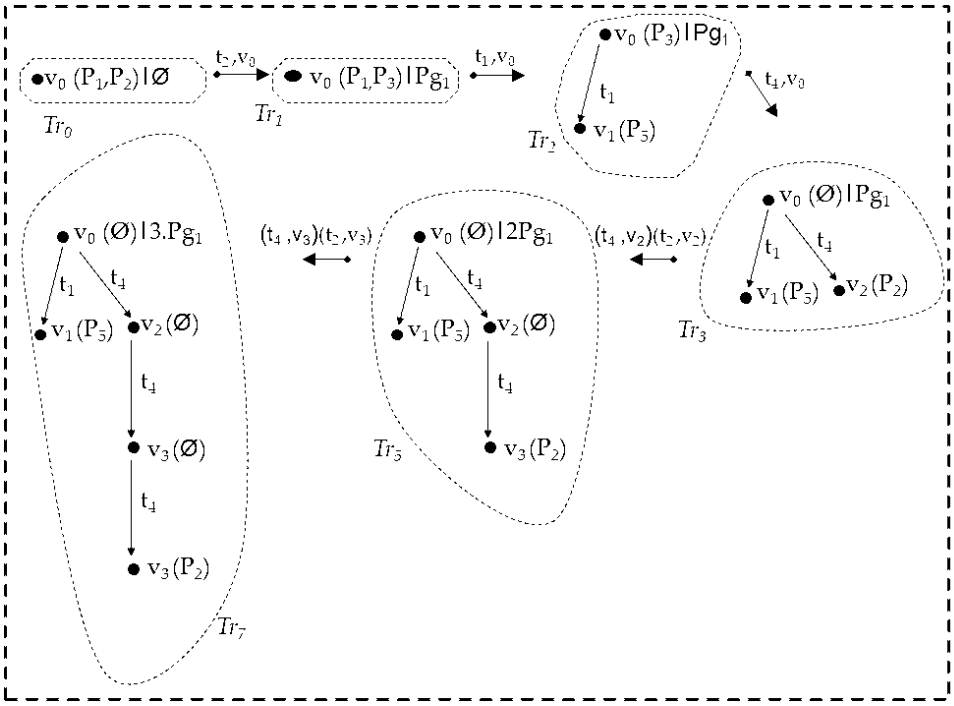

FIG. 7.2 – Une séquence de tir de Rp_1

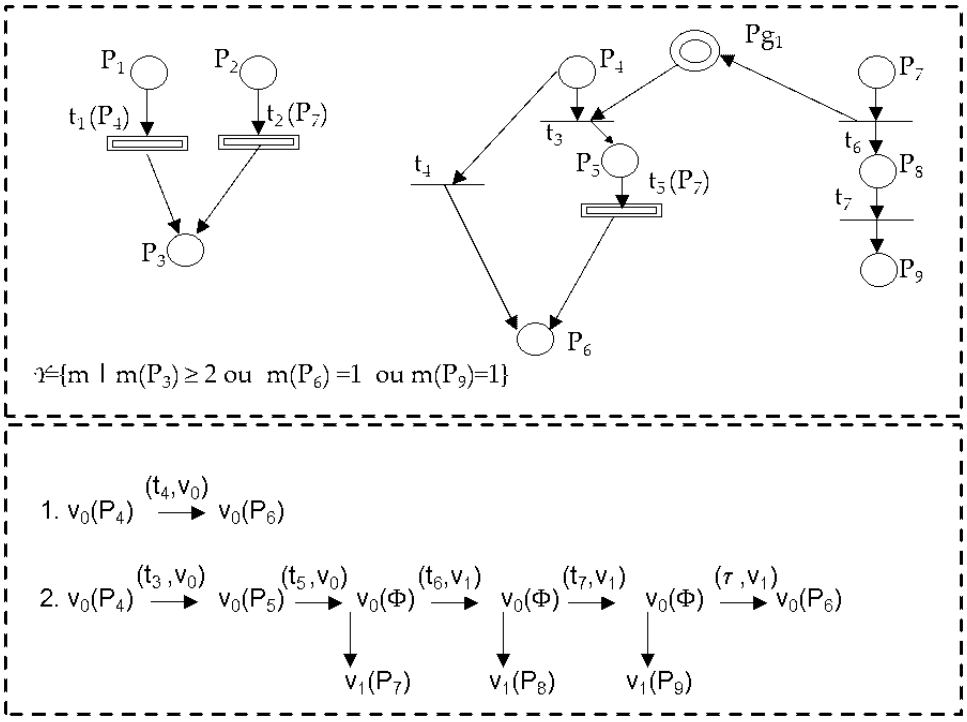

FIG. 7.3 – Un deuxième RdPRp (Rp_2) et deux séquences de tir de $(Rp_2 \setminus \{Pg_1\})$

que $\exists p \in P_{par}$, avec $W^-(p,t) \geq 1$ ou $W^+(p,t) \geq 1$. L'ensemble des transitions d'interface est noté T_{in}.

Définition 7.7 (*Transition d'interface franchissable en isolation par une transition abstraite*) Une transition d'interface est franchissable en *isolation* par une transition abstraite t dans le réseau récursif avec places partagées Rp si elle est franchissable dans le réseau récursif $((Rp \setminus \{P_{par}\}), \Omega(t))$.

En d'autres termes, le franchissement d'une transition d'interface franchissable en isolation par une transition abstraite dépend uniquement des marquages des places partagées.

Définition 7.8 (*Transition abstraite d'ordre 0*) Soit Rp un RdPRp, une transition abstraite t est d'ordre 0 si $\forall \omega$ une séquence de tir de $(Rp, \Omega(t))$ alors ω ne contient aucun tir d'une transition abstraite.

Exemple 7.2. Contrairement à la transition t_6, la transition t_3 du RdPRp Rp_2 de la figure 7.3 est franchissable en isolation par la transition abstraite t_1. Par ailleurs, les transition t_2 et t_5 du RdPRp Rp_2 sont d'ordre 0, alors que t_1 ne l'est pas.

7.3.1 Langage en Isolation des Transitions Abstraites

On dénote par $L(Rp \setminus \{P_{par}\}, \Omega(t))$ l'ensemble des séquences de tir du réseau récursif marqué $(Rp \setminus \{P_{par}\}, \Omega(t))$ (i.e. définies dans $(T \cup \tau)^*$). Chaque séquence de ce langage mène à partir d'un

marquage étendu, réduit à un *seul* noeud et marqué par le marquage ordinaire $\Omega(t)$, à un autre marquage étendu limité également à un *seul noeud* marqué soit par un marquage *final*, soit par un marquage *mort* (voir la définition d'un langage d'une transition abstraite dans [31]). Ce langage correspond au *langage en isolation* de la transition abstraite t. D'autre part, les *comportements observables* de la transition abstraite que nous allons considérer sont définies via une fonction d'étiquetage h. h permet *d'observer uniquement les transitions d'interface* et est donc définie de $(T \cup \{\tau\})$ dans $(T_{in} \cup \{\varepsilon\})$. Comme d'habitude, h est étendue aux séquences et aux langages de séquences. On note par $\Delta(t)$ l'ensemble $h(L(Rp \setminus \{P_{par}\}, \Omega(t)))$; chaque élément de $\Delta(t)$ est appelé *séquence observée* de t.

Définition 7.9 (*Séquence fermante*) Soit $\sigma \in \Delta(t)$, σ est une séquence observée fermante de la transition abstraite t si et seulement si $\exists\omega$ tel que :
- $\Omega(t) \xrightarrow{\omega} \perp$ est une séquence de tir de $(Rp \setminus \{P_{par}, \Omega(t)\})$,
- $\sigma = \omega_{|T_{in}}$,

Exemple 7.3. Considérons le réseau récursif $(Rp_2 \setminus \{Pg_1\})$ de la figure 7.3. En partant d'un noeud v_0 marqué par $\Omega(t_1)$, il est possible de construire les deux séquences de tir présentées dans la partie basse de la figure 7.3. Ainsi , nous avons $L(t_1) = \{t_4, t_3t_5t_6t_7\tau\}$ et $\Delta(t_1) = \{\varepsilon, t_3t_6\}$,
De manière similaire, nous avons $L(t_2) = L(t_5) = \{t_6t_7\}$ et $\Delta(t_2) = \Delta(t_5) = \{t_6\}$.
Ici, toutes les séquences de $\Delta(t_1)$ sont des séquences observées fermantes de t_1 (ceci est aussi valable pour les éléments de $\Delta(t_2)$ et $\Delta(t_5)$).

Remarque : Le langage du réseau d'interface d'une transition abstraite t correspond à $\Delta(t)$.

7.3.2 Construction des Ensembles $Delta$ des Transitions Abstraites

Notre objectif est de construire, pour chaque transition abstraite t d'un RdPRp Rp, son langage $\Delta(t)$. De plus, nous déterminons les éléments de $\Delta(t)$ qui sont des séquences observées fermantes de t. Pour ce faire, nous retirons les transitions abstraites du RdPRp Rp ; nous obtenons un réseau ordinaire que l'on note par Rp^0_{ord}. A partir du réseau $(Rp^0_{ord} \setminus \{P_{par}\})$, il est possible de calculer les langages $Delta$ associés aux transitions abstraites d'ordre 0.

En effet, chaque séquence observée d'une transition abstraite d'ordre 0 amet une séquence sous-jacente ne comportant aucun tir d'une transition abstraite. En revanche, les langages $Delta$ des autres transitions sont partiellement calculés (à ce niveau, ils contiennent uniquement les projections, par rapport aux transitions d'interface, de séquences de tir ne faisant intervenir aucun tir d'une transition abstraite). Par la suite, nous substituons chaque transition abstraite t par un sous réseau de Petri *ordinaire* dont le langage correspond à $\Delta(t)$. Initialement, seuls les transitions d'interface d'ordre 0 sont substituées par leurs réseaux d'interface respectifs ; le réseau ordinaire obtenu après les substitutions des transitions abstraites est Rp^1_{ord}. Si Rp contient au moins une transition abstraite qui n'est pas d'ordre 0, nous poursuivons le calcul des ensembles $Delta$ en utilisant le réseau $Rp^1_{ord} \setminus \{P_{par}\}$. Dans ce cas, les nouvelles séquences générées dans

les ensembles $Delta$ permettent de construire le réseau Rp^2_{ord}. Ainsi nous procèdons par étapes, à l'étape i le réseau ordinaire obtenu est noté Rp^i_{ord}.

D'une manière générale, les ensembles $Delta$ calculés, à l'étape i, permettent d'enrichir le réseau ordinaire Rp^i_{ord} par de nouveaux comportements *des transitions abstraites par rapport au marquage des places partagées* (i.e. il s'agit d'une construction progressive des réseaux d'interface associés aux transitions abstraites). Quand les ensembles $Delta$ de toutes les transitions abstraites *se stabilisent*, le processus de calcul de ces ensembles est terminé.

Nous montrons comment construire progressivement, à partir des différentes versions de l'ensemble $\Delta(t)$ d'une transition abstraite t, son réseau d'interface. Supposons qu'à l'étape i (avec $i > 0$), $new(t)$ est l'ensemble des *nouvelles* séquences observées [4] de la transition abstraite t déduites à partir de Rp^{i-1}_{ord}. Chaque $\sigma \in New(t)$ permet de simuler un nouveau comportement de la transition abstraite t, par rapport aux places partagées, de la manière suivante :
(i) σ est une séquence vide. Dans ce cas, la transition t sera simulée par une transition élémentaire t'. Les entrées de t sont reliées à t'. De plus, les sorties de t sont reliées à t' si σ est une séquence observée fermante de la transition t.
(ii) σ contient une seule transition d'interface. Dans ce cas, la transition sera simulée par une transition élémentaire t' comme précédemment. De plus, t' a les mêmes pré-conditions et post-conditions en termes de places partagées que la transition de σ.
(iii) σ contient m (où m est la longueur de σ) transitions. Cette fois-ci la transition sera simulée par m transitions élémentaires (t'_1, ...,t'_m) construites de manière séquentielle reproduisant *l'impact* de la transition t sur le marquage des places partagées du réseau. Les entrées de t sont reliées à t'_1. En revanche, les sorties de t sont reliées à t'_m si σ est une séquence observée fermante de la transition t. De plus, chaque t'_i a les mêmes pré-conditions et post-conditions en terme de places partagées que la ième transition d'interface de σ.

L'algorithme **4** permet de construire les ensembles $Delta$ des transitions abstraites d'un RdPRp Rp. Nous utilisons la fonction $Copy$ qui permet de générer une nouvelle copie d'un RdP donné. Etant donné un RdP, la fonction $add.trans$ (resp. $add.interface$) permet de rajouter une transition élémentaire au RdP (resp. à l'ensemble des transitions d'interface). De même, la fonction $add.place$ rajoute une place locale fraiche à un RdP donné. Par ailleurs, Les fonctions $affect.input.locale$, $affect.outputd.locale$ et $affect.outputi.locale$ affecte respectivement les pré-conditions, les post-conditions différées et les post-conditions immédiates (en terme de places locales) d'une transition abstraite à une transition élémentaire rajoutée. La fonction $affect.par$, pour sa part, affecte les pré-conditions et les post-conditions en terme de places partagées d'une transition d'interface à une transition élémentaire rajoutée. La fonction $identifier.place$ permet de générer un identificateur frais d'une place locale. La fonction $identifier.trans$, appliquée à une transition abstraite ou d'interface, permet de générer un identificateur frais que l'on associe à une transition élémentaire rajoutée ; l'identificateur porte une "mention" de la transition abstraite ou d'interface. La fonction $Delta$ donne le langage d'un réseau ordinaire marqué par le marquage de départ

[4] Par rapport à l'étape $i - 1$.

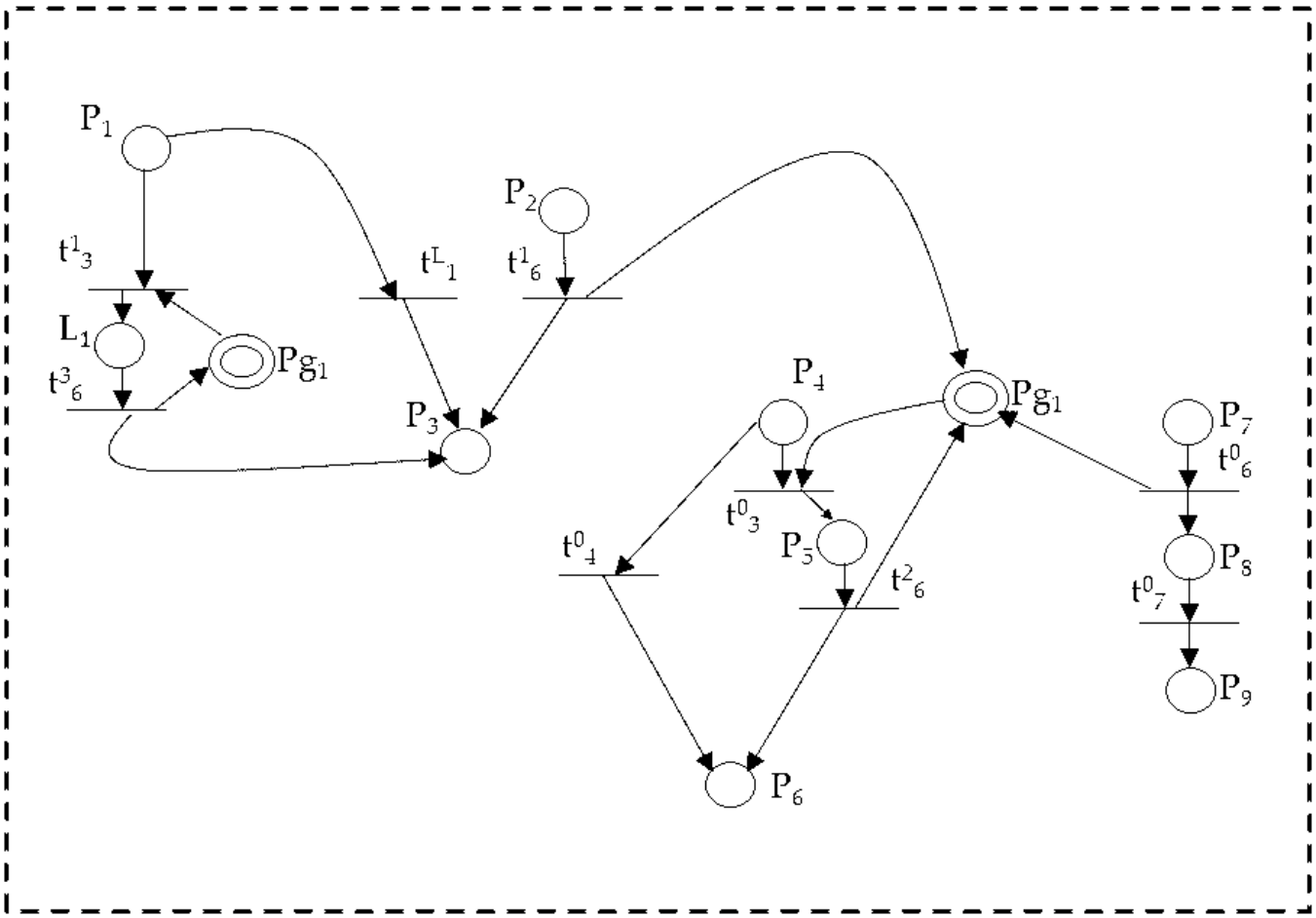

FIG. 7.4 – Le réseau ordinaire Rp_{2ord}

d'une transition abstraite et étiqueté par la fonction h. La fonction booléenne $closing$ permet de vérifier si une séquence de transitions d'interface σ, admet une séquence sous-jacente qui partant d'un certain $\Omega(t)$, dans un réseau ordinaire, mène à un marquage final. Soit σ une séquence de transitions d'interface, $\sigma.k$, avec $k \in 1...|\sigma|$, est la kème transition d'interface de la séquence σ.

Lemme 7.1. Si $\forall t \in T_{ab}$, le langage de $(Rp \setminus \{P_{par}\}, \Omega(t))$ est fini alors l'algorithme 1 se termine.

La preuve est évidente. Si les langages des différentes transitions abstraites sont finis, leurs projections sur les transitions d'interface le sont. L' ensemble $Delta$, pour chaque transition, se stabilise à une certaine itération de l'algorithme.

Exemple 7.4. Considérons l'exemple de la figure 7.3. Dans ce RdPRp, toutes les séquences de transitions de tir sont finies. Nous allons passer en revue les différentes étapes nécessaires pour construire les ensembles $Delta$ des transitions abstraites de Rp_2 :

Itération 1 :
$New(t_1) = \{\varepsilon^+, t_3^{0^-}\}$, $New(t_2) = \{t_6^{0^+}\}$ et $New(t_5) = \{t_6^{0^+}\}$ [5].
Le comportement de t_1 dicté par la séquence observée vide (fermante) sera simulé par une transition élémentaire t_1^L équivalente à t_1. En revanche, le comportement de t_1 dicté par la séquence observée t_3^0 (non fermante) sera simulée par une transition élémentaire t_3^1. Le comportement de t_2 dicté par la séquence observée t_6^0 (fermante) sera simulé par une transition élémentaire t_6^1. De la même manière, le comportement de t_5 dicté par la séquence observée t_6^0 (fermante) sera simulé par une transition élémentaire t_6^2.

Itération 2 :
$New(t_1) = \{\ t_3^0 t_6^{2^+}\}$, $New(t_2) = \{\emptyset\}$ et $New(t_5) = \{\emptyset\}$. Le comportement de t_1 dicté par la séquence observée $t_3^0 t_6^2$ est simulé par t_3^1 (déja définie) et t_6^3.

Itération 3 :
$New(t_1) = New(t_2) = New(t_5) = \{\emptyset\}$. L'algorithme s'arrête. Le réseau ordinaire obtenu est donné dans la figure 7.4.

7.3.3 Procédure d'Accessibilité

Le problème d'accessibilité consiste à vérifier si un marquage étendu Tr' est accessible à partir d'un autre marquage étendu Tr. Rappelons que, pour les RdPR, ce problème est ramené à un problème d'accessibilité dans les réseaux ordinaires. D'une manière similaire, le réseau ordinaire, associé à un certain $RdPRp$, construit par l'algorithme 4 est utilisé. Dans ce qui suit, nous présentons la procédure de décision de l'accessibilité. Nous allons passer en revue trois cas possibles selon que Tr et/ou Tr' soit (soient) vide(s).

[5] '+' (resp. '-') est utilisé pour les séquence fermantes (resp non fermantes).

a. Marquages étendus vides

La proprété d'accessibilité est vérifiée car l'arbre vide peut être atteint à partir de l'arbre vide.

b. Marquages étendus : source non vide et destination vide

Soit Tr un marquage étendu d'un RdPRp Rp. Pour prouver la propriété d'accessibilité de l'arbre vide à partir de Tr, nous construisons un réseau ordinaire, noté $Rp^*_{ord}(Tr)$, comme suit :

$Rp^*_{ord}(Tr)$=$\bigcup_{i=0}^{n-1} Rp^i_{ord}$, où :

1. n est le nombre de noeud(s) de Tr.

2. Rp^i_{ord} est une copie fraiche de Rp_{ord} que l'on associe au noeud v_i de Tr.

3. Finalement, il reste à relier les différentes copies entre elles de la manière suivante : pour chaque arc $v_i \xrightarrow{t} v_j$ de Tr, nous relions la copie de v_j à celle de v_i par le biais d'une transition élémentaire *tirable au plus une fois* et dite **finale**. Une telle transition permet de simuler une coupure τ à partir du noeud v_j ; Ses pré-conditions concernent des places locales de la copie de v_j et elle est sensiblisée uniquement pour des marquages finaux de la copie associée à v_j. Ainsi, son tir génére les sorties de la transition abstraite t dans la copie du noeud père v_i.

4. Enfin, chaque copie Rp^i_{ord} est marquée par le marquage du noeud v_i de Tr (en tenant compte bien évidemment du renommage des places dans les différentes copies). Le marquage des places partagées est également déduit à partir de Tr.

Le problème d'accessibilité opère sur le réseau marqué $Rp^*_{ord}(Tr)$ décrit ci-dessous et consiste à vérifier si l'on peut atteindre , dans la copie de v_0, un marquage final.

Exemple 7.5. La partie basse de la figure 7.5 présente un marquage étendu Tr associé au réseau Rp_2 de la figure 7.3, alors que la partie haute donne le réseau $Rp_{2^*_{ord}}(Tr)$. Comme Tr contient 3 noeuds, $Rp_{2^*_{ord}}(Tr)$ est composé de 3 copies fraiches de $Rp_{2_{ord}}$ (une copie par noeud). Notons que, par souci de clarté, nous avons représenté uniquement les modules "actifs" de chaque copie. La copie associée à v_1 (resp. v_2) est reliée à la copie de v_0 par la transition finale $t_{f,1}$ (resp. $t_{f,2}$). Notons que la place de contrôle marquée $F_{1,1}$ (resp. $F_{1,2}$) rend la transition $t_{f,1}$ (resp. $t_{f,2}$) *tirable au plus une fois*. Dans $Rp^*_{ord}(Tr)$, la place Pg_1 contient une marque. Aussi la place $P_{7,1}$ (resp. $P_{8,2}$) contient une marque (notons que ces marques sont déduites du marquage de Tr). Le problème d'accessibilité consiste à déterminer si l'on peut aboutir dans v_0 à un marquage final. En d'autres termes, il s'agit de déterminer si l'on peut atteindre un marquage final dans la copie de v_0 à partir de ($P_{7,1}$, $P_{8,2}$, Pg_1). Il est clair que la réponse est positive et donc $\perp$ peut être atteint à partir de Tr.

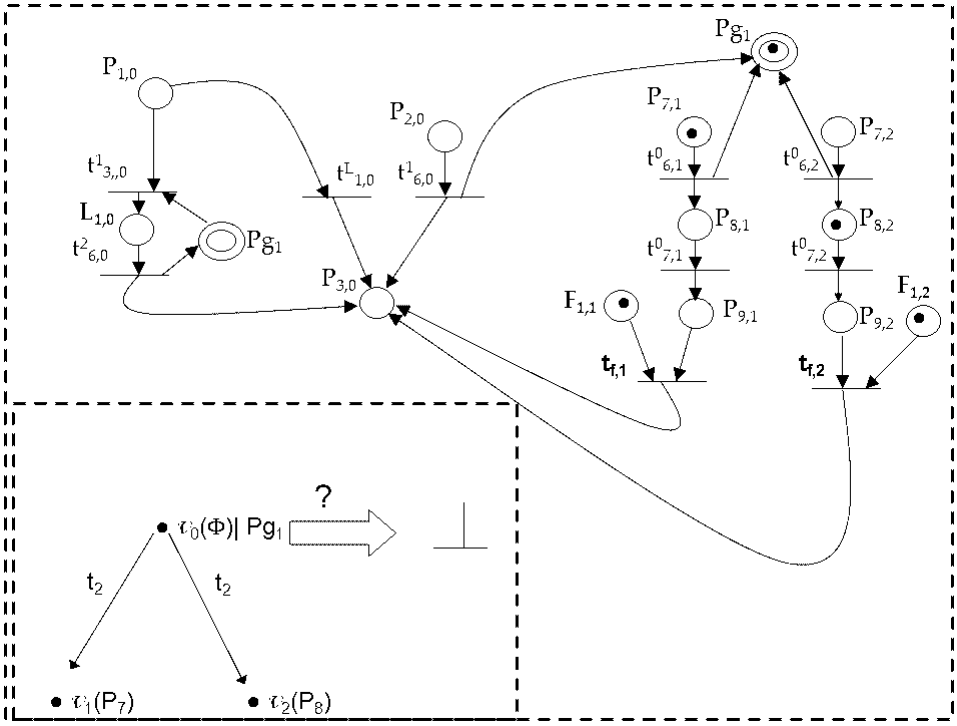

FIG. 7.5 – Accessibilité de l'arbre vide à partir d'un arbre non vide

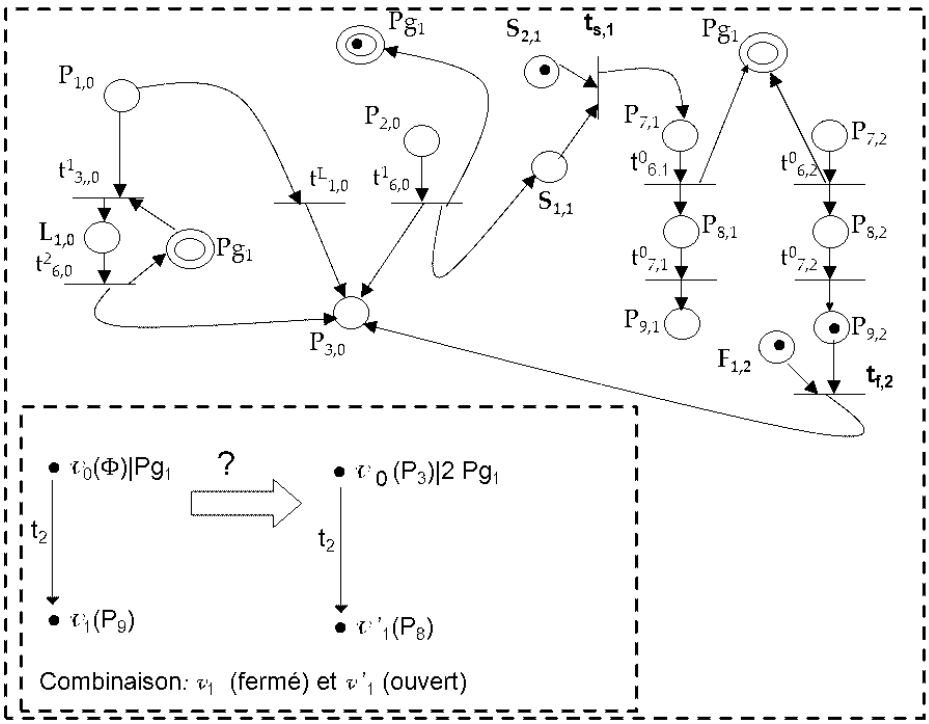

FIG. 7.6 – Accessibilité d'un arbre non vide à partir d'un arbre non vide

c. Marquages étendus non vides

Soient deux marquages étendus Tr et Tr' non vides d'un RdPRp Rp, pour vérifier l'accessibilité de Tr' à partir de Tr, il est indispensable de combiner les noeuds de Tr à ceux de Tr' (i.e. définir les prédictions entre les statuts des noeuds de Tr et ceux de Tr'). A ce niveau, nous considérons l'ensemble de toutes les combinaisons possibles, noté $CB(Tr, Tr')$, entre les noeuds de Tr et Tr' (voir le chapitre 2). La procédure d'accessibilité doit explorer ces combinaions tant que la propriété d'accessibilité n'est pas prouvée. Si toutes ces combinaisons échouent, Tr' n'est pas accessible à partir de Tr.
Etant donnée une combinaison $c \in CB(Tr, Tr')$ nous construisons un réseau ordinaire marqué, appelé $Rp^{**}_{ord}(Tr, Tr', c)$, associé à Tr et Tr' conformément à la combinaison c, défini comme suit :

$Rp^{**}_{ord}(Tr, Tr', c)$=$\bigcup_{i=0}^{n-1} Rp^{i}_{ord}$, où :

1. Les Rp^{i}_{ord} sont des copies fraiches de Rp_{ord} associées aux noeuds de Tr (une copie pour chaque noeud) et uniquement aux noeuds Tr' prédits ouverts selon la combinaison c. Ainsi si Tr comprend k noeuds et Tr' m noeuds dont p nouveau(x) ($p < m$) alors $n = k + p$.

2. Pour chaque arc $v_i \xrightarrow{t} v_j$ de Tr tel que v_j est prédit fermé selon c, relier la copie de v_j à celle de v_i exactement comme pour le **cas b**.

3. Pour chaque arc $v_i \xrightarrow{t} v_j$ de Tr' tel que v_j est prédit ouvert selon la combinaison c, relier la copie de v_i à celle de v_j par le biais d'une transition élémentaire *tirable au plus une seule fois*. Cette transition est sensibilisée dés qu'une transition simulant le tir de la transition abstraite t, dans la copie associée à v_i, est tirée. Quand la transition rajoutée est tirée, la copie associée à v_j devient marquée avec le marquage de départ de t. Ainsi le tir de cette dernière permet de simuler la création du noeud v_j marqué par $\Omega(t)$.

4. Le marquage initial de $Rp^{**}_{ord}(Tr, Tr', c)$ consiste à marquer les copies correspondant aux noeuds de Tr par leurs marquages déduits de Tr. Le marquage initial des places partagées est également déduit à partir de Tr.

5. Le problème d'accessibilité de Tr' à partir de Tr consiste à vérifier si l'on peut à atteindre, dans le réseau *marqué* $Rp^{**}_{ord}(Tr, Tr', c)$, les marquages, déduits à partir de Tr', des noeuds persistants, ceux ouverts dans leurs copies respectives et ceux des places partagées.

Exemple 7.6. La figure 7.6 présente un problème d'acessibilité de deux marquages étendus Tr et Tr' qui sont associés au réseau de la figure 7.3. Le réseau $Rp_{2^{**}_{ord}}(Tr, Tr', c)$ est donné dans la partie haute de la figure 7.6 (la prédiction c est donnée dans la même figure). Nous associons deux nouvelles copies de $Rp_{2_{ord}}$ aux deux noeuds de Tr et une troisième au noeud de Tr' prédit ouvert selon c (i.e. v'_1). La copie associée à v_1 est reliée par la transition finale (*tirable au plus une fois*) $t_{f,2}$ à la copie de son père (i.e. la copie de v_0). En revanche, la copie associée au noeud v'_1 de Tr' est reliée par la transition de contrôle $t_{s,1}$, *tirable au plus une fois*, qui permet de générer le marquage de départ de t_2 dans la copie de v'_1. Initialement, la place Pg_1 contient une marque. Le problème d'accessibilité consiste à déterminer si l'on peut atteindre $(2Pg_1, P_{3,0}, P_{8,1})$ à partir de $(Pg_1, P_{9,2})$. Il est évident que la réponse est négative.

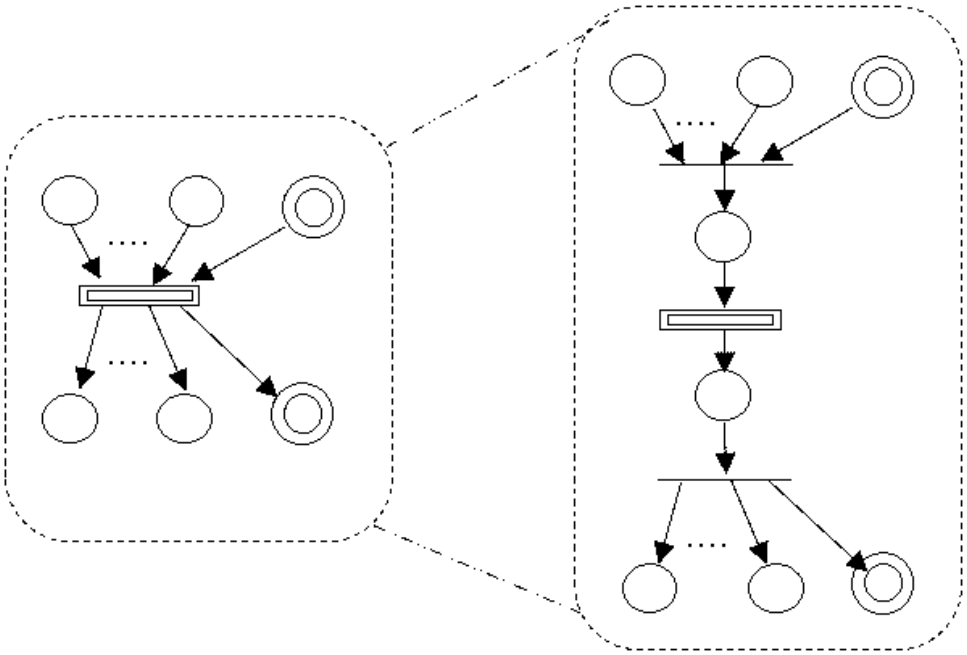

FIG. 7.7 – Extension des pré et post conditions des transitions abstraites aux places partagées

Remarques.
-La prise en charge des indices des RdPR est possible en prévoyant, dans chaque copie v_i (avec v_i un noeud à fermer), une transition *finale par indice*.

- Les pré-conditions et post-conditions des transitions abstraites peuvent être étendues aux places partagées sans affecter la décidabilité du problème d'accessibilité (voir figure 7.7).

7.4 Classes de Réseaux Récursifs Infinis

Le problème d'accessibilité est indécidable pour un RdPRp comportant au moins une transition abstraite ayant un langage infini (lemme 7.1). D'autre part, les transitions récursives ne sont pas sans poser de problème pour le procédure d'accessibilité. Nous allons définir des contraintes sémantiques que doivent vérifier les RdPRp afin de préserver leur caractère décidable, vis-à-vis du problème d'accessibilité. Dans ce qui suit, nous énonçons ces contraintes sémantiques :

1. Le nombre de tirs d'une transition d'interface ou d'une transition abstraite par un processus est borné par une valeur spécifiée par le modélisateur (cette borne sera notée par Max).
2. Les transitions abstraites apparaîssant dans une séquence de tir comportant au moins une transition d'interface doivent être différentes. En d'autres termes, les transitions abstraites récursives et les tran-

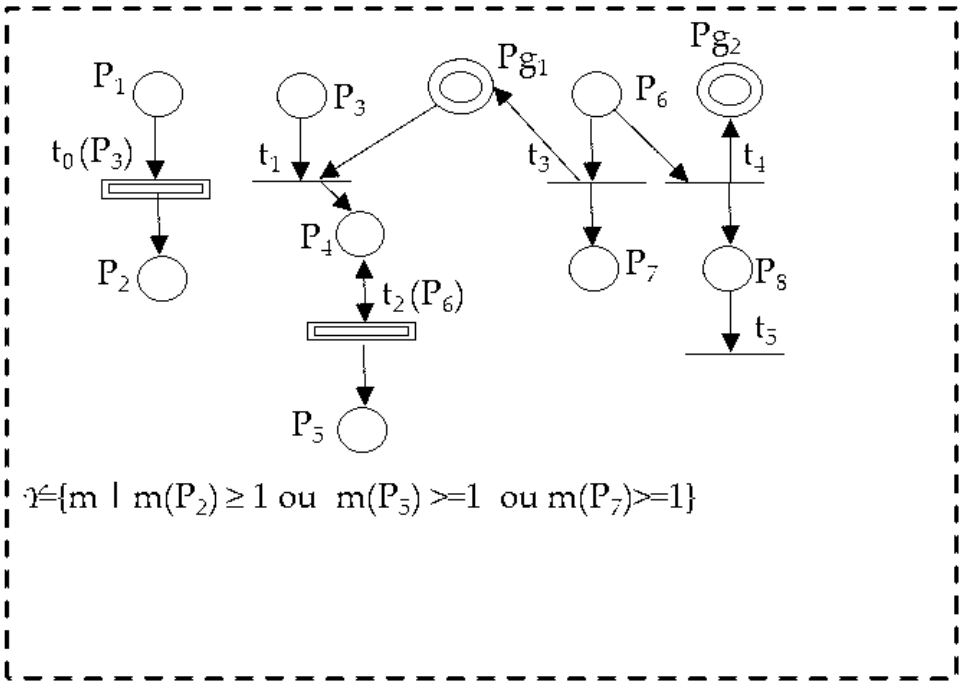

FIG. 7.8 – Un troisième RdPRp (Rp_3)

sitions d'interface ne doivent jamais apparaître dans la même séquence de tir.

Ces contraintes assurent que le $Delta$ de n'importe quelle transition abstraite est fini.

7.4.1 Adaptation de la Procédure d'Accessibilité au Cas Infini

A ce niveau, nous allons adapter l'algorithme **4** pour prendre en charge des RdPRp infinis mais *vérifiant les contraintes sémantiques* énoncées ci-dessous. Pour cela, nous associons à chaque transition abstraite t, deux transitions élémentaires que l'on note par t^I et t^{IO}. La première simule le tir de t, et a donc les pré-conditions et post-conditions immédiates de t. La deuxième transition t^{IO} est *équivalente* à t et simule, pour sa part, le tir de t suivi de sa fermeture. On note par $TS_{ab} = \{t^I, t^{IO} | t \in T_{ab}\}$ l'ensemble des transitions élémentaires qui sont associées à toutes les transitions abstraites. Nous procédons comme pour le cas fini, c'est à dire il s'agit de déterminer les réseaux d'interface des transitions abstraites et de construire un réseau ordinaire qui sera utilisé par la procédure d'accessibilité. Pour ce faire, une transition t est substituée par t^I et éventuellement par t^{IO} si elle s'avère "fermable". Ici, les séquences observées des transitions abstraites calculées, afin de construire les réseaux d'interface, s'expriment en fonction de ces transitions élémentaires additives (i.e. les éléments de TS_{ab}). Par ailleurs, les contraintes imposées permettent de générer un ensemble fini de séquences observées pour chaque transition abstraite. L'ensemble des séquences observées que l'on peut associer à une transition abstraite est borné par un ensemble particulier $Bigset$. Cet ensemble contient toutes les séquences de transitions d'interface et/ou de transitions de TS_{ab}, telles que, dans n'importe quelle séquence de cet ensemble, le nombre de franchissements d'une transition d'interface

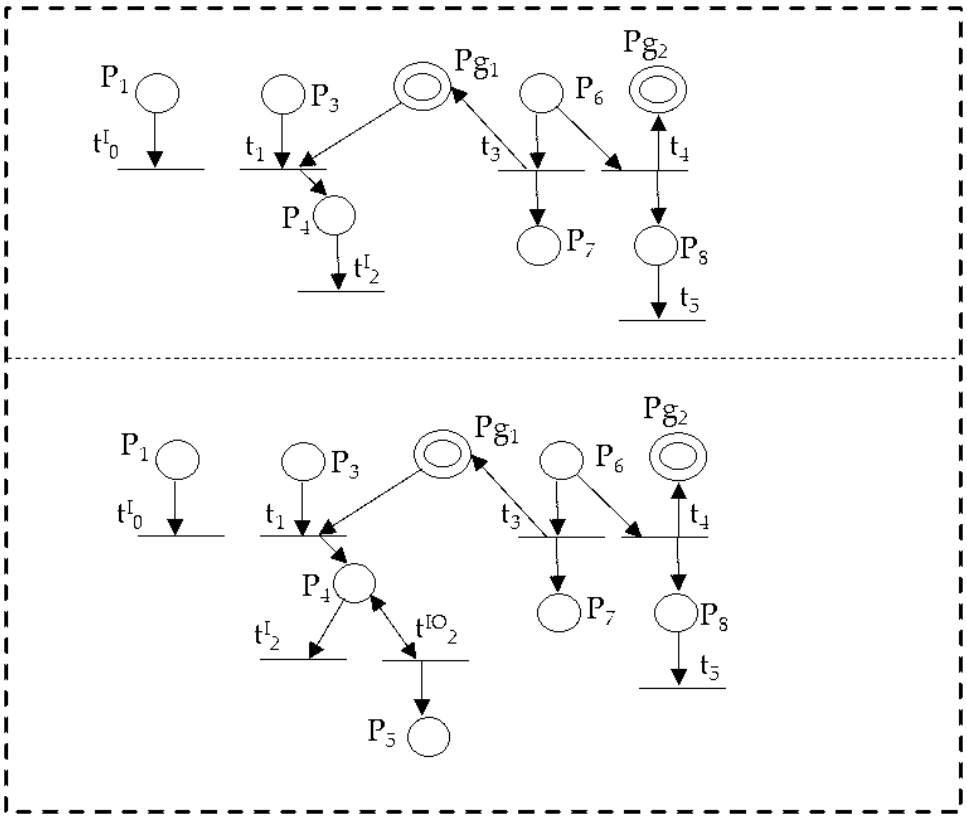

FIG. 7.9 – (Rp_3^0 et Rp_3^1)

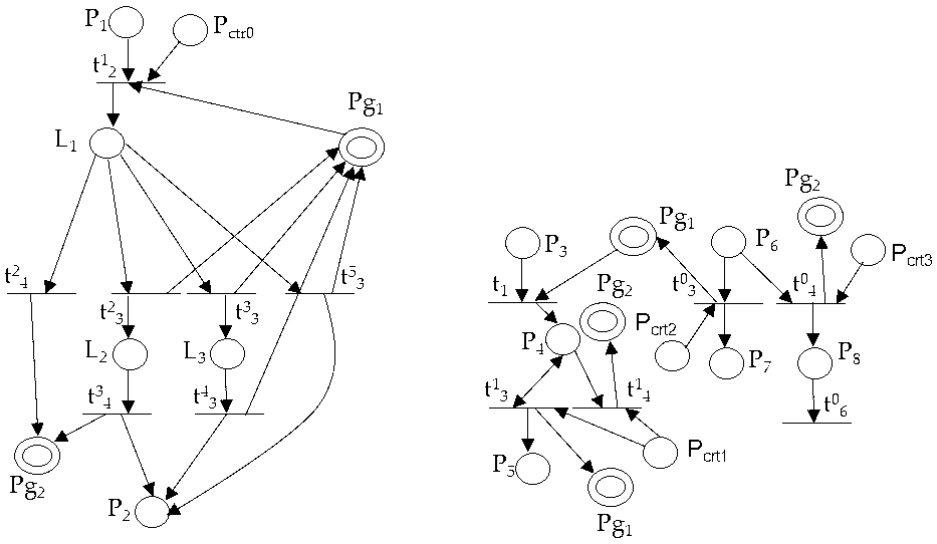

FIG. 7.10 – Le réseau ordinaire $Rp_{3_{ord}}$

soit borné par Max et la somme des occurrences des transitions simulant une *même transition abstraite* soit aussi bornée par Max. $Bigset$ est défini comme suit :

$$Bigset = \{\sigma | \sigma \in (T_{in} \cup TS_{ab})^* \cup \{\varepsilon\} \wedge \forall t \in T_{in}, |\sigma_{|t}| \leq Max \wedge \forall t \in T_{ab}, |\sigma_{|t^I}| + |\sigma_{|t^{IO}}| \leq Max\}.$$

A la fin du calcul itératif, ces transitions additives sont remplacées par des séquences de transitions d'interface. Nous allons passer en revue les principales étapes de la procédure d'accessibilité que nous proposons.

1. Rp^0_{ord} est obtenu en remplaçant chaque transition abstraite t par t^I. Initialement, les ensembles $Delta$ associées aux transitions abstraites sont vides. La fonction d'étiquetage h sera étendue pour permettre d'observer, en plus des transitions d'interface, les transitions de TS_{ab}. Ainsi, les éléments de $Delta$ seront, dans un premier temps, un mélange de ces deux types de transitions.

2. A chaque itération k de l'algorithme, pour chaque transition t, nous faisons l'intersection entre le langage du réseau ordinaire marqué et étiqueté $(Rp^{k-1}_{ord}, \Omega(t), h)$ et l'automate d'états finis qui reconnait les mots de l'ensemble $Bigset$. Le langage de l'intersection, forcément fini [6], donne la valeur de $\Delta(t)$ à l'itération k. Si il existe $\sigma \in \Delta(t)$ une séquence observée fermante de t alors la transition t^{IO} est rajoutée au réseau ordinaire Rp^k_{ord}.

3. Vu les contraintes sémantiques posées, les ensembles $Delta$ sont égaux ou inlus dans l'ensemble *fini* $Bigset$. Chaque ensemble $Delta$ finit donc par se stabiliser.

4. A la fin de l'alogrithme 4, les ensembles $Delta$ sont combinés entre eux afin de remplacer les transitions de TS_{ab} par des séquences composées *uniquement de transitions d'interface*. Il faut souligner que t^I est substiuée par des *séquences observées non fermantes* de la transition abstraite t, alors que t^{IO} par des *séquences observées fermantes*.

Comme pour le cas fini, ces ensembles $Delta$ sont utilisés pour construire les réseaux d'interface des transitions abstraites. Ainsi, un réseau ordinaire est construit à l'aide de ces réseaux d'interface. Des copies fraiches du réseau ordinaire construit sont dédiées aux noeuds (i.e. aux processus) des marquages étendus à comparer. A ce stade, dans chaque copie, aucune conrainte sur les nombres de franchissements des transitions d'interface et d'activations des réseaux d'interface n'existe. Afin de tenir compte des contraintes imposées à chaque processus (donc à chaque copie du réseau ordinaire construit), des places de contrôle fraiches seront rajoutées au réseau ordinaire ; elles permettent de contrôler d'une part le nombre de franchissements de chaque transition d'interface et d'autre part, le nombre "d'activation" de n'importe réseau d'interface d'une transition abstraite. Notons que la seule différence par rapport au cas fini consiste dans le marquage initial de chaque copie. Ce marquage, en plus des marques décrites plus haut, englobe Max jetons dans chaque place de contrôle.

Exemple 7.7. Considérons l'exemple de la figure 7.8. Dans ce RdPRp, les séquences de transitions peuvent être infinies. Nous supposons que $Max = 2$, Le réseau ordinaire initial associé à ce RdPRp est donné dans la partie haute de la figure 7.9. L'analyse du RdPRp Rp_3 se fait comme suit :

[6] Le calcul du langage résultant de l'intersection d'un RdP marqué et étiqueté et d'un automate d'états finis est décidable.

Itération 1 : L'intersection de Rp_3^0 marqué par le marquage de départ de t_0 avec l'automate fini qui reconnait les mots de $Bigset$, permet de calculer les séquences observées de t_0 définies comme suit :
$\Delta(t_0) = \{t_1 t_2^{I^-}\}$.
De même, l'intersection de ce réseau ordinaire marqué par le marquage de départ de t_2 avec l'automate fini associé à l'ensemble $Bigset$, permet de calculer les séquences observées de t_2 définies comme suit :
$\Delta(t_2) = \{t_3^+, t_4^-\}$.
Contraitement à t_0, à cette itération t_2 s'avère fermable (elle admet une séquence fermante). Par conséquent, le réseau ordinaire en cours de construction est enrichi par la transition élémentaire t_2^{IO} (voir la partie basse de la figure 7.9).
Itération 2 : Comme pour l'itération précédente, l'intersection entre les langages du réseau Rp_3^1 marqué par le marquage de départ de t_0 et l'automate associé à $Bigset$ est calculée. Le même calcul est fait pour la transition abstraite t_2 et l'automate associé à $Bigset$. Ceci nous permet d'obtenir les ensembles $Delta$:
$\Delta(t_0) = \{t_1 t_2^{I^-}, t_1 t_2^{IO^+}, t_1 t_2^{IO} t_2^{IO^+}, t_1 t_2^{IO} t_2^{I^+}, \}$ et $\Delta(t_2) = \{t_3^+, t_4^-\}$.

A l'itération 3, l'algorithme se termine car les ensembles $Delta$ se stabilisent. Les ensembles $Delta$ sont combinés entre eux afin de sustituer les transitions de TS_{ab} par des transitions d'interface :
$\Delta(t_0) = \{t_1 t_4{}^-, t_1 t_3{}^+, t_1 t_3 t_3{}^+, t_1 t_3 t_4{}^+\}$ et $\Delta(t_2) = \{t_3^+, t_4^-\}$.

Le réseau ordinaire associé à Rp_3 est donné à la figure 7.10. La place de contrôle P_{ctr0} (resp. P_{ctr1}) permet de limiter le nombre d'activations du réseau d'interface de la transition abstraite t_0 (resp. t_2), alors que la place de contrôle P_{ctr2} (resp. P_{ctr3}) est utilisée pour limiter le nombre de franchissements de la transition d'interface t_3 (resp. t_4).

7.5 Application

Nous rajoutons le temps aux RdPRp en suivant les mêmes étapes que pour les RdPRT. Nous utilisons ce nouveau modèle temporel pour spécifier le troisième scénario multimédia (avec synchronisation extra parent) présenté dans le chapitre 6. Le modèle obtenu est donné dans la figure 7.11. Dans cette section, nous nous focalisons sur les deux relations de synchronisation "extra parent" décrites dans cet exemple. Rappelons que ces deux relations de synchronisation sont :

- $begin(aud) = begin(img_1) + 3$
- $begin(img_2) = end(aud)$ +2

Les médias img_1 et img_2 ont le même parent seq_0, alors que aud a un parent différent par_1.
Un module est dédié à chaque objet de base ou composite. Des places partagées sont utilisées pour faire communiquer les processus (i.e. les instances d'exécution des modules) entre eux. Une première place partagée Pp_1 est utilisée pour la synchronisation entre img_1 et aud. Un arc immédiat représenté en pointillé relie la transition ts_{img_1} à cette place. Aussitôt que le processus, créé par le tir de ts_{seq_0}, tire la transition abstraite ts_{img_1}, Pp_1 reçoit une marque. Cette place est en entrée de la transition ts_{aud} conditionnant ainsi son tir ; il faut noter que le tir de ts_{aud} se fera par le processus créé suite au tir de la transition ts_{par_1}. Par ailleurs,

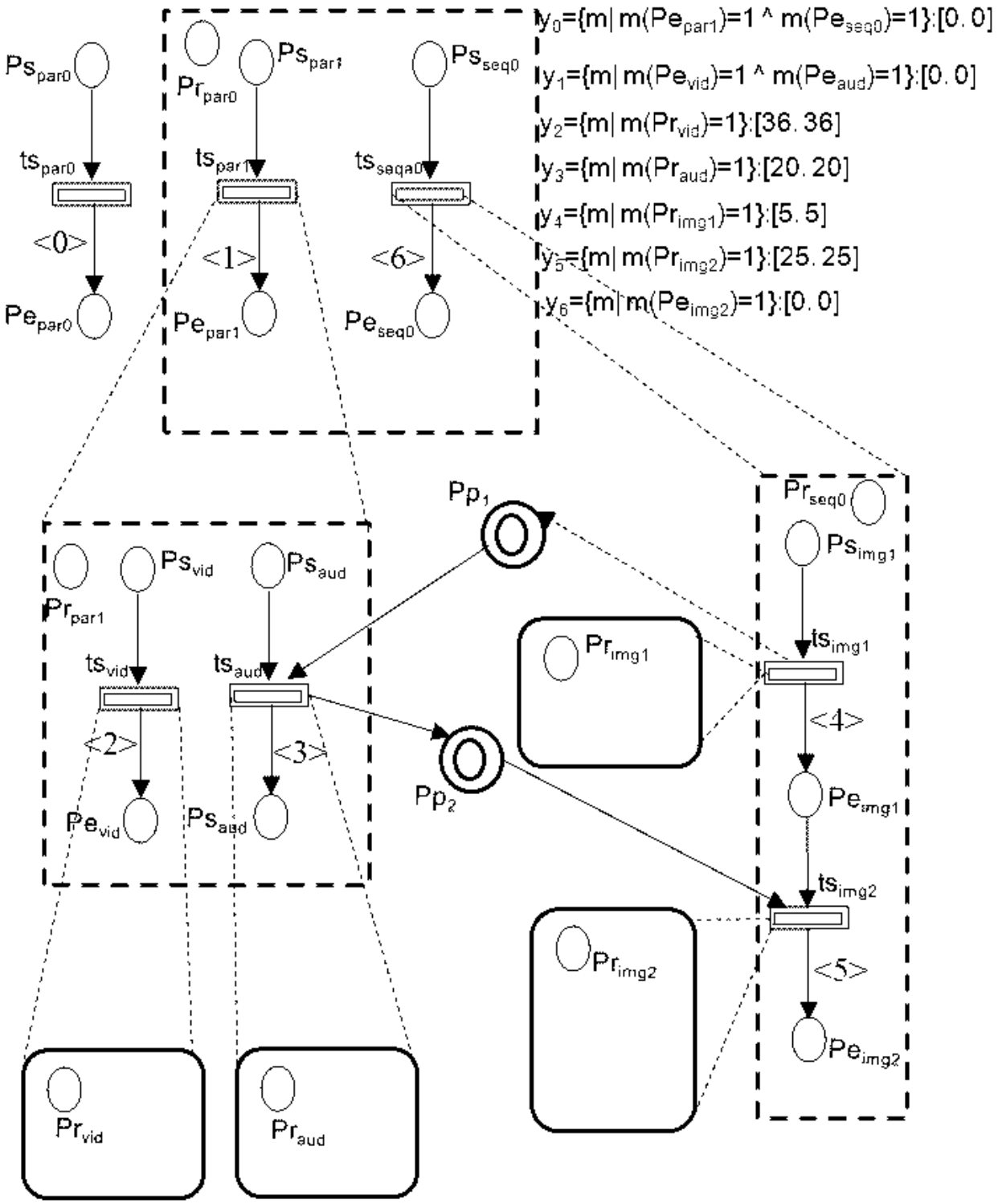

FIG. 7.11 – Modélisation d'un document multimédia à l'aide des RdPRp temporels

l'intervalle temporel $[3, 3]$ est attaché à ts_{aud} (voir la relation de synchronisation entre ts_{img1} et ts_{aud}). De la même manière, une deuxième place partagée Pp_2 est utilisée pour la synchronisation entre img_2 et aud. Cette place sera marquée quand la transition ts_{aud} est fermée au sein du processus créé par ts_{par_1}. Pp_2 est en entrée de ts_{img_2} conditionnant ainsi le tir de celle-ci par le processus processus créé par ts_{seq_0}. De plus, l'intervalle temporel $[2, 2]$ est attaché à la transition ts_{img_2} (voir la relation de synchronisation entre ts_{aud} et ts_{img_2}). Il est à noter que l'intervalle temporel $[0, 0]$ est attaché aux autres transitions du modèle. Dans ce modèle, une marque est produite dans chaque place partagée par un processus donné et elle est consommée par un autre processus.

7.6 Conclusion

Dans ce chapitre, nous avons proposé une extension des RdPR : les réseaux de Petri récursifs avec places partagées. Les RdPRp contrôlent mieux les naissances des processus à l'aide des sorties immédiates des transitions abstraites. Par ailleurs, la notion de places partagées entre les processus augmente les capacités de communication entre les processus. Nous avons défini la procédure d'accessibilité pour ce modèle étendu. Cette procédure se ramène à un problème d'accessibilité dans les réseaux classiques. En effet, chaque transition abstraite est substituée par un sous réseau ordinaire qui fait abstraction des comportements locaux de la transition abstraite. Nous avons montré que la procédure devient indécidable pour les réseaux dont les langages sont infinis,. Nous avons été amenés à introduire des contraintes sémantiques que doivent vérifier les RdPRp pour préserver le caractère décidable du problème d'accessibilité. Le temps peut être ajouté à ce nouveau modèle comme cela a été fait pour les RdPR. Le modèle temporel obtenu prend en charge avec élégance la modélisation des documents multimédias incluant des relations de synchronisation "extra parent".

Algorithm 4 Calcul des ensembles $Delta$

1: $Rp' = Rp \setminus \{T_{ab}\}$; $copy(Rp^0_{ord}, Rp')$; $T'_{in} = T_{in}$;
2: **for** $t \in T_{ab}$ **do**
3: $Old(t) = \{\emptyset\}$; ;
4: **end for**
5: $finish = false$; $i = 0$;
6: **while** $not finish$ **do**
7: $finish = true$; $i = i + 1$; $copy(Rp^i_{ord}, Rp^{i-1}_{ord})$;
8: **for** $t \in T_{ab}$ **do**
9: $New(t) = Delta(t, Rp^{i-1}_{ord}) \setminus Old(t)$;
10: **for** $\sigma \in New(t)$ **do**
11: **case** $|\sigma|$ **of** :
12: **0** : /* *le comportement de* t *sera simulé par une nouvelle transition élémentaire**/
13: $t' = identifier.trans(t)$; $affect.input.locale(t, t')$; $affect.outputi.locale(t, t')$;
14: **if** $closing(\sigma, t, Rp^{i-1}_{ord})$ **then** $affect.outputd.locale(t, t')$;
15: $add.trans(t', Rp^i_{ord})$;
break ;
16: **1** : /* *le comportement de* t *sera simulé par une nouvelle transition élémentaire**/
17: $t' = identifier.trans(\sigma.1)$; $affect.input.locale(t, t')$; $affect.par(\sigma.1, t')$; $affect.outputi.locale(t, t')$;
18: **if** $closing(\sigma, t, Rp^{i-1}_{ord})$ **then** $affect.outputd.locale(t, t')$;
19: $add.trans(t', Rp^i_{ord})$; $add.interface(t', T'_{in})$;
break ;
20: **Otherwise** : **/*** *le comportement de* t *sera simulé par un ensemble de nouvelles transitions élémentaires* $t'_1, ..., t'_m$ *(où* m *est la longueur de* σ*) qui sont séquentielles*/*
21: $t'_1 = identifier.trans(\sigma.1)$; ... ; $t'_m = identifier.trans(\sigma.m)$;
/* *Les transitions de chaque paire* (t'_k, t'_{k+1}) *(avec* $1 \leq k < m$*) sont reliée par une nouvelle place locale non marquée* L_k */
22: **for** $k \in 1..|\sigma| - 1$ **do**
23: $L_k = identifier.place()$; $add.place(L_k, Rp^i_{ord})$; $W^+(L_k, t'_k) = 1$; et $W^-(L_k, t'_{k+1}) = 1$;
24: **end for**
/* *Les entrées de* t *sont associées à la première transition de la séquence. Alors que les sorties de* t *à la dernière transition de la séquence, si* σ *est une séquence observée fermante de* t **/*
25: $affect.input.locale(t, t'_1)$; $affect.outputi.locale(t, t'_m)$;
26: **if** $closing(\sigma, t, Rp^{i-1}_{ord})$ **then** $affect.outputd.locale(t, t'_m)$;
/**De plus, chaque* t'_k *a les mêmes pré-conditions et post-conditions en terme de places partagées que la kème tranition de* σ **/*
27: **for** $k \in 1..|\sigma|$ **do**
28: $affect.par(\sigma.k, t'_k)$; $add.trans(t'_k, Rp^i_{ord})$; $add.interface(t'_k, T'_{in})$;
29: **end for**
endcase
30: **end for**
31: $Old(t) = Old(t) \cup New(t)$;
32: **IF** $New(t) \neq \{\emptyset\}$ **then** $finish = false$;
33: **end for**
34: **end while**

Chapitre 8

Conclusions et Perspectives

Bilan : Avec l'essor des systèmes temps réel en taille et en complexité, leur vérification s'impose. En particulier, des modèles puissants et proches du comportement temporel réel des applications sont nécessaires. Cette thèse apporte des nouvelles contributions pour ces directions. En effet, nous avons proposé un modèle temporel basé sur les réseaux de Petri récursifs permettant de spécifier, de modéliser et d'analyser des systèmes temps réel complexes. En particulier, des systèmes temps réel à structure dynamique sont bien pris en charge par le modèle que nous proposons.

Dans un premier temps, nous avons proposé un modèle temporel basé sur les sémantiques respectives des réseaux de Petri T-temporels et des réseaux de Petri récursifs. La définition d'une sémantique formelle, claire et concise en terme d'un système de transitions temporisé du modèle proposé a permis de le décrire sans ambiguité. Par ailleurs, les capacités de modélisation du modèle ont été mises en evidence grâce à la modélisation d'un système de serveurs distants. En effet, nous avons montré qu'un même composant peut être utilisé non seulement avec des contextes différents (marquages différents) mais aussi avec des contraintes temporelles différentes (ceci évite la duplication des composants dans les modèles). De plus, le concept de préemption externe permet de contrôler les composants d'un modèle. La préemption d'un composant peut avoir lieu à partir du moment où certaines préconditions et/ou certaines contraintes temporelles sont satisfaites.

Dans un premier temps, une première méthode de calcul de l'espace d'états de ce modèle est proposée : le graphe des classes d'états étendu. Des conditions de finitude d'un tel graphe sont clairement présentées et justifiées dans cette thèse. Dans un deuxième temps, nous avons défini une deuxième technique modulaire pour la construction des graphes des classes d'états. Cette deuxième technique évite les comparaisons des marquages étendus qui constituent des opérations plus au moins coûteuses en terme de temps d'exécution. Par ailleurs, les graphes des classes d'états sont classiques et les outils d'analyse des réseaux de Petri T-temporels peuvent donc être utilisés.

Des modélisations de documents multimédias à l'aide des réseaux de Petri récursifs temporels ont permis de mettre en valeur, encore une fois, les capacités de modélisation de notre modèle. En effet, les modèles générés sont compacts, lisibles et élégants. Par ailleurs, ils reflètent les structures temporelles hiérarchiques de ces documents. Toutefois, la modélisation des relations synchronisation extra parent a montré les limites du modèle proposé, et donc de son modèle sous-jacent les réseaux de Petri récursifs, en terme de communication entre processus.

La deuxième partie de notre thèse concerne une deuxième extension des réseaux de Petri récursifs. Dans cette deuxième extension, la notion de places partagées entre les processus est introduite dans le but de les faire communiquer. Le résultat principal de cette extension est la procédure d'accessibilité que nous avons développée pour ce modèle étendu. Cette procédure se ramène à un problème d'accessibilité dans les réseaux de Petri classiques. En effet, chaque transition abstraite est substituée par un sous réseau ordinaire qui fait abstraction des comportements locaux de la transition abstraite et met l'accent sur les transitions élémentaires qui manipulent les places partagées. Le modèle étendu est doté de contraintes sémantiques afin de préserver la décidabilité du problème d'accessibilité pour des réseaux dont les langages sont infinis.

Perspectives : Dans la continuité de ce travail un certain nombre de points restent à réaliser pour notre première extension des réseaux de Petri récursifs :

- À court terminer, nous pensons utiliser l'outil des RdPRT qui est en cours d'expérimentation pour analyser des applications réelles.

- Définir des techniques permettant de vérifer des propriétés temporelles quantitatives des modèles RdPRT.

Quant à la deuxième partie de cette thèse, nous proposons de renforcer les moyens de communication entre les processus. Pour cela, nous proposons de distinguer les jetons mis dans les places partagées. Par exemple, on pourrait penser à associer l'identité d'un processus à un jeton. Ainsi, un processus emetteur peut donner son identité à un jeton et peut attendre des jetons qui lui sont destinés, c'est à dire ceux qui portent son identité. Par conséquent, avec ce modèle récursif **coloré**, il est possible d'établir une communication entre les processus **sans confusion**. Il est clair que le modèle proposé doit préserver la décidabilité des propriétés de base telles que l'accessibilité. Le modèle récursif coloré prendra en charge des applications diverses telles que les protocoles d'applications mobiles et autres.

Bibliographie

[1] P.A. Abdulla and A. Nylén. Timed Petri nets and bqos. In *22nd International Conference on Application and Theory of Petri Nets (ICATPN'01)*, volume 2075 of *Lecture Notes in Computer Science*, pages 53–70, Newcastle upon Tyne United Kingdom, june 2001. Springer-Verlag.

[2] R. Alur and D.L. Dill. A theory of timed automata. In *Theoretical Computer Science*, volume 126(2), pages 183–235, 1994.

[3] A. ARNOLD. Finite transition system. Prentice Hall, 1994.

[4] B. Aspvall and Y. Shiloach. A polynomial time algorithm for solving systems of linear inequalities with two variables per inequality. In *SIAM Journal on Computing*, volume 9, pages 827–845, 1980.

[5] B. Berthomieu and M. Diaz. Modeling and Verification of Time Dependent Systems Using Time Petri Nets. *Journal IEEE Transactions on Software Engineering*, 17(3) :259–273, March 1991.

[6] B. Berthomieu, P.O. Ribet, and F. Vernadat. The tina tool : Construction of abstract state space for petri nets and time petri nets. In *Journal of Production Research*, volume 42, 2004.

[7] B. Berthomieu and F. Vernadat. State class constructions for branching analysis of time petri nets. In *Rapport Interne LAAS N'02130*, Paris, 2002.

[8] H. Boucheneb and J. Mullins. Analyse des réseaux temporels : Calcul des classes en o(n2) et des temps de chemin en o(mŒn).

[9] M. Boyer and R.H Olivier. Comparison of the expressiveness of arc, place and transition time Petri nets. In *28th International Conference on Application and Theory of Petri Nets and other models of concurrency (ICATPN'07)*, volume 4546 of *Lecture Notes in Computer Science*, pages 63–82, Siedlce, Poland, june 2007. Springer-Verlag.

[10] M Boyer and F. Vernadat. Language and bisimulation relations between subclasses of timed Petri nets with strong timing semantic. In *Technical report*, LAAS, 2000.

[11] B. Bérard, F. Cassez, S. Haddad, D. Lime, and H.O. Roux. Comparison of different semantics for time Petri nets. In *Automated Technology for Verification and Analysis (ATVA'05)*, volume 3707 of *Lecture Notes in Computer Science*, Taiwan, 2005. Springer.

[12] B. Bérard, F. Cassez, S. Haddad, D. Lime, and O.H Roux. When are timed automata weakly timed bisimilar to time Petri nets ? In *25th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2005)*, volume 3821 of *Lecture Notes in Computer Science*, Hyderabad, India, December 2005. Springer.

[13] B. Bérard, F. Cassez, S. Haddad, and D. Lime O.H. Roux. Comparison of the expressiveness of timed automata and time petri nets. In *3nd International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS 05)*, volume 3829 of *Lecture Notes in Computer Science*, Uppsala, Sweden, september 2005. Springer-Verlag.

[14] F. Cassez and O.H. Roux. Traduction structurelle des réseaux de Petri temporels vers les automates temporisés. In *Modélisation des Systèmes Réactifs, (MSR'03)*, Metz, France, October 2003.

[15] F. Cassez and O.H. Roux. Structural translation from time Petri nets to timed automata. In *Fourth International Workshop on Automated Verification of Critical Systems (AVoCS'04)*, Electronic Notes in Theoretical Computer Science, London (UK), 2004. Elsevier.

[16] A. Cerone and A. Maggiolo-Schettini. Timed based expressivity of time petri nets for system specification. volume 216, pages 1–53, 1999.

[17] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. Introduction à l'algorithmique. In *DUNOD*, Paris, 1994.

[18] J.P Courtiat and R.C. Oliveira. Proving temporal consistency in a new multimedia synchronization model. In *Proc of ACM Multimedia'96*, pages 141–152, Boston USA, November 1996.

[19] D. Dahmani, J.M Ilié, and M. Boukala. Time recursive Petri nets. In *PNSE'07, International Workshop on Petri Nets and Software Engineering*, pages 256–26, Siedlce, Poland, 2007.

[20] D. Dahmani, J.M Ilié, and M. Boukala. Time recursive Petri nets. In *Journal Transactions on Petri Nets and Other Models of Concurrency (ToPNoC) K. Jensen, W. van der Aalst, and J. Billington (Eds.)*, volume ToPNoc I, LNCS 5100 of *Lecture Notes in Computer Science*, pages 104–118, Berlin Heidelberg, 2008. Springer-Verlag.

[21] G. B. Dantzig. Linear programming and extensions. In *PhD thesis*, Princeton University Press, 1963. Princeton, NJ.

[22] M. Diaz. Les réseaux de petri : Modèles fondamentaux. Hermes, 2001.

[23] C. Dufourd, A. Finkel, and P. Schnoebelen. Reset nets between decidability and undecidability. In *Proc of the 25th Int. Colloquium on Automata, Lan- guages and Programming*, volume 1443 of *Lecture Notes Computer Science*, pages 103–115, Aalborg, Denmark, July 1998. Springer-Verlag.

[24] S. Eilenberg and M. Schutsenberger. Rational sets in commutative monoids. In *Journal of algebra*, volume 13, pages 173–191, 1969.

[25] D.F Escrig, V.V. Ruiz, and O.M. Alonso. Decidability of properties of timed-arc petri nets. In *21st International Conference on Application and Theory of Petri Nets (ICATPN'00)*, volume 1825 of *Lecture Notes in Computer Science*, pages 187–206, Aarhus, Denmark, june 2000. Springer Verlag.

[26] L. Gallon. Le modèle réseaux de petri temporisées stochastiques : extensions et applications. In *PhD thesis*, Université Paul Sabatier Toulouse France, 1997.

[27] G. Gardey, D. Lime, M. Magnin, and O.H. Roux. Romeo a tool for analyzing time petri nets. in computer aided verification. In *17th International Conference, CAV 2005*, volume 3576, Edinburgh, Scotland, UK, 2005. Springer-Verlag.

[28] S. Haddad and D. Poitrenaud. Decidability and undecidability results for recursive Petri nets. Rapport de Recherche LIP6 1999/019, Université Paris 6 - CNRS - Laboratoire d'informatique de Paris 6, Paris, France, September 1999.

[29] S. Haddad and D. Poitrenaud. Theoretical aspects of recursive Petri nets. In S. Donatelli and J. Kleijn, editors, *Proceedings of the 20th International Conference on Application and Theory of Petri Nets (ICATPN'99)*, volume 1639 of *Lecture Notes in Computer Science*, pages 228–247, Williamsburg, Virginia, USA, June 1999. Springer Verlag.

[30] S. Haddad and D. Poitrenaud. Modelling and analyzing systems with recursive Petri nets. In *Proceedings of the 5th Workshop on Discrete Event Systems (WODES'2000)*, pages 449–458, Ghent, Belgium, August 2000. Kluwer Academic Publishers.

[31] S. Haddad and D. Poitrenaud. Checking linear temporal formulas on sequential recursive Petri nets. In *Proceedings of the 8th International Symposium on Temporal Representation and Reasonning (TIME'01)*, pages 198–205, Cividale del Friuli, Italie, 2001. IEEE Computer Society.

[32] S. Haddad and D. Poitrenaud. Recursive Petri Nets, an Expressive Model for Discrete Event Systems. *Acta Informatica*, 2007.

[33] H.M. Hanisch. Analysis of place/transition nets with timed-arcs and its application to batch process control. In *14th International Conference on Application and Theory of Petri Nets (ICATPN'93)*, volume 691 of *Lecture Notes in Computer Science*, pages 282–299. Springer Verlag, 1993.

[34] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. In *Information and Computation*, volume 111(2), pages 193–244, 1994.

[35] M. Jantzen. On the hierarchy of petri net languages. volume 13(1), pages 19–30, June 1979.

[36] K. Jensen. Coloured petri nets. basic concepts, analysis methods and prac- tical use. *Basic Concepts. Monographs in Theoretical Computer Science*, 1, 1997.

[37] N. D. Jones, L. H. Landweber, and Y. E. Lien. Complexity of some problems in petri nets. volume 4, pages 277–299, 1977.

[38] M. Jourdan, N. Layaïda, C. Roisin, L. Sabry-Ismail, and L. Tardif. Madeus, an authoring environment for interactive multimedia documents. In *ACM Multimedia'98*, pages 267–272, Bristol (UK), September 1998.

[39] W. Khansa, J.P. Denat, and S. Collart-Dutilleul. P-time Petri nets for manufacturing systems. In *International Workshop on Discrete Event Systems, WODES'96*, pages 94–102, Edinburgh (U.K.), august 1996.

[40] W. Khanza. Réseau de petri P-temporels. contribution à l'étude des systèmes à évènements discrets. In *PhD thèse*, Université de Savoie, 1992.

[41] A. Kiehn. Petri nets systems and their closure properties. *In Advances in Petri Nets*, 424 :306–328, 1989.

[42] M. Koler and H. Rolke. Properties of object petri nets. In *Proc. of the 25th Int. Conf. on Application and Theory of Petri Nets*, volume 3099 of *Lecture Notes in Computer Science*, pages 278–297, Bologna, Italy, june 2004. Springer-Verlag.

[43] S.R. Kosaraju. Decidability of reachability in vector addition systems. In *Proc. 14th Annual Symposium on Theory of Computing*, pages 267–281, 1982.

[44] J. L. Lambert. Some consequences of the decidability of the reachability problem for Petri nets. In *Advances in Petri Nets*, volume 340 of *Lecture Notes in Computer Science*, pages 266–282. Springer-Verlag, June 1988.

[45] D. Lime. Vérification d'application temps réel à l'aide de réseaux de petri temporels étendus. In *Thèse de doctorat*, Université de Nantes, France, décembre 2004.

[46] I. Lomazova and Ph. Schnoebelen. Some decidability results for nested Petri nets. In *Proc. of the 3rd Int. Andrei Ershov Memorial Conf. Perspectives of System Informatics*, volume 1755 of *Lecture Notes in Computer Science*, pages 208–220, Novosibirsk, Russia, july 2000. Springer-Verlag.

[47] T. Madi and A. El Haddad. Modélisation des aspects temporels du langage Smil par des réseaux de Petri de haut niveau. In *Mémoire d'ingéniorat encadré par D. Dahmani et S. Mazouz*, Université Houari Boummedienne Alger, Juin 2007.

[48] E.W. Mayr. An algorithm for the general Petri net reachability problem. In *Proc. 13th Annual Symposium on Theory of Computing*, pages 238–246, 1981.

[49] S. Mazouz, D. Dahmani, and M. Boukala. Analyzing Smil Documents by Using Time Recursive Petri Nets. In *18th International Conference on Computer Theory and Applications ICCTA'2008*, Egypt, October 2008.

[50] S. Mazouz, D. Dahmani, and K. Kaddouri. A formal approach for the coherence control of smil documents. In *ICTIS'05*, Maroc, June 2005.

[51] S. Mazouz, D. Dahmani, and K. Kaddouri. A formal approach for the coherence control of smil documents. In *International journal of Computer Science and Applications Special Issue on Internet Technologies and services*, volume III issue II, 2006.

[52] M. MENASCHE. Analyse des réseaux de petri temporises et application aux systèmes distribués. In *Thèse de Doc. Ing*, Université Paul Sabatier de Toulouse (France), 1982.

[53] P.M. Merlin. A study of the recoverability of computing systems. In *PhD thesis*, Department of Information and Computer Science, University of California, Irvine, CA, 1974.

[54] M. Pezze and M. Toung. Time petri nets. a primer introduction. september 1998.

[55] A. Pnueli. The temporal logic of programs. In *18th IEEE Symposium on Foundations of Computer Science (FOCS'77)*, pages 46–57. IEEE Computer Society Press, 1977.

[56] G. Ramalingam, J. Song L., Joskowicz, and R. E. Miller. Solving systems of difference constraints incrementally. In *Algorithmica*, volume 23, pages 261–275, 1999.

[57] C. Ramchandani. Analysis of asynchronous concurrent systems by timed Petri nets. In *PhD thesis*, Massachusetts Institute of Technology, Cambridge, MA, 1974. Project MAC Report MAC-TR-120, 1974.

[58] K. Reinhardt. Reachability in petri nets with inhibitor arcs. In *Unpublished manuscript. See www-fs.informatik.uni-tuebingen.de.reinhard*, 1995.

[59] O. H. Roux. Vérification des réseaux de petri temporels et à chronomètres. In *Habilitation à diriger les recherches*, Université de Nantes France, 2005.

[60] A. Seghrouchni and S. Haddad. A formal model for coordinating plans in multiagents systems. In *Proceedings of Intelligent Agents Workshop*. Augusta Technology Ltd, Brooks University, November 1995.

[61] A. Seghrouchni and S. Haddad. A recursive model for distributed planning. In *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS-96)*, pages 307–314. ICMAS-96, AAAI Press, 1996.

[62] R. Valk. On the computational power of extended Petri nets. In *Proceedings of the 7th, Int.Symposium on Mathematical Foundations of Computer Science*, volume 64 of *Lecture Notes Computer Science*, pages 526–535, Zakopane, Poland, September 1978. Springer-Verlag.

[63] R. Valk. Self-modifying nets, a natural extension of Petri nets. In *Proceedings of the 5th, colloquium on Automata, Languages and Programming*, volume 62 of *Lecture Notes Computer Science*, pages 464–476, Udine, Italy, July 1978. Springer-Verlag.

[64] R. Valk. *Object Petri Nets : Using the Nets-within-Nets Paradigm In Book Lectures on Concurrency in Petri Nets*, volume 3098 of *Lecture Notes Computer Science*, pages 819–848. Springer-Verlag, 2004.

[65] Consortium W3C. Smil, recommendation de smil2.0 du w3c http ://www.w3c.org/tr/smil20.