Modélisation et analyse temporelle par réseaux de Petri et logique linéaire

Nicolas Rivière

THÈSE

Préparée au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

en vue de l'obtention du
Doctorat de l'Institut National des Sciences Appliquées de Toulouse

Spécialité : **Systèmes Informatiques**

par
Nicolas RIVIÈRE
Titulaire du Diplôme d'Études Approfondies en Systèmes Informatiques de l'Université Paul Sabatier de Toulouse

Modélisation et analyse temporelle par réseaux de Petri et logique linéaire

Soutenue le 26 Novembre 2003 devant le jury :

Président :	M. DIAZ
Rapporteurs :	J.-P. ELLOY S. HADDAD
Examinateurs :	G. MOTET F. VERNADAT
Directeurs de thèse :	B. PRADIN-CHÉZALVIEL R. VALETTE

« À tous ceux qui m'entourent et qui m'ont soutenu constamment depuis toujours sans porter de jugement sur mes choix même dans les moments de doute. »

« À la confiance et l'amour de mes parents et de mon frère. »

« À Stéphanie et à celle ou celui qui viendra émerveillé nos jours dans peu ... »

Avant-Propos

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes (L.A.A.S.) du C.N.R.S., au sein du groupe Outils et Logiciels pour la Communication (O.L.C.). Je remercie MM. Jean-Claude Laprie et Augustin Martinez, respectivement directeur et directeur-adjoint du L.A.A.S. au moment de mon entrée au laboratoire ainsi que le nouveau directeur M. Malik Ghallab et son directeur-adjoint M. Antonio Muñoz Yague, de m'avoir accueilli dans ce laboratoire et pour avoir mis à ma disposition les ressources nécessaires à l'aboutissement de cette thèse.

Je tiens à remercier tout particulièrement M. Jean-Pierre Elloy professeur à l'Ecole Centrale de Nantes et M. Serge Haddad professeur à l'Université Paris-Dauphine qui ont accepté de prendre du temps pour lire ce manuscrit de thèse et en être les rapporteurs.

Un double remerciement à M. Michel Diaz, qui fut le directeur du groupe de recherche OLC durant mes trois années de thèse, pour m'avoir accueilli dans son équipe et pour avoir accepté de présider le jury de cette thèse.

De même, je remercie M. Gilles Motet professeur à l'Institut National des Sciences Appliquées de Toulouse et M. François Vernadat maître de conférences (habilité à diriger des recherches) à l'Institut National des Sciences Appliquées de Toulouse d'avoir accepté d'examiner ce travail.

Je ne sais pas si des remerciements seront suffisants pour les deux personnes sans qui cette thèse ne serait pas. Je veux parler de mes directeurs de thèse : Mme Brigitte Pradin-Chézalviel professeur à l'Institut Universitaire de Technologie Toulouse 3 et M. Robert Valette directeur de recherche au LAAS du CNRS. En effet, ces travaux de thèse sont la suite de ceux qu'ils ont initiés au laboratoire il y a une dizaine d'années et qui ont été mis en valeur par plusieurs thèses. Je tiens à les remercier pour leurs compétences sans faille, leur soutien et leurs conseils à chaque fois que j'y avais recourt. Et ce, même pendant leurs vacances, ce qui m'a permis de connaître leur havre de paix loin du tumulte de la recherche. Pour résumé, comme dirait mon collègue de bureau Christophe, que je tiens à remercier pour avoir accepté que je mette un peu de bazar dans la quiétude de son

bureau, j'ai eu droit à une qualité de service de bout en bout de leur part.

Je n'oublie pas toutes les personnes que j'ai cotoyées depuis mon stage de maîtrise à aujourd'hui et sur qui j'ai pu compter ; la liste est trop longue pour toutes les citer.

Je remercie les services techniques pour leurs compétences avec une note particulière pour Christian Berty pour son œil avisé des *faenas*.

Merci à Gérard Mouney de m'avoir fait découvrir ce laboratoire et à Patrick Danès de m'avoir persuadé que je pouvais faire une thèse.

Merci à Sarhane pour son partage de l'amitié et des connaissances. Et pour terminer, je remercierai mon ami Yann pour son « tutorat » constant durant ces quatre années. Il m'a gentiment passé le relais de la soutenance que je transmets aujourd'hui à Guillaume qui, j'espère, en fera bon usage très bientôt ...

Table des matières

Table des figures

Introduction générale : contexte et motivations

L'évolution croissante des techniques dans le domaine de l'informatique, que ce soit dans le sens des performances des éléments matériels ou logiciels ou dans le sens de sa facilité de mise en œuvre, a fait que de nombreuses structures (entreprise, établissement scolaire, administration, etc) font appel aux systèmes informatiques. En parallèle, depuis quelques années, nous sommes témoins d'une explosion dans le domaine de la communication en général. La communication est devenue le maître-mot dans le fonctionnement de toutes ces structures mais aussi dans notre quotidien. Beaucoup de méthodes de travail ont été repensées autour de la communication.

Dans le domaine de l'informatique, la communication est souvent mise en œuvre pour faire coopérer les systèmes informatiques entre eux. Les systèmes sont dits *distribués*. Chaque entité du système distribué global peut parfois fonctionner de manière indépendante des autres entités tout en étant amenée à d'autres instants à fournir un service aux autres entités. Les exigences modernes font que ces services sont de plus en plus sollicités en temps réel. Une entité ne possède pas un temps indéfini pour fournir le service qui lui est demandé. Il est souhaitable qu'elle le fournisse dans un temps qui lui est imparti.

Il est apparu deux domaines de recherche importants ces dernières années : le domaine des systèmes « coopératifs » et le domaine des systèmes « temps réel ». Chacun d'eux s'est attaché à trouver des solutions qui permettent de prendre en compte au mieux leur problématique : la coopération des entités distribuées pour le premier et la réalisation de tâches en respectant les échéances temporelles pour le second.

Pour le premier, nous pouvons citer des applications telles que le travail coopératif à l'aide d'ordinateur (CSCW, computer supported cooperative work en anglais), les applications multimédia, l'ingénierie des systèmes distribuée (DSE, distributed systems engineering), le calcul scientifique distribué (super-calculateurs) etc. Pour le second, nous pouvons citer les systèmes embarqués (calculateurs de vols, gestion électronique des automobiles), les systèmes de contrôle des centrales nucléaires, etc.

Grâce aux performances des technologies actuelles du domaine de l'informatique, un nouveau domaine de recherche a pris en forme en englobant les deux cités précédemment : le domaine des systèmes distribués temps réel.

Notre but a été de contribuer à l'élaboration de méthodes d'aide à la conception de systèmes coopératifs en prenant en compte les contraintes temporelles de façon quantitative. Pour cela, nous nous sommes positionné au croisement de différents domaines :

- les automates,
- les réseaux de Petri,
- les graphes d'activités et
- les graphes de contraintes temporelles.

Notre approche consiste à utiliser les réseaux de Petri afin d'analyser les relations de cause à effet entre les activités sans passer par la construction de l'ensemble des états accessibles. Pour cela, nous nous sommes appuyés sur une logique formelle : la logique linéaire de Girard. Ensuite, nous avons considéré le graphe de relations de cause à effet comme un graphe d'activités. Finalement, nous avons exprimé les contraintes temporelles quantitatives à l'aide d'un graphe de contraintes déduit du graphe d'activités.

Organisation du mémoire

Ce mémoire de thèse est organisé autour de deux parties : la première partie porte sur l'analyse *qualitative* des systèmes à événements discrets et la deuxième partie porte sur l'analyse temporelle *quantitative* de ces systèmes.

Le chapitre 1 présente un *rapide* état de l'art des différents modèles, approches et outils utilisés pour faire de l'analyse temporelle qualitative et quantitative de systèmes à événements discrets comportant du parallélisme. En effet, faire un état de l'art complet aurait nécessité un chapitre pour chaque modèle, chaque approche et chaque outil. Dans ce chapitre, nous justifierons les choix retenus pour le reste du mémoire.

Le chapitre 2 présente le modèle retenu, les réseaux de Petri pour représenter des systèmes à événements discrets ainsi qu'une méthode d'analyse, basée sur la logique linéaire, permettant de prouver l'accessibilité d'un état du système modélisé à partir d'un autre état tout en prenant en compte le parallélisme du système sans entrelacement. Nous verrons que cette approche logique permet d'extraire des ordres partiels entre les franchissements de transition par une technique d'annotation.

Dans le chapitre 3, nous présentons une autre technique qui permet aussi d'extraire des ordres partiels entre les franchissements de transition : les processus finis de réseaux de Petri. Nous montrons que les ordres partiels obtenus par l'approche du chapitre 2 sont exactement les ordres partiels associés aux processus finis de réseau de Petri. Par la suite, nous montrons comment la technique d'annotation est utilisée afin de caractériser un seul des ordres partiels possibles entre les franchissements de transition. Cela nous permet d'avoir si besoin une approche compositionnelle. C'est également un point de départ pour faire de l'analyse temporelle quantitative.

Le chapitre 4 entame la deuxième partie du mémoire. Il porte sur les systèmes de contraintes temporelles (TCSP) et sur différentes extensions temporelles des réseaux de

Petri. Nous montrons en effet comment nous pouvons représenter les contraintes temporelles de ces réseaux de Petri sous la forme de systèmes de contraintes temporelles. Nous montrons qu'il est plus naturel dans cette approche de choisir les réseaux de Petri p-temporels comme modèle pour expliciter les contraintes temporelles d'un système. Un exemple avec application numérique est développé en fin de chapitre.

Le chapitre précédent ayant montré qu'il n'était pas possible d'obtenir un seul système de contraintes temporelles de type TCSP à partir d'un ordre partiel dans le cas des réseaux de Petri t-temporels, nous proposons dans le dernier chapitre une autre méthode d'analyse temporelle. Cette méthode, basée sur le calcul de dates *symboliques* directement à l'aide d'arbres de preuves en logique linéaire, permet de caractériser l'accessibilité des marquages à partir de scénarios de réseaux de Petri t-temporels. Elle est appliquée dans un cadre de sémantique faible de ce modèle. Il est ensuite montré que l'utilisation de la sémantique forte permet à certains scénarios d'en invalider d'autres.

Nous terminerons en citant les résultats importants et les perspectives de recherche à ces travaux.

Chapitre 1

Logique, ordre partiel et raisonnement temporel

1.1 Choix du modèle

1.1.1 Approche temporelle dans les systèmes distribués

La prise en compte du temps dans les systèmes distribués est aujourd'hui un problème qui est de plus en plus essentiel du fait de l'utilisation de l'informatique dans des systèmes toujours plus critiques. Cela dit, cette prise en compte est intuitivement liée à l'évolution croissante des technologies de l'information et de la communication. Seule la performance des réseaux de communication actuels permet de développer des applications distribuées dites temps réel.

Le terme temps réel s'applique à une large gamme de systèmes informatiques : systèmes embarqués, systèmes de production, systèmes coopératifs. Ces différents domaines sont souvent caractérisés par les notions suivantes : systèmes temps réel critiques ou non, systèmes temps réel à contraintes strictes ou relatives, systèmes dirigés par le temps ou par les événements, systèmes distribués ou centralisés, systèmes multiprocesseurs ou monoprocesseurs, systèmes tolérant les fautes ou non, etc.

Nous pouvons affirmer que dans tous ces types de systèmes et applications l'ordonnancement de tâches joue un rôle primordial. Il existe deux types d'ordonnancement de tâches temps réel : l'ordonnancement « statique » et l'ordonnancement « dynamique ». Le premier est résolu *hors ligne* et ne prend pas en compte les changements éventuels qui peuvent avoir lieu. Le deuxième par contre se résout pendant le déroulement des activités au cours du temps en fonction des priorités et des durées des activités suivant les algorithmes mis en œuvre. Notre travail se situe dans un contexte dynamique avec analyse hors ligne.

Certes le respect des échéances temporelles est important mais il est aussi important de respecter les contraintes logiques. Ce sont elles qui vont apporter des résultats concernant

les relations de précédence et de synchronisation entre les événements. Que serait une pizza si nous mettions dans un plat le fromage puis le coulis de tomate et enfin la pâte ? Certes nous respecterions les contraintes de temps, si nous en avions, mais nous ne serions pas certains du résultat obtenu ... Est-ce qu'un joueur de rugby arriverait à faire une passe à un partenaire si ce dernier n'est pas près à recevoir le ballon ? Il nous paraît évident que des systèmes impliquant plusieurs entités doivent respecter des contraintes d'ordre et de synchronisation entre les événements avant même de respecter les échéances temporelles qui leur sont imposées afin d'atteindre correctement des objectifs communs.

Il apparaît essentiel d'avoir des méthodes qui permettent de modéliser et analyser les phénomènes temporels dans les systèmes distribués pour pouvoir prouver le respect des contraintes de temps avant le lancement effectif d'une application distribuée temps réel. Les contraintes n'étant pas seulement d'ordre quantitatif comme nous l'avons dit précédemment, il faudra avoir deux approches :

- *qualitative* pour prendre en compte toutes les contraintes d'ordre et de synchronisation entre les événements,
- *quantitative* pour respecter les échéances fixées.

1.1.2 Parallélisme et entrelacement

Les systèmes distribués impliquent la coordination d'activités exécutées par plusieurs entités réparties, plus ou moins distantes. Une partie de ces activités est exécutée localement par les entités de façon autonome. La représentation globale d'un tel système par toutes les activités qui le composent donne un modèle comportant un enchaînement de synchronisations et d'activités parallèles.

Définition 1 (Parallélisme) *Deux activités sont en parallèle si elles s'exécutent simultanément de façon indépendante l'une de l'autre : aucun événement de l'une ne doit précéder un événement de l'autre (pas de contrainte de précédence).*

À partir de là nous distinguons différents types de parallélisme. En logiciel il faut distinguer le parallélisme « vrai » (physique), où des processus au comportement séquentiel (tâches) se déroulent ou s'exécutent simultanément et indépendamment les uns des autres, du parallélisme « logique » où l'exécution a lieu par *entrelacement*.

Cette différence entre parallélisme vrai et logique provient de l'éventuel partage des ressources. Si deux processus sont exécutés simultanément chacun sur une machine différente (deux processeurs différents), nous parlons de parallélisme vrai car rien ne relie les deux processus même pas les ressources nécessaires à leur exécution. Si deux processus sont exécutés sur la même machine (même processeur), nous parlerons de parallélisme logique (pseudo-parallélisme) traité par entrelacement car même si les deux processus sont indépendants ils ne pourront pas être exécutés simultanément du moins de manière physique.

Suivant le parallélisme que nous traitons, deux aspects temporels s'imposent à l'évidence. Dans le parallélisme vrai, les processus sont indépendants et s'exécutent sans aucune notion des uns et des autres : chaque processus s'exécute avec sa propre échelle du temps. Dans le parallélisme logique, les processus s'exécutent en s'entrelaçant, c'est-à-dire que chaque pas de l'évolution du système produit un nouvel état : il n'existe qu'une seule échelle temporelle.

Nous en déduisons qu'en parallélisme vrai, nous travaillons dans un environnement multi-horloges alors qu'en parallélisme logique nous travaillons dans un environnement mono-horloge.

1.1.3 Raisonnement sur les états et sur les événements

Les différentes méthodes d'analyse des systèmes distribués (et plus généralement des systèmes à événements discrets) existantes sont de deux types :

- orientés « états »,
- orientés « événements ».

Les méthodes orientés états permettent de construire le graphe d'états qui représente tous les états accessibles du système modélisé avec les événements permettant de changer d'état. Dans ces méthodes, il faut observer les états du système et analyser comment ils changent. Un des problèmes de ces méthodes est que, en général, le traitement du parallélisme est effectué par l'entrelacement. En effet, nous considérons qu'il y a un seul changement d'état élémentaire à la fois. Même si l'ordre dans lequel les actions doivent être exécutées est respecté, les véritables contraintes de précédence sont cachées car elles sont représentées de la même façon que l'entrelacement de changements d'états parallèles. Elles n'apparaissent pas explicitement dans le graphe d'états, ce qui ne nous permet plus d'extraire des relations de causalité entre les événements. De plus, ces méthodes sont souvent confrontées au problème de l'explosion combinatoire, c'est-à-dire que la construction du graphe d'états peut devenir d'autant plus grande que le système est grand et complexe. La croissance est même souvent exponentielle. L'utilisation de ce graphe n'est donc pas adapté pour vérifier certaines propriétés attendues d'un système. Un désavantage certains pour l'analyse temporelle des systèmes est que les activités ne sont prises en compte que par des états globaux, ce qui ne simplifie pas l'évaluation exacte de la durée d'une suite d'activités car les durées de ces dernières ne peuvent être associées ni à ces états globaux ni aux événements.

Les méthodes orientés événements ne construisent pas le graphe d'états. Elles observent les événements et analysent comment, quand et dans quel ordre ils apparaissent. Ces méthodes-là construisent généralement des graphes de précédence mettant en évidence des relations de causalité. Chaque graphe de précédence décrit un type de comportement donné du système. Ce graphe ne permet par contre de retrouver qu'une partie des états accessibles. Seuls ceux pouvant apparaître lors du comportement représenté pourront être

obtenus. Si nous voulons retrouver le graphe d'états complet, il faudra considérer tous les comportements possibles.

Étant donné que nous ne considérons pas les états globaux du système mais chaque activité indépendamment les unes des autres, nous avons la possibilité d'exprimer le parallélisme vrai du système (qui est perdu lors de la construction du graphe d'états). Nous pouvons plus facilement travailler avec plusieurs horloges.

Analyser les événements permet de connaître les relations d'ordre et de synchronisation existant entre eux. Il est ainsi possible de faire de l'analyse qualitative du temps à partir de ces premiers résultats. À partir de là il devient naturel d'évaluer des durées que nous associons aux activités dont nous pouvons aisément distinguer le début et la fin.

1.1.4 Pourquoi un réseau de Petri ?

Lorsque nous modélisons des systèmes, nous ne cherchons pas seulement à les représenter mais aussi à en obtenir une vision assez abstraite pour pouvoir en déduire certaines propriétés. Dans le domaine des systèmes informatiques, nous nous plaçons dans le cadre des systèmes à événements discrets, c'est-à-dire que tout passage d'un état discret à un autre se fait par l'intermédiaire d'un événement.

Il y a trois notions importantes dans les systèmes à événements discrets : les activités, les événements et les entités (ou objets) impliqués. Suivant l'aspect que nous souhaitons privilégier, nous avons de nombreuses façons de représenter un système à événements discrets. Par exemple :

- les automates finis [26],
- les graphes d'activités,
- les réseaux de Petri [42, 28, 41, 43].

Si nous partons d'un réseau de Petri, l'automate fini est construit à partir d'un marquage initial donné et correspond au graphe des marquages accessibles. Tous les états globaux sont donc énumérés avec, sur les arcs reliant deux états successifs, l'événement reliant le passage d'un état vers un autre. C'est un modèle orienté « état » où les activités sont prises en compte par les états globaux et les événements sont vus comme des changements de ces états ; les entités chargées d'exécuter les activités n'étant pas prises en compte. Les noeuds représentent les états et les arcs les événements.

Définition 2 (Automate fini) *Un automate est défini par un ensemble d'états S, un ensemble fini d'événements E, une relation de transition qui est un sous-ensemble F de $S \times E \times S$ et un état initial $s_0 \in S$. L'évolution de l'automate est une suite de transitions $(s_i, e_j, s_k) \in F$ à partir de s_0.*

Comme exemple de graphe d'activités, nous allons considérer un graphe de précédence entre événements. Les nœuds sont les événements et les arcs les activités. Un graphe de précédence est donc un outil centré sur les événements et les activités. Il est construit

à partir d'un système, d'un ensemble initial d'activités et d'un ensemble fini d'événements. Chaque activité impliquant une relation de précédence entre le nœud source de l'activité (début d'activité) et le nœud destination (fin d'activité), un graphe d'activité décrit une relation d'ordre entre les événements du graphe. Contrairement à un automate, les états globaux ne sont pas représentés. Une activité ne peut se produire que lorsque toutes les activités qui la précèdent sont terminées. De plus, nous considérons les activités indépendamment les unes des autres.

Définition 3 (Graphe de précédence) *Un graphe de précédence est un graphe orienté acyclique dont les sommets sont des événements E et les arcs des activités (ou des relations de précédence) A. E est un ensemble fini d'événements et le sous-ensemble A de $E \times E$ une relation de précédence. Les éléments a_k de A sont donc des paires (e_i, e_j) telles que $e_i \prec e_j$, $e_i \in E$ et $e_j \in E$.*

Les états (ensemble d'activités pouvant être actives simultanément) sont les coupes maximales du graphe de précédence.

Notre choix

Suivant que nous souhaitons privilégier les états globaux ou les événements, les représentations que nous devons utiliser sont différentes. La représentation par automate donne tous les états accessibles d'un système en confondant tous les comportements. Quant à la représentation par graphe de précédence, elle permet de bien caractériser chaque comportement individuellement. Par contre les états ne sont pas explicités et pour pouvoir les obtenir tous, il faut construire tous les graphes de précédence qui sont potentiellement en nombre infini.

Les réseaux de Petri permettent d'obtenir l'une ou l'autre des représentations en fonction du besoin. Ils permettent de prendre en compte explicitement les trois notions des systèmes à événements discrets. Pour cela, les réseaux de Petri possèdent deux types de noeuds : les *places* et les *transitions*. Les places sont associées aux activités et les transitions sont associées aux événements. Les jetons se déplaçant de places en places marquent la (ou les) activité(s) en cours et représentent les entités.

L'approche fondée sur les réseaux de Petri essaie de contourner au mieux le problème de l'explosion combinatoire des états et des événements en évitant une énumération systématique *a priori* de ces derniers. L'ensemble des états accessibles et l'ensemble des comportements possibles sont définis de façon implicite (par *intention*) et non de façon explicite (par *extension*), en les énumérant tous explicitement comme dans le cas d'un automate fini.

De façon similaire, alors qu'une représentation d'un système à événements discrets par un graphe de précédence entre activités nécessite une énumération de tous les événements (débuts et fins d'activités), un réseau de Petri décrit les enchaînements entre les franchissements de transitions (les événements) de façon implicite. Différents scénarios peuvent

être générés à partir d'un seul réseau de Petri et une transition peut être franchie plusieurs fois (et donc correspondre à plusieurs événements) au cours d'un seul scénario.

En plus de cette puissance d'expression par intention, les réseaux de Petri présentent l'avantage d'avoir un aspect multi-horloge. En effet, chaque jeton d'un réseau de Petri peut être vu comme une horloge logique induisant une relation de précédence entre l'événement (un franchissement de transition) qui le produit et l'événement qui le consomme.

Or, si nous voulons avoir la possibilité de représenter le parallélisme vrai, il faut avoir un aspect multi-horloges où chacune d'elles représente une séquence indépendante d'actions du système analysé.

Un automate fini ne peut pas représenter le parallélisme vrai car il représente un entrelacement d'états successifs. Il présente un aspect mono-horloge du point de vue logique. Un produit d'automates finis peut apporter cet aspect multi-horloges car chaque automate possède sa propre horloge logique. Par contre, dans le cas d'un produit d'automates le nombre d'horloges est fixe et constant. Les changements d'états au sein d'un automate se font en ayant l'horloge de l'automate pour référence alors que les communications entre automates impliquent la synchronisation des horloges des divers automates à travers le mécanisme du « rendez-vous ». L'avantage présenté par les réseaux de Petri est qu'une transition est toujours une transition. Qu'elle ait une seule place en entrée et en sortie (événement interne à un automate) ou plusieurs places en entrée et/ou en sortie (communication), le mécanisme est toujours le même.

1.1.5 Pourquoi les réseaux de Petri avec du temps ?

Comme nous l'avons dit au début de cette section, l'analyse des systèmes distribués implique de vérifier le respect des contraintes logiques de synchronisation mais aussi d'échéances temporelles quantitatives.

Il existe plusieurs extensions des réseaux de Petri qui prennent en compte les contraintes quantitatives : le temps associé aux places, le temps associé aux transitions et le temps associé aux arcs. Ces contraintes temporelles quantitatives sont des contraintes supplémentaires que doivent vérifier les scénarios de franchissements de transitions. Elles n'ont pas d'influence sur la spécification des contraintes qualitatives.

Pour pouvoir expliciter les contraintes temporelles quantitatives entre les événements possibles, nous devons d'abord générer les scénarios de franchissements de transitions. L'approche fondée sur les réseaux de Petri permet cette génération car les transitions sont associées aux événements. Nous obtenons une représentation explicite de la relation de *causalité* entre les événements et les contraintes quantitatives viennent se greffer sur cette structure.

Notre choix d'aborder le temps dans les réseaux de Petri en deux étapes peut paraître restrictif dans le sens où nous nous donnons la possibilité de manipuler des durées uniquement si les contraintes de relation d'ordre sont vérifiées. Mais d'un autre côté cela nous permet de maîtriser plus facilement la résolution des contraintes. Ce qui motive

notre choix vis-à-vis des réseaux de Petri est la possibilité qu'ils offrent de pouvoir traiter des domaines où les notions d'événements et d'évolutions simultanées sont importantes. L'apport du temps n'est qu'un plus permettant de caractériser de manière quantitative ces évolutions.

Une démarche analogue pourrait-elle être développée à partir des automates ?

Un automate temporisé [1, 2] est une extension de l'automate ordinaire auquel sont associés un ensemble fini d'horloges, des conditions et des actions temporelles concernant les horloges sur les arcs. Si nous construisons un automate temporisé nous avons un ensemble fini d'horloges qui sont déclenchées à l'état initial de l'automate et qui peuvent être réinitialisées. Nous n'avons pas de parallélisme vrai explicite. Ces horloges, quantitatives, ne sont pas vues comme des composantes de la représentation de l'état, alors que dans le cas d'un réseau de Petri p-temporel [30, 29], les jetons correspondent à des horloges logiques pour l'approche qualitative puis à des horloges physiques pour l'approche quantitative.

La composition des automates introduit une difficulté supplémentaire. La notion de rendez-vous raté n'est pas gérée dans les automates temporisés classiques (*time-lock*) mais elle l'est par évitement dans les automates temporisés à échéance. La complexité due à l'explosion de la construction des états accessibles se combine ici avec la complexité de raisonner à la fois sur les états et sur les contraintes temporelles. L'automate obtenu ne permet pas d'extraire simplement les contraintes temporelles quantitatives entre les événements possibles.

1.1.6 Application du modèle

Si un système distribué est modélisé par un réseau de Petri ou un ensemble d'automates, c'est pour pouvoir vérifier par des méthodes mathématiques le comportement du système représenté. Cela permet de déduire certaines propriétés. La propriété qui nous intéresse dans notre travail est l'accessibilité d'un état cible à partir d'un état source. C'est la vérification de cette accessibilité qui nous permettra d'affirmer qu'un état cible est accessible depuis un état source tout en respectant l'ordre imposé par les contraintes logiques entre les événements de la séquence. Cela nous permet d'assurer qu'un ensemble de documents multimédias a été correctement présenté, ou bien qu'un ensemble de tâches a été correctement exécuté dans un temps imparti.

1.2 Vérification temporelle orientée « états »

1.2.1 Deux méthodes de base en vérification

Il existe deux méthodes de base en vérification de systèmes informatiques :

- la preuve de théorème (theorem proving),
- la vérification par modèle (model checking).

Le principe de la preuve de théorème est que la propriété à montrer peut être prouvée formellement (par un mécanisme déductif ou par un système de réécriture) à partir d'un ensemble d'axiomes ou d'un ensemble d'hypothèses décrivant le système étudié. La preuve de théorème permet de raisonner sur les systèmes à états infinis (réseaux de Petri non bornés) car lorsque le théorème est prouvé, il est vrai pour tout système incluant le système pour lequel il a été prouvé quel que soit sa taille (monotonie de la logique).

La vérification par modèle impose quant à elle l'énumération de tous les états du système étudié. La vérification faite, elle n'est valable que pour un modèle donné. Si une modification est apportée au modèle, il faut reconstruire tous les états pour pouvoir de nouveau affirmer que la propriété vérifiée est vraie. La question que nous nous posons est « Est-ce qu'un modèle satisfait une formule ? »

Notre approche, fondée sur la preuve de théorème à l'aide du calcul des séquents, sera détaillée dans les chapitres suivants. Nous allons, ici, donner les grandes lignes de la vérification par modèle.

1.2.2 La vérification par modèle

En partant de l'hypothèse qu'un système possède un espace d'états fini, la vérification d'une propriété consiste à vérifier qu'aucune des exécutions possibles dans cet espace d'états n'invalide la propriété à prouver. Les diverses techniques qui ont été développées sont basées sur une représentation discrète du temps car l'exécution des réseaux de Petri est réalisée en temps discret ou bien par un recouvrement de l'ensemble des états accessibles par un ensemble de classes, de régions ou de zones. Les différentes étapes d'exécution d'un réseau de Petri sont les états accessibles d'un système. Quand il n'y a pas de contraintes temporelles quantitatives, le travail est effectué directement sur le graphe des marquages accessibles.

La propriété à vérifier est traduite sous la forme d'une formule logique et il faut vérifier que l'ensemble des états accessibles est bien un modèle (au sens logique du terme) de la formule.

Le système à états finis est donc représenté par un graphe de transitions étiqueté où les labels d'un état sont des propositions logiques atomiques. Les propriétés du système sont exprimées sous la forme de formules en logique temporelle. La vérification consiste alors à parcourir le système de transitions et à vérifier qu'il satisfait la formule représentant une propriété : le système est un modèle de la propriété. La question est : *« Est-ce qu'un modèle M satisfait une formule logique φ pour tous les états accessibles ? »*

Effectuer du raisonnement temporel n'est pas simple en logique. En effet, il faut être capable de prendre en compte les variations des valeurs (vérités) des propositions dans le temps. La logique modale contourne ce problème en considérant un ensemble de mondes logiques : une vérité vraie ne peut devenir fausse dans un même monde (*monotonie*), mais elle peut l'être dans un autre monde. Les logiques temporelles sont des extensions de la logique modale et sont donc capables de prendre en compte ces changements de

valeurs des propositions à un instant donné ou pendant une certaine durée. Parmi les différentes logiques qui correspondent à différentes vues du temps, nous ne considérons que les logiques basées sur une représentation discrète et non quantitative du temps :

- la logique temporelle arborescente CTL [13],
- la logique temporelle linéaire LTL [44].

CTL est une logique où le temps est arborescent, c'est-à-dire qu'elle permet de travailler sur la représentation arborescente des processus. En effet, il peut exister plusieurs successeurs à un état. En CTL il y a deux types de formules : les formules d'états qui sont évaluées pour un instant donné et les formules de chemin qui sont évaluées le long d'un chemin. Nous avons la possibilité dans cette logique de vérifier pour chaque type si les formules sont vraies pour un chemin ou pour tous les chemins.

En LTL, chaque état ne possède qu'un seul successeur, nous travaillons sur une séquence d'exécution donnée. Le temps croît de manière linéaire à partir d'un instant initial. Il n'y a que des formules de chemins.

CTL et LTL sont incomparables dans le sens où certaines formules d'une logique ne peuvent s'exprimer dans l'autre et inversement. Cependant, il n'est pas possible d'exprimer le non-déterminisme des systèmes parallèles avec une logique à temps linéaire. C'est pour cela que les logiques à temps arborescent telles que CTL sont souvent meilleures pour analyser les systèmes réactifs. Par contre, une logique à temps arborescent n'est pas adaptée pour exprimer des propriétés d'*équité* bien qu'une extension appelée Fair-CTL [14] ait été développée pour prendre en compte ces contraintes. Les logiques à temps linéaire telles que LTL sont préférées pour analyser des propriétés sur des chemins.

1.2.3 Extension de la vérification par modèle : prise en compte du temps

La prise en compte du temps quantitatif dans les réseaux de Petri se fait en associant des contraintes temporelles aux éléments du réseau : soit aux places, soit aux transitions, soit aux arcs. Des outils ont été développés afin de pouvoir traiter ce temps et vérifier des propriétés.

Comme nous l'avons dit plus haut, il faut recouvrir l'ensemble infini (si le temps est dense) en un ensemble fini de classes [36, 3]. Cette approche permet ensuite une analyse semblable à la méthode du graphe des marquages utilisée pour l'analyse des réseaux de Petri ordinaires.

L'ensemble des états d'un réseau temporel est infini étant donné que les transitions peuvent être tirées à tout instant dans leur intervalle de tir qui appartient à l'ensemble des réels. Le principe de la méthode consiste à regrouper les états ayant même marquage et des domaines de tir sensiblement égaux c'est-à-dire ne variant que par un décalage de certaines composantes et une troncature. Le nombre de classes d'états devient alors fini si le réseau est borné. Les contraintes temporelles sont représentées sous la forme

d'inéquations. Le graphe des classes est la représentation graphique de ces classes d'états où les nœuds sont les classes (paire état-systèmes d'inéquations) et les arcs reliant deux classes disctinctes portent le nom de la transition permettant de changer de classe.

Ce graphe des classes peut ensuite être exploité comme le graphe des marquages accessibles pour vérifier des propriétés exprimées dans des logiques temporelles comme CTL ou LTL. Cela permet ainsi une prise en compte de contraintes temporelles quantitatives lors de la vérification de propriétés. Toutefois pour pouvoir faire intervenir des considérations temporelles quantitatives dans l'expression des propriétés, il faut faire appel à d'autres logiques que CTL ou LTL. De plus, dans le graphe des classes, la durée de la séquence de franchissement permettant de passer d'un marquage à un autre ne peut être caractérisée de manière précise. En effet, à chaque franchissement de transition une imprécision est introduite.

Comme nous l'avons vu précédemment, il est également possible d'associer un ensemble d'horloges à un automate pour obtenir un automate temporisé. Tout comme précédemment dans le cas des réseaux de Petri temporels, l'ensemble des états est infini, mais il peut être recouvert par un ensemble de régions ou de zones. Il est alors possible de vérifier des propriétés par modèle.

Nous venons de voir, bien rapidement, un ensemble d'approches permettant la modélisation et la vérification de propriétés de systèmes distribués. Nous avons dit que notre préférence allait aux réseaux de Petri à cause d'une prise en compte du parallélisme vrai plus simple et plus systématique (une transition décrit aussi bien un changement d'état interne qu'une communication entre entités séquentielles). De plus, les contraintes temporelles quantitatives sont spécifiées de façon cohérente avec le modèle car les horloges sont associées soit aux jetons, soit aux transitions sensibilisées. Toutefois, ces divers points ne seraient pas des avantages si nous ne pouvions pas analyser les ordres partiels existants entre les franchissements des transitions et si toutes les méthodes d'analyse devaient être basées sur le graphe des marquages accessibles.

1.3 Ordre partiel et techniques de dépliage

Le traitement du parallélisme d'un réseau de Petri s'effectue en effet souvent par l'intermédiaire de l'entrelacement même si des transitions du réseau sont structurellement en parallèle. Nous avons vu que cette méthode engendre une explosion combinatoire du nombre d'états lors de la construction du graphe d'accessibilité. Les séquences contenant ces transitions-là se différencient seulement par l'ordre de franchissement des transitions, ce qui fait apparaître des redondances qui pourraient être évitées afin de limiter la taille du graphe.

1.3.1 Méthodes basées sur la représentation de l'ordre partiel d'un réseau de Petri

Les méthodes basées sur la limitation de la taille du graphe d'accessibilité par élimination des redondances dans les séquences sont appelées méthodes d'*ordre partiel* car les relations de dépendance et d'indépendance caractérisent l'ordre partiel des franchissements de transitions. Deux voies distinctes s'ouvrent :

- la réduction du graphe d'accessibilité par élimination des séquences redondantes pour la représentation de traces,
- la représentation directe de l'ordre partiel des tirs de transitions.

C'est A. Mazurkiewicz qui a donné les bases d'une nouvelle théorie en définissant la notion de *traces* [34].

Définition 4 (Trace) *Une trace représente un ensemble de séquences de tirs tel que deux séquences d'une classe peuvent être obtenues l'une de l'autre en permutant sucessivement les occurrences de tir adjacentes et indépendantes.*

Les séquences partent du même état initial et arrivent au même état final. Elles définissent un ordre partiel entre les franchissements de transition.

Cette approche permet de réduire significativement la taille du graphe d'accessibilité, ce qui est une bonne chose pour vérifier des propriétés. Elle est efficace dans la préservation de certaines propriétés et en particulier des blocages. Il existe diverses techniques basées sur cette approche que nous ne détaillerons pas ici car les ordres partiels ne sont pas représentés explicitement :

- la recherche d'ensembles persistants [62],
- la recherche d'ensembles dormants [22],
- les graphes de pas couvrants [58],
- les graphes de pas persistants, combinaison de la première et de la troisième [49].

Des travaux basés sur ces techniques ont été menés afin de diminuer l'explosion combinatoire tout en prenant en compte le parallélisme vrai.

1.3.2 Une deuxième approche : la représentation de l'ordre partiel des franchissements de transition

La représentation directe des tirs de transitions se base sur les notions de parallélisme/concurrence et de conflits dans les réseaux places-transitions. Une opération de traduction est effectuée à partir d'un réseau de Petri vers un graphe étiqueté où chaque transition et chaque place peut être repésentée plusieurs fois : l'opération est un *dépliage* et le graphe est un *processus de branchement* [4, 16, 35].

Plutôt que de représenter des entrelacements de tirs de transitions, l'idée est de représenter les tirs de transitions indépendantes par des transitions indépendantes. Comme

pour le graphe d'accessibilité, le but est de caractériser tous les marquages accessibles à partir d'un marquage initial. Il est aussi de mettre en évidence des relations de causalité et d'indépendance entre certains franchissements de transition. Par contre, les dépliages ne représentent pas tous les ordres partiels possibles.

Parmi les représentations basées sur les dépliages, citons les réseaux causaux étiquetés communément appelés processus de réseau de Petri [38]. Un processus est un réseau où les conflits sont résolus car chaque place possède au plus une transition en entrée et une transition en sortie : chaque paire de transitions d'un processus est soit en dépendance causale soit indépendante.

Un processus de réseau de Petri décrit donc un ordre partiel entre un ensemble d'événements (les franchissements de transition) qui sont tous nécessaires , c.-à-d. qu'ils peuvent tous se produire au cours d'un même scénario. Deux événements non reliés par une relation de causalité sont donc parallèles (parallélisme vrai).

Comme nous avons une structure du même type que les réseaux de Petri avec les événements qui sont associés aux transitions et les activités aux places, l'approche est orientée événements. Les états sont des coupes maximales du graphe représentant le processus. Nous détaillerons ces divers points au chapitre 3 et nous montrerons la relation existante entre ces travaux et notre approche.

Les processus de réseau de Petri ont été étendus pour prendre en compte le temps de façon qualitative [61]. Toutefois ces travaux ne font pas le lien avec les approches permettant l'analyse d'un ensemble de contraintes temporelles quantitatives. Ils ne répondent pas à la question du calcul effectif des dates de franchissement des transitions permettant le respect des contraintes.

1.4 Systèmes de contraintes temporelles

1.4.1 Les représentations de type TCSP et STP

Pour répondre au problème mentionné ci-dessus, un certain nombre d'outils pour la résolution de contraintes temporelles dans les systèmes à événements discrets ont été développés dans un cadre formel : les problèmes à satisfaction de contraintes temporelles, communément appelés par leur abréviation anglaise *TCSP* (temporal constraint satisfaction problems) [8]. Un TCSP est une représentation d'un ensemble de contraintes temporelles dérivée des CSP (constraint satisfaction problems) [39].

Définition 5 (Problème à satisfaction de contraintes temporelles) *Un problème à satisfaction de contraintes temporelles est un couple (X, C) où l'ensemble X des variables prennent leurs valeurs dans un ensemble de domaines temporels (intervalle de temps) et l'ensemble C des contraintes représente des relations temporelles logiques ou numériques entre les variables. Toutes les contraintes des TCSP sont binaires [55], c'est-à-dire qu'elles impliquent deux variables.*

Les variables et les contraintes d'un TCSP peuvent être représentées sous forme graphique. Les nœuds du graphe sont les variables et les arcs sont les contraintes. Les domaines des variables et des contraintes sont des intervalles de $\mathbb{R}$. Cela correspond bien à notre attente dans le sens où les réseaux de Petri temporels manipulent des domaines de valeurs continues eux-aussi et où les dates de franchissement sont des réels.

Les problèmes temporels simples (STP, simple temporal problem) [8] sont un cas particulier intéressant de TCSP. Les domaines des variables sont des intervalles (et non des unions d'intervalles) et les contraintes sont binaires et de la forme (X_i et X_j sont des variables, l'événement associé à X_i précède celui associé à X_j et la durée entre ces événements appartient à l'intervalle $[a_{ij}, b_{ij}]$) : $a_{ij} \leq X_j - X_i \leq b_{ij}$. La contrainte s'exprime aussi sous la forme de deux inéquations linéaires :

$$X_j - X_i \leq b_{ij}$$
$$X_i - X_j \leq -a_{ij}$$

Nous aurons donc deux arcs orientés, l'un reliant X_i à X_j et de longueur b_{ij} et l'autre reliant X_j à X_i et de longueur $-a_{ij}$.

Un réseau de contraintes représentant un STP s'exprime donc sous la forme d'un système d'inéquations linéaires. Ce système peut se résoudre par des méthodes connues du monde de la recherche opérationnelle en un temps polynômial. De puissants mécanismes de propagation de contraintes [55] ont également été définis. Ils permettent une délimitation précise des domaines des variables.

1.4.2 Principales utilisations

Les domaines d'utilisation de ces méthodes de résolution de contraintes temporelles sont la recherche opérationnelle, la plannification en robotique et l'ordonnancement. Dans tous ces domaines, il est nécessaire de pouvoir calculer les durées des différentes activités afin de connaître la durée d'exécution d'un scénario donné pour un robot autonome ou la durée d'un cycle d'exécution d'une chaîne de fabrication. Il ne s'agit pas seulement de montrer que nous pouvons, ou que nous ne pouvons pas, vérifier simultanément toutes les contraintes, ce qui correspondrait à la vérification d'une propriété temporelle pour un système distribué. Il s'agit également de trouver une bonne solution c'est-à-dire d'affecter une valeur à chaque variable de telle sorte que les contraintes soient vérifiées et que la valeur d'un critère soit satisfaisante. L'objectif ultime, en général non accessible dans le cas des systèmes complexes, étant de trouver la solution optimale.

Toutefois, ces diverses communautés n'expriment pas les contraintes de la même manière. La communauté de l'intelligence artificielle exprime les contraintes sous la forme présentée ci-dessus. La communauté de l'ordonnancement inverse le point de vue et ex-

prime une contrainte sous la forme suivante :

$$X_j - X_i \leq b_{ij}$$
$$X_j - X_i \geq a_{ij}$$

Ce qui va donner un arc reliant X_i à X_j de longueur a_{ij} et un arc reliant X_j à X_i de longueur $-b_{ij}$. Dans le cadre de l'approche précédente, la non satisfaisabilité de l'ensemble des contraintes se traduit par l'existence d'un circuit (orienté) de longueur négative alors qu'avec cette nouvelle représentation ce sera celle d'un circuit de longueur positive.

Revenons à l'approche précédente (processus de réseau de Petri). Supposons que les transitions t_i et t_j soient franchies une fois dans un processus et que t_i soit reliée par une relation de causalité à t_j. Nous en déduisons que t_i précède t_j dans l'ordre partiel et donc que $0 \leq X_j - X_i$. Nous voyons que la relation de précédence se traduit par une contrainte temporelle sur les dates de franchissement. Dans ce cadre, la relation proposée par la communauté travaillant sur l'ordonnancement est plus naturelle car l'arc du graphe de contrainte est orienté de la même manière que le chemin exprimant la causalité dans le processus.

Une faiblesse de l'approche fondée sur les processus est que la notion de causalité sur laquelle est fondée la construction des ordres partiels ne correspond pas à celle communément admise en logique classique. Le fait que la proposition A soit la cause de B ou de C n'implique pas que B et C ne puissent pas être simultanément vrais. Or la vérification (ou la preuve) de propriétés tout comme les mécanismes de propagation de contraintes sont fondés sur la logique. C'est pourquoi il est important d'étudier les travaux associant les réseaux de Petri, la logique formelle et la notion d'ordre partiel en dehors de l'approche fondée sur les concepts de dépliage et de processus. Les premiers travaux qui ont associé la logique, les ordres partiels et les réseaux de Petri ont été réalisés par V. Gehlot au début des années 1990. Ils sont fondés sur la logique linéaire et nous allons les détailler maintenant.

1.5 La logique linéaire et les réseaux de Petri

1.5.1 Origine de la logique linéaire

La logique linéaire a été introduite pour la première fois par J.Y. Girard en 1987 [20] comme une variante de la logique propositionnelle. Girard a choisi le calcul des séquents introduit par Gentzen en 1934 pour définir la logique linéaire et conduire les preuves. Cette logique est *non monotone*, elle est capable de représenter le changement d'état d'un système ce qui est un gros changement par rapport à la logique classique seulement capable de traiter de vérités éternelles. La caractéristique importante de cette logique est de pouvoir distinguer deux exemplaires d'une ressource d'un seul exemplaire : pour cela, il a fallu abandonner les règles d'*affaiblissement* et de *contraction* du calcul des séquents

de la logique classique. Cela a abouti à une réécriture de la conjonction, de la disjonction et de l'implication.

Les propositions de la logique linéaire sont vues comme des ressources : une déduction linéaire *consomme* les propositions qui sont les hypothèses et *produit* les propositions qui sont la conclusion. Une proposition doit être autant de fois disponible dans les hypothèses qu'elle doit être consommée et elle doit apparaître autant de fois dans la conclusion qu'elle aura été produite. La preuve d'un séquent en logique linéaire peut être vue comme un ensemble d'actions sur des ressources. L'observation de ces actions donne une impression de dynamisme et sous-tend donc une dimension temporelle : des relations de causalité et de précédence apparaissent.

1.5.2 Analogie entre réseau de Petri et logique linéaire

Depuis 1987, date des premières publications sur la logique linéaire, diverses personnes ont effectuées des travaux montrant un lien entre les réseaux de Petri et la logique linéaire. En effet, la capacité de la logique linéaire à raisonner sur des ressources (dénombrement, consommation, production, dynamisme) nous amène à faire un rapprochement certain avec l'étude des réseaux de Petri.

À partir des différents travaux sur les liens entre les réseaux de Petri et la logique linéaire, nous pouvons dire qu'il y a eu trois approches distinctes :

- une approche « réseau de Petri comme modèle pour la logique linéaire »,
- une approche « structurelle »,
- une approche « réseau ».

Approche réseau de Petri comme modèle pour la logique linéaire

Une première approche sur les relations entre la logique linéaire et les réseaux de Petri totalement différente des deux autres a été définie par U. Engberg et G. Winskel [15]. Les deux autres approches sont syntaxiques alors que celle-ci est sémantique. Ils interprètent le réseau de Petri comme un modèle pour la logique linéaire. Leur but s'apparente à de la vérification sémantique de modèles.

Ils donnent des résultats sur l'accessibilité en logique intuitionniste et souhaitent apporter des résultats de non-accessibilité dans les réseaux. L'un de leurs objectifs est la caractérisation de la vivacité d'un réseau de Petri à l'aide de la logique linéaire. Toutefois il y a un problème dans leur approche car leur interprétation de la négation linéaire n'est pas satisfaisante. Leurs résultats sur la non-accessibilité sont prouvables sémantiquement mais pas syntaxiquement.

Approche structurelle

Ce sont C.A. Gunter et V. Gehlot [24, 19] qui ont effectué les travaux les plus significatifs dans cette approche. La non-monotonie de la logique linéaire ne doit pas pour autant nous empêcher de considérer des vérités éternelles qui seront vues comme des axiomes. Si nous nous focalisons sur la structure d'un réseau, nous remarquons que, effectivement, des vérités éternelles peuvent être établies. Par exemple, l'existence d'une transition n'est jamais remise en cause par l'évolution du marquage. C'est ce qui justifie le choix de C.A. Gunter et V. Gehlot de représenter les transitions par des axiomes écrits sous la forme de séquents.

L'ensemble de ces séquents constitue un ensemble d'axiomes propres qui sont rajoutés à la logique. D'une manière générale, pour un réseau de Petri donné nous associons :

- un jeton dans une place p à un atome propositionnel P,
- une transition à un séquent axiome propre $M_1 \vdash M_2$ où M_1 et M_2 sont les formules des marquages d'entrée et de sortie de la transition.

Tout séquent de la forme $M_1 \vdash M_2$ correspond à une accessibilité de l'état M_2 depuis l'état M_1. Nous obtenons une réécriture du réseau sous forme logique.

Le but de leur travail a été ensuite de trouver des stratégies de preuve, par la construction d'arbres, afin de caractériser le parallélisme du réseau considéré. Pour construire une preuve d'accessibilité dans le cadre de l'approche structurelle, il est inévitable d'utiliser la règle de coupure *Cut*. En effet, l'ajout d'axiomes fait sortir du cadre strict de la logique linéaire et la règle de coupure cesse d'être redondante pour le calcul des séquents. Seule cette règle permet de mettre en séquence deux transitions. Par exemple, elle permet de prouver l'accessibilité de M_2 à partir de M_1 s'il existe une transition $M_1 \vdash M'$ et une transition $M' \vdash M_2$. Ceci nous donne un ordre de franchissement des transitions. Toutefois, l'arbre de preuve généré n'est pas unique. Les auteurs ont travaillé sur des règles de réécriture des arbres de preuve afin de générer un nouvel arbre qui caractériserait un parallélisme « maximal ».

Un défaut de cette approche est l'introduction des axiomes propres correspondant aux transitions du réseau. Un autre défaut est que malheureusement il n'est possible de caractériser que le parallélisme structurel car le marquage n'est pas pris en compte explicitement dans la traduction du séquent d'accessibilité. Si des jetons sont ajoutés en plus des jetons strictement nécessaires aux franchissements, les résultats quant à l'accessibilité resteront vrais. Cependant, les résultats sur les ordres partiels ne sont plus forcément justes car les jetons supplémentaires peuvent introduire de nouvelles relations de parallélisme. Si une transition est sensibilisée dès l'état initial, elle peut être franchie indépendamment des autres.

Pour faire de une analyse temporelle précise, il est indispensable de caractériser le parallélisme « dynamique » (avec marquage) en plus du parallélisme structurel. C'est le seul moyen d'avoir des ordres partiels corrects.

Approche réseau

Parallèlement aux travaux menés sur l'approche structurelle, C. Brown initia une approche différente qui consiste à traduire tout un réseau de Petri par une formule [7]. Pour cela, elle représente chaque transition par une formule implicative qui prend la signification de franchissement de transition. Une transition est en effet représentée par une formule pouvant être utilisée un nombre de fois non spécifié dans la preuve. C. Brown a introduit la notion de marquage en le représentant par une conjonction multiplicative d'atomes propositionnels. Sa formule de logique linéaire composé d'un marquage M et d'une conjonction multiplicative de formules implicatives est la traduction d'un réseau de Petri *marqué*.

F. Girault [21] et L.A. Künzle [32] reprirent la notion de marquage représenté par une conjonction d'atomes et la représentation d'un franchissement de transition par une formule implicative de logique linéaire. Par contre, au lieu de représenter le réseau de Petri marqué par un séquent, ils choisirent d'utiliser les séquents pour définir des problèmes d'accessibilité spécifiques entre deux marquages, comme C.A. Gunter et V. Gehlot l'avaient fait dans le cadre de l'approche structurelle. Simplement, ce séquent doit comprendre la liste des franchissements nécessaires. Le fait de ne plus avoir d'axiomes propres rajoutés permet de rester dans le cadre strict de la logique linéaire et de ne plus avoir nécessairement à utiliser la règle de coupure lors de la construction de preuves.

Une preuve d'accessibilité est alors définie par :

- un marquage initial,
- une liste de franchissements de transitions,
- un marquage final.

En prenant en compte le marquage d'un réseau, il alors possible de mieux caractériser le parallélisme qu'il soit structurel ou dynamique.

1.5.3 Discussion

Le travail développé par V. Gehlot n'a permis que de caractériser le parallélisme structurel car il manquait la notion de marquage courant. Dans cette approche, les séquents ne traduisaient qu'une relation d'accessibilité entre deux marquages sans donner d'indication sur les franchissements de transition amenant au marquage final.

L.A. Künzle a repris les travaux de V. Gehlot et, à l'aide de l'approche avec marquage développée conjointement avec F. Girault, a caractérisé ce qu'il appelle le parallélisme dynamique. Il a dû pour cela introduire deux règles de calcul des séquents SEQ et SYNC (voir [32]) exprimant respectivement l'exécution en séquence et l'exécution en parallèle de deux scénarios de franchissements de transitions.

La construction d'une preuve caractérise un scénario en donnant l'ordre partiel des franchissements de transition. Cependant, les relations SEQ et SYNC ne permettent d'extraire que des ordres partiels de type « série-parallèle ». Cette méthode a permis d'évaluer

le temps d'exécution des scénarios générés. Toutefois cette évaluation n'est correcte que si l'ordre partiel entre les franchissements peut se mettre sous une forme « série-parallèle ». Dans le cas contraire, des relations de précédence parasites sont introduites et elles peuvent fausser le calcul.

Un autre point important à souligner est le fait que l'approche est compositionnelle. La composition de deux sous-scénarios par les deux relations est possible et une preuve valable pour un scénario reste valable après composition. Ce résultat est valable dans le cas structurel mais aussi dans le cas dynamique.

Par la suite, B. Pradin-Chezalviel et R. Valette se sont attachés à rechercher les liens de causalité entre les différents franchissements de transition sans utiliser les deux règles introduites par L.A. Künzle et en utilisant des preuves sans la règle de coupure [46]. De cette façon les relations de précédence parasites n'apparaissent plus et il est posssible d'appuyer un calcul symbolique sur l'arbre de preuves pour calculer une durée des scénarios sous une forme algébrique qui est cette fois exacte. L'inconvénient est que l'ordre partiel entre les franchissements de transition n'est plus explicité.

1.6 Conclusion

Nous avons vu tout au long de ce chapitre en examinant différents modèles, différentes approches et différents outils que l'utilisation des réseaux de Petri avec du temps présentait le meilleur compromis si le but était l'analyse qualitative et quantitative de scénarios dans un système distribué.

Pourquoi ? Un réseau de Petri est le modèle abstrait le mieux approprié pour représenter des systèmes dynamiques sans faire de différence entre un acteur et les ressources nécessaires aux actions. Il est possible d'avoir une approche orientée activité comme une approche orientée événement suivant les propriétés que nous souhaitons mettre en évidence et ce indépendamment de l'information temporelle quantitative apportée par les extensions temporelles des réseaux de Petri.

Nous avons vu qu'il était possible de développer une approche fondée sur les événements avec une prise en compte correcte du parallélisme vrai par l'intermédiaire de la notion de processus de réseau de Petri basée sur un dépliage du graphe. Nous avons vu, par contre, que si le but était non pas la vérification d'une propriété mais plutôt le choix des dates de franchissement des transitions pour obtenir un certain comportement, alors que les graphes de contraintes temporelles (STP) étaient l'outil le plus adapté. Enfin nous avons vu que la logique linéaire permettait également l'analyse qualitative (ordre partiel) et quantitative de sénarios menant d'un marquage à un autre.

Le but de notre travail a été de développer ce dernier point et de le relier aux autres. En effet il nous semble que la notion de causalité en logique linéaire est au cœur de la notion de relation de précédence et qu'elle peut permettre de passer d'un temps qualitatif fondé sur le parallélisme vrai à un temps quantitatif avec une grande cohérence. C'est ce

que nous allons présenter maintenant.

Chapitre 2

Accessibilité et logique linéaire

2.1 Introduction à la logique linéaire

Une façon classique de traiter un problème d'accessibilité dans un réseau de Petri passe par la construction du graphe d'accessibilité. Mais cette approche n'est pas intéressante lorsque nous souhaitons conserver le parallélisme vrai car le graphe est construit par entrelacement. Nous nous retrouvons avec une représentation graphique purement orientée états énumérant tous les marquages accessibles entre un état initial et final.

Si l'on ne souhaite pas énumérer tous les états mais nous souhaitons avoir une approche orientée événements pour pouvoir conserver le parallélisme du réseau donc sans traiter l'entrelacement. Nous avons vu au chapitre précédent qu'il existait une méthode logique initiée par F. Girault et L.A. Künzle et améliorée par B. Pradin-Chezalviel et R. Valette qui répond justement à cette attente.

Cette méthode permet de traduire un problème d'accessibilité dans un réseau de Petri en une preuve de logique linéaire, logique en phase avec le dynamisme des réseaux car elle manipule les propositions comme des ressources : nous parlons de production/consommation de ressources. Pour cela, J.Y Girard a dû abandonner les règles d'affaiblissement et de contraction du calcul des séquents de la logique classique qui entrainaient l'apparition et la disparition de formules, ce qui a imposé une redéfinition des connecteurs logiques.

La logique linéaire est composée de trois familles de connecteurs qui sont :

- les connecteurs binaires *multiplicatifs*,
- les connecteurs binaires *additifs*,
- les connecteurs unaires *exponentiels*.

Le passage des connecteurs classiques aux connecteurs multiplicatifs linéaires s'effectue par le rejet de la propriété d'idempotence de la conjonction et de la disjonction classiques, propriété fondamentalement liée aux deux règles classiques énoncées plus haut.

Les connecteurs linéaires ont été définis directement dans le cadre du calcul des séquents. De manière formelle, un *séquent* est une expression logique constituée de deux membres, chacun étant une suite de formules propositionnelles, acceptant implicitement toute substitution et ce, que nous soyons dans un cas de logique classique ou linéaire. Un séquent s'écrit sous la forme :

$$A, B, \ldots \vdash E, F, \ldots$$

où le *méta-connecteur* $\vdash$ sépare le membre gauche du membre droit. La virgule qui est un *méta-connecteur* n'a pas le même sens suivant qu'elle est utilisée à droite ou à gauche : à gauche elle représente la conjonction des formules qu'elle sépare alors qu'à droite elle représente la disjonction des conclusions. Nous pouvons l'interpréter de la manière suivante : à partir d'un ensemble d'hypothèses, nous pouvons déduire plusieurs conclusions. Soit, le membre gauche contient les prémisses qui vont permettre de déduire les conclusions contenues dans le membre droit.

Pour exprimer le sens d'un séquent en logique linéaire au sens des ressources, je ne ferai que citer une phrase de la thèse de F. Girault [21] : « En logique linéaire, les séquents expriment des déductions sur des propositions, qui ne répondent pas au principe de transmission de la vérité, car les prémisses sont consommées pour produire les conclusions. »

Les règles du calcul des séquents en logique linéaire sont données dans l'annexe A. Elles définissent d'un point de vue syntaxique les connecteurs. Voici une présentation succinte de la négation linéaire et des trois familles de connecteurs et de certaines de leurs propriétés.

La négation linéaire

La négation linéaire est nommée ***nil*** et se note $(-)^{\perp}$. Elle vérifie la propriété d'involution :

$$(A^{\perp})^{\perp} \dashv\vdash A \qquad \text{où } A \text{ est une formule}$$

Nous interprétons la négation *nil* comme un inverseur involutif de l'orientation *consommable/productible* (respectivement $A/A^{\perp}$) des exemplaires de ressources vis-à-vis de la dimension temporelle implicite à une action.

Les multiplicatifs

l'implication linéaire Elle permet de représenter une action car elle exprime la possibilité de *produire* une ressource (ou une formule) en en consommant une autre (ou une autre formule) ; nous l'appellerons aussi *ressource d'action* pour son utilisation dans le calcul des séquents. Elle se nomme *entraîne* et se note $\multimap$. Par exemple, la formule $A \multimap B$ exprime la possibilité de produire B en consommant A. Mais pour que cela soit effectivement possible, il faut qu'une ressource A soit disponible (*consommable*).

la conjonction multiplicative Elle permet de représenter la notion de cumul « conjonctif » d'exemplaires de ressources. Elle se nomme *fois* et se note $\otimes$. Par exemple, la proposition $A \otimes A$ indique que la ressource A est disponible deux fois et peut aussi se noter $A^{\otimes 2}$. Elle est associative, commutative et possède un élément neutre nommé *un* qui se note 1. À gauche du tourniquet, elle est équivalente à la virgule. Nous avons donc la propriété caractéristique suivante :

$$A, B \vdash A \otimes B$$

la disjonction multiplicative Elle permet de représenter le cumul « disjonctif » d'exemplaires de ressources. Elle se nomme *par* et se note $\parr$. Par exemple, la proposition $A \parr A$ indique également que les deux exemplaires de A sont disponibles. Tout comme la conjonction, elle est associative, commutative et possède un élément neutre nommé *faux* qui se note $\bot$. À droite du tourniquet, elle est équivalente à la virgule.

Elle vérifie donc la propriété caractéristique suivante :

$$A \parr B \vdash A, B$$

La différence entre les cumuls conjonctifs et disjonctifs est donc la suivante. La formule $A \otimes A$ représente deux ressources A pouvant être consommées indépendamment puisqu'à gauche du méta-connecteur $\vdash$ cette formule est équivalente à A, A. Par contre la formule $A \parr A$ représente deux ressources A pouvant être produites indépendamment puisqu'à droite du méta-connecteur $\vdash$ cette formule est équivalente à A, A. Il est important de souligner que le séquent $A \otimes B \vdash A \parr B$ n'est pas prouvable en logique linéaire alors qu'en logique classique $A \wedge B \vdash A \vee B$ l'est.

Le connecteur $\parr$ permet de faire la correspondance avec l'implication linéaire par l'intermédiaire de l'équivalence suivante :

$$A \multimap B \quad \equiv \quad A^{\bot} \parr B$$

qui indique que nous ne pouvons pas avoir A et B à la fois, en effet c'est la consommation de A qui va permettre de produire B.

La combinaison des connecteurs multiplicatifs permet d'exprimer certains aspects liés à la gestion des ressources :

- le partage d'exemplaires de ressources,
- la concurrence pour la consommation d'exemplaires communs à des actions indépendantes.

Exemple : Soit le système de séquents suivant.

$$\begin{array}{lcl} A \otimes (A \multimap B) & \vdash & B \\ A \otimes (A \multimap C) & \vdash & C \end{array}$$

Cet exemple montre bien que A est un exemplaire de ressource indépendant en conflit pour deux ressources d'action $(A \multimap B)$ ou $(A \multimap C)$. Le séquent $A \otimes (A \multimap B) \otimes (A \multimap C) \nvdash B \otimes C$ est faux alors qu'il serait vrai avec les connecteurs $\wedge$ et $\vee$ de la logique classique.

Les additifs

Contrairement aux multiplicatifs, ces connecteurs sont idempotents. Le dédoublement des connecteurs $\wedge$ et $\vee$ de la logique classique en quatre connecteurs (deux multiplicatifs et deux additifs) est une conséquence directe de la suppression des règles de contraction et d'affaiblissement.

la conjonction additive Elle se nomme *avec* et se note : $\&$. Elle est idempotente tout en restant dans un contexte général de linéarité : $A \dashv\vdash A \& A$. Elle nous permet de gérer un conflit entre deux ressources de manière externe, c'est-à-dire que pour nous, en tant qu'observateur, il nous est possible de faire un choix. Par exemple, nous pouvons avoir :

- soit $A \ \& \ B \not\vdash A$,
- soit $A \ \& \ B \not\vdash B$.

Nous pouvons aussi avoir : $A \otimes [(A \multimap B) \ \& \ (A \multimap C)] \vdash B \ \& \ C$.

La conjonction additive est associative, commutative et possède elle aussi un élément neutre nommé *vrai* et noté $\top$ qui est absorbant pour le connecteur *par*.

la disjonction additive Elle se nomme *plus* et se note : $\oplus$. Elle est idempotente tout en restant dans un contexte général de linéarité : $A \dashv\vdash A \oplus A$. Elle ne nous permet pas de gérer un conflit entre deux ressources contrairement au connecteur précédent. Nous n'avons pas la maîtrise du choix ici, nous subissons. La notion de choix *interne* peut représenter ce phénomène. La formule $A \oplus B$ indique qu'il n'y a qu'une seule ressource de disponible mais nous ne pouvons pas savoir laquelle. Ce qui donne :

$$A \not\vdash A \oplus B \quad \textit{ou} \quad B \not\vdash A \oplus B$$

On retrouve bien ici le principe de la disjonction.

La disjonction additive est associative, commutative et possède elle aussi un élément neutre nommé *zéro* et noté 0 qui est absorbant pour le connecteur *fois*.

En termes d'informatique, la distinction $\&/\oplus$ correspond à la distinction entre les indéterminismes externe et interne.

Les exponentiels

Ces connecteurs permettent de réintroduire la propriété d'idempotence pour les multiplicatifs de façon « controlée » par l'intermédiaire des deux connecteurs unaires suivants :

La conjonction exponentielle : Elle se nomme *bien sûr* et se note : $!(-)$. Elle permet de rétablir l'idempotence du connecteur « fois » dans le membre gauche d'un séquent.

La disjonction exponentielle : Elle se nomme *pourquoi pas* et se note : $?(-)$. Elle permet de rétablir l'idempotence du connecteur « par » dans le membre droit d'un séquent.

La conjonction exponentielle est le connecteur qu'a utilisé C. Brown dans son approche sur les liens entre la logique linéaire et les réseaux de Petri pour représenter l'utilisation en quantité indéfinie d'une ressource de franchissement de transition.

Nous montrerons par la suite que le fragment multiplicatif de la logique linéaire intuitionniste (MILL) est suffisant pour l'approche que nous avons retenue. C'est-à-dire que nous n'utiliserons que les connecteurs « fois » et « implication linéaire ».

2.2 Les réseaux de Petri

2.2.1 Rappels et notations

Comme nous l'avons souligné auparavant, nous avons retenu les réseaux de Petri pour leur capacité à modéliser des systèmes dynamiques avec parallélisme et à prendre en compte aussi bien les états (représentés par des places) que les événements (représentés par des transitions).

Définition 6 (Réseau de Petri) *Un réseau de Petri R est un triplet $< P, T, W >$ où P est un ensemble fini de places, T est un ensemble fini de transitions et W est la matrice de pondération des arcs telle que :*
$W \subseteq (P \times T) \cup (T \times P)$.

Il y a deux types de nœuds dans les réseaux : les places et les transitions.
Soit $X = P \cup T$ l'ensemble des éléments de R :

1. L'ensemble des éléments d'entrée d'un nœud $x \in X$, noté ${}^\bullet x$, est l'ensemble $\{y \in X, W(y,x) \neq 0\}$.
2. L'ensemble des élements de sortie d'un nœud $x \in X$, noté $x^\bullet$, est l'ensemble $\{y \in X, W(x,y) \neq 0\}$.

L'ensemble ${}^\bullet x \cup x^\bullet$ s'écrit ${}^\bullet x^\bullet$.

$Pre(t,p)$ et $Post(t,p)$ sont les multi-ensembles appelés respectivement incidence avant et incidence arrière tels que :

$$\forall p \in P, t \in T : \begin{cases} Pre(t,p) & = W(p,t) \\ Post(t,p) & = W(t,p) \end{cases}$$

L'état du système représenté est associé à un marquage du réseau de Petri. Un marquage M d'un réseau de Petri R est un multi-ensemble $M : P \longrightarrow \mathbb{N}$.

Définition 7 (Réseau de Petri marqué) *Un réseau marqué N est un couple $< R, M_0 >$ où R est un réseau de Petri et M_0 est le marquage initial.*

Soit deux multi-ensembles de places M_1 et M_2 : $M_1 \sqsubseteq M_2$ si $\forall p \in P$, $M_1(p) \leq M_2(p)$.

2.2.2 Accessibilité entre marquages dans un réseau de Petri

L'accessibilité d'un marquage dans un réseau de Petri traduit le fait qu'un certain état du réseau sera atteint à partir d'un état donné après une séquence de franchissements de transitions. C'est la propriété que nous nous efforcerons d'étudier par la suite.

Pour qu'une transition soit franchissable, il faut qu'elle soit *sensibilisée* par un marquage.

Définition 8 (Franchissement de transition) *Une transition t est franchissable pour un marquage M si et seulement si :*

$$\forall p \in P, M(p) \geq Pre(t, p)$$

La notation $M[t >$ est utilisée.

Le franchissement d'une transition t fait évoluer le marquage M vers un nouveau marquage M' tel que $M'(p) = M(p) - Pre(t, p) + Post(t, p)$ pour tout p.

Soit σ une séquence de franchissements de transitions faisant évoluer un marquage M vers un marquage M'.

Définition 9 (Accessibilité d'un marquage) *M' est accessible à partir de M si et seulement si :*

$$\exists \sigma \in T^*, M[\sigma > M'$$

2.3 Traduction en logique linéaire

La traduction en logique linéaire des réseaux de Petri utilisée ici a été proposée par F. Girault dans sa thèse [21]. Dans cette partie, nous ne ferons que reprendre ses formulations.

2.3.1 Traductions dans le fragment MILL

Pour traduire un réseau de Petri en logique linéaire nous n'avons pas besoin d'utiliser tous les connecteurs. Dans l'approche suivie, seuls les connecteurs du fragment multiplicatif (MILL, multiplicative intuitionnist linear logic) sont utilisés :

- l'implication linéaire assimilée à une ressource d'action,
- la conjonction multiplicative qui représente un cumul de ressources.

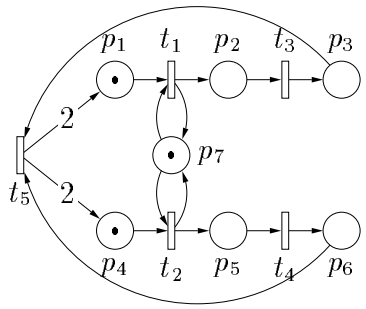

FIG. 2.1: Exemple de réseau de Petri

La disjonction multiplicative n'est pas utilisée car il est difficile de travailler simultanément avec deux connecteurs exprimant le cumul des ressources. Lorsqu'une transition est franchie, un certain nombre de ressources sont consommées « simultanément » et produites aussi « simultanément ». C'est ce que nous avons choisi d'exprimer par la conjonction multiplicative $\otimes$.

Traduction des marquages Un marquage traduit un état du système, c'est-à-dire une présence simultanée de ressources qui est traduite en logique linéaire comme un cumul conjoint de ressources.

La traduction logique d'un marquage est donc une conjonction d'atomes propositionnels p_i, où chaque atome représente un jeton de la place correspondante, qui a la forme : $p_i \otimes p_j \otimes p_k \otimes \ldots$

Considérons le fragment de réseau de la figure 2.1. C'est un réseau dont le marquage initial M_0 est tel que les places p_1, p_4, p_7 contiennent un jeton. Il est traduit par le monôme en $\otimes$ suivant : $p_1 \otimes p_4 \otimes p_7$.

Traduction des transitions Le franchissement d'une transition est assimilé à une action qui produit des ressources en en consommant. Pour traduire cette action en logique linéaire nous utilisons l'implication linéaire $\multimap$. Elle est l'implication causale de production et consommation de ressources.

La traduction logique du franchissement d'une transition t comme ressource d'action est donc :

$$t \quad : \quad Pre(t) \multimap Post(t) \tag{2.1}$$

où $Pre(t)$ et $Post(t)$ sont des monômes en $\otimes$ comme des marquages.

Considérons la transition t_5 du fragment de réseau de la figure 2.1. Elle est traduite par la formule : $p_3 \otimes p_6 \multimap p_1 \otimes p_1 \otimes p_4 \otimes p_4$.

Si nous souhaitons représenter une séquence de franchissements de transitions, nous sommes obligés d'énumérer autant de formules *implicatives* que de transitions à franchir. Cette énumération de franchissements de transitions est représentée par une liste non ordonnée de formules de la forme 2.1 séparées par le méta-connecteur « , ». La virgule étant commutative, c'est pour cela que nous n'avons pas d'ordre.

Considérons le fragment de la figure 2.1. La séquence de franchissements $\sigma = \{t_1 ; t_2 ; t_3 ; t_4 ; t_5\}$ sera traduite en logique linéaire par la liste l suivante : $p_1 \otimes p_7 \multimap p_2 \otimes p_7$, $p_4 \otimes p_7 \multimap p_5 \otimes p_7$, $p_2 \multimap p_3, p_5 \multimap p_6$, $p_3 \otimes p_6 \multimap p_1 \otimes p_1 \otimes p_4 \otimes p_4$.

Même si une transition d'un réseau de Petri est une ressource utilisable indéfiniment, nous devons énumérer une transition autant de fois que nous souhaitons l'utiliser dans notre preuve en logique linéaire. En effet, chaque formule correspond à un franchissement de transition et non pas à une description statique d'une transition.

2.3.2 Traduction de l'accessibilité

L'étude de l'accessibilité reste une raison essentielle du choix de la logique linéaire. Pour pouvoir traduire un problème d'accessibilité d'un réseau en logique linéaire, nous avons besoin de traduire le marquage initial M_0, le marquage final M_n et la liste l_0 des transitions que nous allons franchir pour aller d'un marquage à l'autre. Ce problème est traduit sous la forme d'un séquent.

À partir d'un réseau, un tel problème s'écrit $M_0[\sigma_0 > M_n$ où σ_0 est un suite ordonnée des éléments de l_0. En logique linéaire la vérification de l'accessibilité se traduit par la prouvabilité du séquent associé :

$$M_0, l_0 \vdash M_n \tag{2.2}$$

Nous nous ramenons donc à une preuve de théorème : le théorème représente notre problème d'accessibilité et nous le posons sous la forme d'un séquent. Le fait que la preuve d'un tel séquent prouve l'accessibilité de M_n depuis M_0 est une conséquence de l'équivalence entre prouvabilité et accessibilité qui a été démontrée par F. Girault [21] dans le cadre de la traduction décrite ci-dessus.

Soit le fragment de la figure 2.1. Les franchissements de transitions sont notés comme suit :

$$\begin{array}{llll} t_1 & : & p_1 \otimes p_7 \multimap p_2 \otimes p_7 & \qquad t_3 \ : \ p_2 \multimap p_3 \\ t_2 & : & p_4 \otimes p_7 \multimap p_5 \otimes p_7 & \qquad t_4 \ : \ p_5 \multimap p_6 \\ t_5 & : & p_3 \otimes p_6 \multimap p_1 \otimes p_1 \otimes p_4 \otimes p_4 & \end{array} \tag{2.3}$$

Le problème de l'accessibilité du marquage $M_n = \{p_1, p_1, p_4, p_4, p_7\}$ à partir du marquage $M_0 = \{p_1, p_4, p_7\}$ *via* les franchissements de la liste $l_0 = \{t_1, t_2, t_3, t_4\}$ est traduit par le séquent :

$$\underbrace{p_1 \otimes p_4 \otimes p_7}_{M_0}, \underbrace{t_1, t_2, t_3, t_4, t_5}_{l_0} \vdash \underbrace{p_1 \otimes p_1 \otimes p_4 \otimes p_4 \otimes p_7}_{M_n} \tag{2.4}$$

Remarque : Une séquence σ est totalement ordonnée alors qu'une liste l_0 ne l'est pas. Les éléments de cette liste étant vues comme des ressources, ils peuvent être utilisés indépendamment. Le séquent ci-dessus exprime donc un problème différent de celui de l'accessibilité de M_n à partir de M_0 par le franchissement de la séquence $\sigma = t_1; t_2; t_3; t_4; t_5$ exprimé par $M_0[\sigma > M_n$.

2.4 Preuve d'un séquent

Le résultat fondamental sur lequel nous nous appuyons est l'équivalence entre prouvabilité d'un séquent et accessibilité dans un réseau de Petri. Le cadre de la preuve est le calcul des séquents (approche syntaxique). Elle consiste à montrer que les séquents sont syntaxiquement corrects.

2.4.1 Introduction des règles du fragment MILL

Chaque connecteur de la logique linéaire possède une règle d'élimination différente suivant que le connecteur se situe dans les prémisses (à gauche) ou dans les conclusions (à droite) du séquent. L'élimination d'un connecteur par une règle montre que le connecteur a été correctement introduit.

Une règle est une arborescence de séquents de profondeur un ou deux. Comme pour un séquent, il y a une partie prémisse et une partie conclusion. La racine s'appelle le *séquent conclusion* et les feuilles (si elles existent) les *séquents prémisses* de la règle.

Lors de la construction d'une preuve, nous cherchons à prouver le séquent conclusion en prouvant chaque séquent prémisse. Suivant le nombre de séquent prémisse, nous parlons de règle binaire, unaire ou axiome. Une règle s'écrit de la manière suivante :

$$\frac{\text{Séquent Prémisse 1} \quad \text{Séquent Prémisse 2}}{\text{Séquent Conclusion}} \text{ Nom Règle}_{attribut}$$

L'attribut associé au nom de la règle est L ou R pour indiquer que l'on applique la règle à gauche (L) ou à droite (R) du tourniquet ($\vdash$) du séquent conclusion concerné. S'il n'y a pas d'attribut, cela signifie que la règle s'applique à l'ensemble du séquent comme c'est le cas pour les règles de coupure et d'identité.

Il existe trois groupes de règles en logique linéaire : les règles d'*identité*, les règles *structurelles* et les règles *logiques*. Dans le cadre de notre travail, nous nous situons dans un fragment réduit MILL de cette logique. Chaque membre du séquent est composé de séquences finies de formules, ces séquences pouvant être éventuellement vides. Soit A un atome, F, G et H des formules, Γ et Δ des blocs de formules (connectés par « , »). Une formule de la forme $\langle F \rangle$ est définie par :

$$\langle F \rangle \quad ::= \quad A \quad | \quad \langle F \rangle \otimes \langle F \rangle \quad | \quad \langle F \rangle \multimap \langle F \rangle$$

Voici les règles du fragment MILL :

Identité Elle se note Id dans une preuve. C'est la règle axiome du groupe identité.

$$\frac{}{A \vdash A}\ \mathrm{id} \tag{2.5}$$

Coupure Elle se note Cut dans une preuve. C'est une règle binaire. Elle indique qu'il est possible de prouver un séquent en le décomposant en deux séquents reliés par l'intermédiaire d'un lemme (la formule F ici).

$$\frac{\Gamma \vdash F \quad F, \Delta \vdash H}{\Gamma, \Delta \vdash H}\ \mathrm{cut} \tag{2.6}$$

Échange Elle se note X_L ou X_R suivant son application à gauche ou à droite d'un séquent. Dans le fragment MILL, seule la règle d'échange à gauche X_L subsiste étant donné que les conclusions des séquents sont des monômes en $\otimes$ (marquages). C'est une règle unaire du groupe structurel. Elle permet de permuter deux formules successives d'un même côté d'un séquent. Cela exprime la commutativité de la virgule en logique linéaire.

$$\frac{\Gamma, F, G, \Delta \vdash H}{\Gamma, G, F, \Delta \vdash H}\ \mathrm{X}_L \tag{2.7}$$

Par la suite, l'utilisation de cette règle étant implicite, elle ne sera plus mentionnée.

Implication linéaire Elle se note $\multimap_L$ ou $\multimap_R$ suivant son introduction à gauche ou à droite d'un séquent. Dans le fragment MILL, seule $\multimap_L$ subsiste étant donné que les conclusions des séquents sont des monômes en $\otimes$ (marquages). Ce sont donc des formules sans le connecteur $\multimap$.

$$\frac{\Gamma \vdash F \quad \Delta, G \vdash H}{\Gamma, \Delta, F \multimap G \vdash H}\ \multimap_L \tag{2.8}$$

Conjonction multiplicative Elle se note $\otimes_L$ ou $\otimes_R$ suivant son introduction à gauche ou à droite d'un séquent. Ce sont deux règles du groupe logique dont l'une ($\otimes_L$) est unaire et l'autre binaire ($\otimes_R$).

$$\frac{\Gamma, F, G \vdash H}{\Gamma, F \otimes G \vdash H}\ \otimes_L \qquad \frac{\Gamma \vdash F \quad \Delta \vdash G}{\Gamma, \Delta \vdash F \otimes G}\ \otimes_R \tag{2.9}$$

Introduction à gauche de $\otimes$ (a) Introduction à droite de $\otimes$ (b)

À gauche, la règle $\otimes_L$ est utilisée pour transformer un marquage en une liste d'atomes séparés par des virgules. À droite, la règles $\otimes_R$ est utilisée pour transformer un séquent tel que $A, B \vdash A \otimes B$ en deux séquent identités $A \vdash A$ et $B \vdash B$. Elle est souvent utilisée pour terminer une preuve.

L'ensemble des règles du fragment MILL de la logique linéaire est donné dans l'annexe A.

2.4.2 Calcul des séquents

La syntaxe de la logique linéaire a été définie à l'aide du *calcul des séquents* introduit par G. Gentzen en 1934. Nous l'utilisons pour construire nos preuves dans le fragment MILL en appliquant successivement les règles introduites précédemment : Id, X_L, $\multimap_L$, $\otimes_L$, $\otimes_R$.

Comme nous travaillons uniquement avec des marquages dans le membre droit de nos séquents, nous n'avons pas d'implication linéaire dans ce membre. Ce membre sera donc réduit à un seul monôme en $\otimes$ dans notre approche (fragment MILL).

La construction d'une preuve à l'aide du calcul des séquents consiste à éliminer successivement tous les connecteurs jusqu'à n'obtenir que des séquents identité du type $A \vdash A$ par une suite d'applications de règles. Cette construction conduit à un arbre qui est une preuve syntaxique. Une preuve est conduite de bas en haut en posant comme racine de l'arbre de preuve le séquent que nous souhaitons démontrer. Un séquent est prouvé si chaque feuille de l'arbre correspond à un séquent axiome.

L'un des points importants du calcul des séquents est l'usage de la règle de coupure 2.6 qui permet de prouver un séquent en introduisant des formules exogènes qui sont des déductions intermédiaires.

Toutefois, il a été démontré que si un séquent est prouvable alors il existe une preuve de ce séquent, plus directe, sans utiliser cette règle.

2.4.3 Arbre de preuve canonique

Un séquent peut être prouvé par plusieurs arbres de preuves avec ou sans règle de coupure. Dans le fragment MILL, les preuves sont décidables. Une méthode de preuve systématique peut donc être définie. De plus, les séquents d'accessibilité ont une forme spécifique. La définition d'une méthode canonique pour construire les arbres de preuves est donc simplifiée.

Comme toute preuve canonique, elle ne fait pas intervenir la règle de coupure. Par souci de simplification, l'utilisation de la règle X_L exprimant la commutativité ne sera pas explicitée. La preuve canonique travaille directement sur les relations de causalité, c'est pourquoi elle ne manipule que des formules de type « franchissements de transition » et des atomes représentant des jetons. Pour cela, nous devons transformer notre marquage inital M_0 du membre gauche de notre séquent à prouver en une liste d'atomes indépendants $\mathcal{M}_0$.

Soit $\mathcal{M} = P_1, P_2, \ldots, P_k$ une liste d'atomes (jetons portant des noms de places). Nous définissons $\eta(\mathcal{M})$ comme le multi-ensemble d'atomes $M = \eta(\mathcal{M})$ avec $M(p) = |\{i \in [1,n] : P_i = p\}|$. Le nombre de places du réseau est n. Le degré de la place P dans $\eta(\mathcal{M})$ est le nombre de fois que le nom de p apparaît dans la liste $\mathcal{M}$ et le nombre de jetons contenus dans p pour le marquage M.

Nous décomposons notre construction d'arbre en trois étapes.

Étape initiale

Principe Démarrant avec un séquent de la forme $M_0, l_0 \vdash M_n$, l'introduction à gauche du $\otimes$ (règle 2.9(a)) est appliquée de manière itérative jusqu'à ce que le marquage M_0 soit transformé en une liste d'atomes séparés par des virgules. D'un point de vue logique, cela signifie que les atomes peuvent être utilisés indépendamment dans les règles. Soit $\mathcal{M}_0$ cette liste qui n'est plus une formule mais un bloc d'atomes. Nous avons $M_0 = \eta(\mathcal{M}_0)$ et le séquent obtenu est $\mathcal{M}_0, l_0 \vdash M_n$.

Exemple Prenons le cas du séquent 2.4 :
$\mathcal{M}_0 = p_1, p_4, p_7$, $l_0 = t_1, t_2, t_3, t_4, t_5$ et $M_n = p_1 \otimes p_1 \otimes p_4 \otimes p_4 \otimes p_7$.
Nous avons :

$$\dfrac{\dfrac{p_1, p_4, p_7, t_1, t_2, t_3, t_4, t_5 \vdash p_1 \otimes p_1 \otimes p_4 \otimes p_4 \otimes p_7}{p_1, p_4 \otimes p_7, t_1, t_2, t_3, t_4, t_5 \vdash p_1 \otimes p_1 \otimes p_4 \otimes p_4 \otimes p_7} \otimes_L}{p_1 \otimes p_4 \otimes p_7, t_1, t_2, t_3, t_4, t_5 \vdash p_1 \otimes p_1 \otimes p_4 \otimes p_4 \otimes p_7} \otimes_L \tag{2.10}$$

Étape itérative

Principe Cette étape doit être exécutée une et une seule fois par élément de la liste l. Elle est exécutée pour les séquents de la forme :

$$\mathcal{M}_i, l_i \vdash M_n \tag{2.11}$$

où $\mathcal{M}_i$ est une liste d'atomes séparés par des virgules, l_i est une liste de tirs de transitions et M_n le marquage final. Elle est décomposée en trois sous-étapes (ou pas itératifs).

Premier pas Le premier pas est l'application de l'introduction à gauche de $\multimap$ (règle 2.8(a)) pour le tir de transition choisi t. Le séquent courant de la preuve est ainsi réécrit sous la forme de deux séquents. Nous avons $F = Pre(t)$ et $G = Post(t)$ dans la règle 2.8(a). Nous nous restreignons au cas où Γ est une sous-liste de $\mathcal{M}_i$. Étant donné qu'un séquent qui est mal équilibré du point de vue des atomes ne peut pas être prouvé, une preuve ne peut être continuée que si Γ contient exactement les atomes de $Pre(t)$ avec la même multiplicité : ceci est une condition nécessaire et suffisante. En conséquence, $Pre(t) \sqsubseteq \eta(\mathcal{M}_i)$ et $\eta(\Gamma) = Pre(t)$. Le bloc Δ est composé du reste de $\mathcal{M}_i$ concaténé avec le reste de l_i après avoir éliminé la formule correspondant à t.

Considérons le franchissement de la transition t et posons $\mathcal{M}_i = \Gamma, \mathcal{M'}_i$ et $l_i = Pre(t) \multimap Post(t), l_{i+1}$, nous obtenons la construction suivante :

$$\dfrac{\Gamma \vdash Pre(t) \quad \mathcal{M'}_i, Post(t), l_{i+1} \vdash M_n}{\Gamma, \mathcal{M'}_i, Pre(t) \multimap Post(t), l_{i+1} \vdash M_n} \multimap_L \tag{2.12}$$

Notons que $\mathcal{M}_i$, $\mathcal{M'}_i$, Γ et l_{i+1} sont des blocs et $Pre(t)$, $Post(t)$ et M_n sont des monômes en $\otimes$ qui sont vus comme des multi-ensembles.

Second pas Ce second pas consiste à appliquer la règle 2.9(a) ($\otimes_L$) itérativement sur la formule $G = Post(t)$ afin de la transformer en une liste d'atomes $\mathcal{M}_{i+1}$. Le séquent obtenu est alors de la forme de 2.11. $\mathcal{M}_{i+1}$ est obtenu à partir de $\mathcal{M}_i$ en y enlevant les atomes de $Pre(t)$ et en y ajoutant les atomes de $Post(t)$. l_{i+1} est obtenu à partir de l_i en enlevant la formule correspondant à t. La partie droite du séquent (M_n) reste inchangée.

Troisième pas Le troisième pas consiste à appliquer la règle 2.9(b) ($\otimes_R$) itérativement sur la formule $F = Pre(t)$ afin d'obtenir autant de séquents identités qu'il y a d'atomes dans F. Nous appliquons alors la règle 2.5 identité pour produire les feuilles de l'arbre de preuve.

Exemple Prenons la preuve du séquent 2.4. Si nous considérons le franchissement de la transition t_1, nous avons le fragment d'arbre de preuve suivant :

$$\dfrac{\dfrac{\dfrac{}{p_1 \vdash p_1}\,\mathrm{id} \quad \dfrac{}{p_7 \vdash p_7}\,\mathrm{id}}{p_1, p_7 \vdash p_1 \otimes p_7}\otimes_R \quad \dfrac{p_2, p_4, p_7, t_2, t_3, t_4, t_5 \vdash M_n}{p_4, p_2 \otimes p_7, t_2, t_3, t_4, t_5 \vdash M_n}\otimes_L}{p_1, p_4, p_7, \underbrace{p_1 \otimes p_7 \multimap p_2 \otimes p_7}_{t_1}, t_2, t_3, t_4, t_5 \vdash M_n} \multimap_L \tag{2.13}$$

Étape finale

Principe L'étape finale qui débute lorsque la liste l_i est vide, consiste à appliquer itérativement la règle $\otimes_R$ dans le but de séparer le séquent en deux ensembles d'identités afin de pouvoir appliquer la règle identité. Cette étape n'est possible que si la dernière liste de jetons $\mathcal{M}_i$ est composée des mêmes jetons que ceux qui composent le marquage final M_n. Si ce n'est pas le cas, la preuve échoue ; une preuve qui échoue ne signifie pas forcément que le séquent n'est pas prouvable car l' ordre d'élimination des franchissements de transition (étape itrative) peut avoir de l'importance.

Exemple Prenons de nouveau la preuve du séquent 2.4, on obtient le fragment de preuve suivant lors de l'étape finale après avoir appliqué, par exemple, l'étape itérative aux franchissements de t_1 puis t_2, t_3, t_4 et t_5 :

$$\dfrac{\dfrac{}{p_1 \vdash p_1}\,\mathrm{id} \quad \dfrac{\dfrac{}{p_1 \vdash p_1}\,\mathrm{id} \quad \dfrac{\cdots \qquad \cdots}{p_4, p_4, p_7 \vdash p_4 \otimes p_4 \otimes p_7}\otimes_R}{p_1, p_4, p_4, p_7 \vdash p_1 \otimes p_4 \otimes p_4 \otimes p_7}\otimes_R}{p_1, p_1, p_4, p_4, p_7 \vdash p_1 \otimes p_1 \otimes p_4 \otimes p_4 \otimes p_7}\otimes_R \tag{2.14}$$

L'arbre de preuve canonique global est donné en annexe B.

Lors de la constuction de l'arbre de preuve, nous vérifions pas à pas que les atomes sont disponibles et peuvent être associés aux ressources d'action (les franchissements). Du

point de vue du réseau de Petri cela veut dire que les jetons sont disponibles pour les franchissements des transitions. Dans l'exemple ci-dessus, l'étape itérative a été appliquée aux franchissements de t_1 puis t_2, t_3, t_4 et t_5. Cela veut donc dire que la séquence $\sigma = t_1; t_2; t_3; t_4; t_5$ est franchissable à partir du marquage M_0 et produit le marquage M_n.

En fait, l'information produite par l'arbre de preuve est plus précise que l'ordre total de la séquence. En effet lors de l'étape itérative, la vérification que les atomes de $Pre(t)$ sont présents dans $\mathcal{M}_i$ est indépendante du contexte (la linéarité de la logique linéaire fait que toute preuve reste valable si nous ajoutons des jetons dans le marquage initial à condition de les ajouter également dans le marquage final). Les seules relations de précédence strictement nécessaires sont donc celles qui relient les règles d'élimination de l'implication linéaire dans les formules de franchissement qui produisent des jetons et celles qui les consomment. Par exemple, dans l'arbre de preuve construit ci-dessus correspondant à la séquence d'élimination $\sigma = t_1; t_2; t_3; t_4; t_5$, il est nécessaire que t_1 précède t_3 car le franchissement de t_3 consomme un jeton produit par t_1 alors que l'ordre entre t_3 et t_4 est indifférent.

Pour mettre ces relations de précédence en évidence, il suffit de savoir quelle règle a introduit un atome P dans la liste $_i$ et quelle règle va l'éliminer. Une relation de précédence existera entre ces deux règles.

2.5 Annotation de séquent et graphe de précédence

L'arbre de preuve canonique d'un séquent peut donc permettre de déduire un ordre partiel pour l'application des règles lors de sa construction. Sans modifier les règles du calcul des séquents, nous allons associer des annotations aux atomes logiques et aux applications des règles pour mettre en évidence cet ordre partiel. Cet ordre partiel est complètement défini par un seul arbre canonique annoté. Cet ordre partiel définit les relations de précédence entre les événements que sont les franchissements des transitions.

2.5.1 Étiquetage de l'arbre de preuve canonique

Chaque fois que nous appliquons la règle $\multimap_L$, nous éliminons une instance de formule décrivant une transition de la liste l_i. Cette élimination intervient au premier pas de l'étape itérative de la construction de l'arbre de preuve canonique. Il est possible d'associer à chaque application de la règle une étiquette avec le nom de la transition associée à la formule implicative éliminée par la règle : (t) . Lorsqu'une transition est franchie plusieurs fois, nous rajoutons en exposant à l'annotation précédente un indice égal au nombre de franchissements effectués. Nous noterons (t^j) l'étiquette signifiant que c'est la j^e élimination d'une formule associée à la transition t.

Si nous considérons le fragment de preuve 2.12, l'annotation produira le nouveau

fragment suivant :

$$\frac{\Gamma \vdash Pre(t) \quad \mathcal{M}'_i, Post(t), l_{i+1} \vdash M_n}{\Gamma, \mathcal{M}'_i, Pre(t) \multimap Post(t), l_{i+1} \vdash M_n} \multimap_L (t^j) \tag{2.15}$$

Les relations de précédence sont liées aux jetons (il n'est pas possible de consommer un jeton s'il n'a pas été préalablement produit). Il faut donc étiqueter les atomes dans l'arbre de preuve. Chaque atome (c'est-à-dire chaque jeton produit ou consommé) peut être étiqueté par l'étiquette d'une règle.

Lorsque cet atome est à gauche du *tourniquet* (symbole $\vdash$), cette étiquette est celle de la règle qui l'a *produit*. Cela signifie que si nous appliquons la règle $\multimap_L$ à la formule $Pre(t) \multimap Post(t)$ pour la j^e fois, tous les atomes de $Post(t)$ seront étiquetés par t^j dans le séquent $\mathcal{M}'_i, Post(t), l_{i+1} \vdash M_n$. Ces atomes sont dans la partie gauche du séquent prémisse de droite du fragment 2.15.

Quand l'atome est à droite du tourniquet, l'étiquette est celle de la règle qui a consommé l'atome. Cela signifie que si nous appliquons la règle $\multimap_L$ à la formule implicative $Pre(t) \multimap Post(t)$ pour la j^e fois, tous les atomes de $Pre(t)$ seront étiquetés par t^j dans le séquent $\Gamma \vdash Pre(t)$. Ces atomes sont dans la partie droite du séquent prémisse de gauche du fragment 2.15.

Nous avons choisi de mettre des étiquettes initiales et finales pour représenter des événements virtuels de production des jetons initiaux et de consommation des jetons finaux. Lorsque cet atome fait partie de la liste initiale $\mathcal{M}_0$, nous l'étiquetons avec I pour indiquer qu'il a été produit à un instant initial inconnu. S'il y a n atomes initiaux dans $\mathcal{M}_0$, ils seront étiquetés par I^k pour k variant de 1 à n. Par souci d'homogénéité avec le marquage final, les atomes de M_n seront étiquetés par F pour indiquer qu'ils seront consommés à un instant final indéfini. S'il y a m atomes dans M_n, ils seront étiquetés par F^k pour k variant de 1 à m.

Tous nos arbres de preuve se terminent par des feuilles de type séquent identité $A \vdash A$. Les feuilles de nos arbres de preuve seront dorénavant de la forme :

A(événement de production) $\vdash$ A(événement de consommation).

Les étiquetages possibles seront les suivants :

- $A(I^j) \vdash A(t_i^k)$ si l'atome A faisait partie du marquage initial et a été consommé à l'instant t_i^k,
- $A(I^j) \vdash A(F^k)$ si l'atome A faisait partie du marquage initial et fait partie du marquage final (en fait cela correspond à une ressource non utilisée),
- $A(t_i^j) \vdash A(t_k^l)$ si l'atome A a été produit à l'instant t_i^j et consommé à l'instant t_k^l,
- $A(t_i^j) \vdash A(F^k)$ si l'atome A a été produit à l'instant t_i^j et qu'il fait partie du marquage final,

Exemple Reprenons l'exemple du fragment d'arbre 2.13.
$\mathcal{M}_0 = p_1, p_4, p_7$ devient $p_1(I^1), p_4(I^2), p_7(I^3)$. Nous choisissons de mettre un indice différent pour chaque instant initial de production I car les atomes initiaux n'ont pas forcément été produits au même instant. En effet les jetons contenus dans les places du réseau de Petri à l'instant initial peuvent provenir de places d'autres fragments de réseau de Petri(approche compositionnelle). Dès le début de la preuve, dans le séquent conclusion nous aurons le marquage initial annoté $M_0 = p_1(I^1) \otimes p_4(I^2) \otimes p_7(I^3)$. Les atomes qui composent les formules implicatives telles que $Pre(t) \multimap Post(t)$ de l_i ne sont pas annotés car ils font partie de formules correspondant à des ressources d'actions.

Dans le fragment 2.16 nous avons représenté le premier franchissement de la transition t_1 par l'utilisation de la règle $\multimap_L$. Suite à l'application de $\multimap_L$, les atomes de $Pre(t_1)$ et de $Post(t_1)$ sont étiquetés par la même étiquette t_1^1, ce qui signifie qu'ils ont été respectivement consommés/produits par le premier franchissement de t_1. Ce pas de l'étape itérative de notre méthode de construction de l'arbre canonique se poursuit par une application de la règle $\otimes_L$ sur $Post(t_1)$ et une application de la règle $\otimes_R$ sur $Pre(t_1)$. L'application de cette dernière règle permet de terminer une branche de notre arbre de preuve par des séquents identité qui montrent quels sont les instants de production et de consommation pour chaque atome. Ici, nous avons une première caractérisation temporelle qualitative des atomes :

- p_1 a été produit à l'instant I^1 et consommé à l'instant t_1^1,
- p_7 a été produit à l'instant I^3 et consommé à l'instant t_1^1,
- p_2 et p_7 (le deuxième exemplaire de p_7 car c'est une ressource partagée dans le réseau de Petri) ont été produits à l'instant t_1^1.

$$\dfrac{\dfrac{\dfrac{}{p_1(I^1) \vdash p_1(t_1^1)}\,\mathrm{id} \quad \dfrac{}{p_7(I^3) \vdash p_7(t_1^1)}\,\mathrm{id}}{p_1(I^1), p_7(I^3) \vdash p_1(t_1^1) \otimes p_7(t_1^1)}\otimes_R \quad \dfrac{p_2(t_1^1), p_4(I^2), p_7(t_1^1), t_2, t_3, t_4, t_5 \vdash M_n}{p_4(I^2), p_2(t_1^1) \otimes p_7(t_1^1), t_2, t_3, t_4, t_5 \vdash M_n}\otimes_L}{p_1(I^1), p_4(I^2), p_7(I^3), \underbrace{p_1 \otimes p_7 \multimap p_2 \otimes p_7}_{t_1}, t_2, t_3, t_4, t_5 \vdash M_n} \multimap_L(t_1^1) \qquad (2.16)$$

De la même façon que pour le premier franchissement de la transition t_1, pour le premier franchissement de la transition t_2, nous obtenons le fragment 2.17 avec la caractérisation temporelle suivante :

- p_4 a été produit à l'instant I^2 et consommé à l'instant t_2^1,
- p_7 a été produit à l'instant t_1^1 et consommé à l'instant t_2^1,
- p_5 et p_7 (le troisième exemplaire) ont été produits à l'instant t_2^1.

$$\dfrac{\dfrac{\dfrac{}{p_4(I^2) \vdash p_4(t_2^1)}\,\mathrm{id} \quad \dfrac{}{p_7(t_1^1) \vdash p_7(t_2^1)}\,\mathrm{id}}{p_4(I^2), p_7(t_1^1) \vdash p_4(t_2^1) \otimes p_7(t_2^1)}\otimes_R \quad \dfrac{p_2(t_1^1), p_5(t_2^1), p_7(t_2^1), t_3, t_4, t_5 \vdash M_n}{p_2(t_1^1), p_5(t_2^1) \otimes p_7(t_2^1), t_3, t_4, t_5 \vdash M_n}\otimes_L}{p_2(t_1^1), p_4(I^2), p_7(t_1^1), \underbrace{p_4 \otimes p_7 \multimap p_5 \otimes p_7}_{t_2}, t_3, t_4, t_5 \vdash M_n} \multimap_L(t_2^1) \qquad (2.17)$$

À la suite de ces deux premières étapes itératives de construction de l'arbre de preuve canonique, nous remarquons plusieurs choses. La première est l'apparition d'un ordre partiel des franchissements des transitions car si nous regroupons les fragments 2.16 et 2.17 (le fragment 2.17 étant la continuité de la branche droite du fragment 2.16), nous remarquons que le premier franchissement de t_1 doit précéder le premier franchissement de t_2 car pour la stratégie de preuve retenue, le jeton de la place p_7 consommé par t_2^1 est celui produit par t_1^1. Ceci est mis en évidence par la feuille identité $p_7(t_1^1) \vdash p_7(t_2^1)$ du fragment 2.17 de l'arbre de preuve. Nous obtenons la relation de précédence $t_1^1 \prec t_2^1$. La deuxième est la caractérisation temporelle précise des atomes par les tirs de transition correspondant aux instants de production et de consommation.

Enfin pour le dernier franchissement de transition t_5 de la liste l_i, nous avons le fragment suivant :

$$\dfrac{\dfrac{\dfrac{}{p_3(t_3^1) \vdash p_3(t_5^1)}\,\mathrm{id} \quad \dfrac{}{p_6(t_4^1) \vdash p_6(t_5^1)}\,\mathrm{id}}{p_3(t_3^1), p_6(t_4^1) \vdash p_3(t_5^1) \otimes p_6(t_5^1)}\otimes_R \quad \dfrac{p_1(t_5^1), p_1(t_5^1), p_4(t_5^1), p_4(t_5^1), p_7(t_2^1) \vdash M_n}{p_7(t_2^1), p_1(t_5^1) \otimes p_1(t_5^1) \otimes p_4(t_5^1) \otimes p_4(t_5^1) \vdash M_n}\otimes_L}{p_3(t_3^1), p_6(t_4^1), p_7(t_2^1), \underbrace{p_3 \otimes p_6 \multimap p_1 \otimes p_1 \otimes p_4 \otimes p_4}_{t_5} \vdash M_n} \multimap_L(t_5^1) \tag{2.18}$$

Pour l'étape finale de la construction de l'arbre de preuve canonique, nous terminons avec le fragment suivant où les feuilles terminales de l'arbre sont des séquents identité avec les atomes des membres droits étiquetés par F^k avec k variant de 1 à 5 :

$$\dfrac{\dfrac{}{p_1(t_5^1) \vdash p_1(F^1)}\,\mathrm{id} \quad \dfrac{}{p_1(t_5^1) \vdash p_1(F^2)}\,\mathrm{id} \quad \dfrac{}{p_4(t_5^1) \vdash p_4(F^3)}\,\mathrm{id} \quad \dfrac{}{p_4(t_5^1) \vdash p_4(F^4)}\,\mathrm{id} \quad \dfrac{}{p_7(t_2^1) \vdash p_7(F^5)}\,\mathrm{id}}{p_1(t_5^1), p_1(t_5^1), p_4(t_5^1), p_4(t_5^1), p_7(t_2^1) \vdash \underbrace{p_1(F^1) \otimes p_1(F^2) \otimes p_4(F^3) \otimes p_4(F^4) \otimes p_7(F^5)}_{M_n}} \otimes_R \tag{2.19}$$

2.5.2 Graphe de précédence

La règle principale pour la construction de l'arbre de preuve canonique est l'élimination de l'implication linéaire à gauche car elle est au cœur de de l'étape itérative. C'est elle qui correspond à l'utilisation du principe de déduction permettant à partir de jetons situés dans les places d'entrée d'une transition de déduire qu'il est possible de les consommer pour produire des jetons dans les places de sortie. Cette règle ne peut pas s'appliquer n'importe quand pour n'importe quelle transition. Pour pouvoir l'appliquer il faut que les jetons soient disponibles, c'est-à-dire qu'ils soient présents dans la liste $\mathcal{M}_i$. Cela veut dire qu'ils ont été préalablement produits par le franchissement d'une transition et donc par l'application de la règle d'élimination à gauche de l'implication linéaire pour cette autre transition. Pour le fragment 2.18, t_5 ne sera franchissable que si les atomes de $Pre(t_5)$ sont contenus dans la liste $\mathcal{M}_i = p_3(t_3^1), p_6(t_4^1), p_7(t_2^1)$. Avec les annotations nous notons

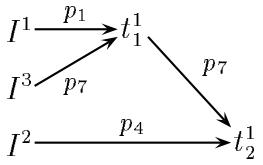

FIG. 2.2: Les relations de précédence pour les deux premiers pas

précisément quelle règle a produit un jeton et quelle règle le consomme. Les applications de la règle *identité*, quant à elles, associent les jetons produits aux jetons consommés.

À partir des feuilles terminales de l'arbre de preuve, nous pouvons construire un graphe de précédence entre les applications des règles du calcul des séquents. À un pas donné de la preuve, deux règles applicables et non reliées par une relation de précédence commutent. L'arbre de preuve obtenu par la commutation de ces règles est équivalent au précédent dans le sens où les deux arbres auront les mêmes feuilles après étiquetage. Comme les règles correspondent à la preuve que les formules implicatives sont syntaxiquement correctes et que les formules correspondent à des franchissements de transition, nous interprétons ces relations de précédence comme des relations de précédence entre les franchissements de transition (définition 3). Les graphes de précédence obtenus sont donc des graphes acycliques composés de nœuds qui sont des événements et d'arcs reliant deux événements successifs. Nos événements sont les instants de franchissements de transition. Les arcs portent le nom des jetons produits et consommés par ces événements.

Chaque arc est défini par une application de la règle *identité* (nœud terminal de l'arbre de preuve). Chacune de ces règles associe un atome logique produit à un atome logique consommé. Pour connaître les relations de précédence entre les règles d'élimination à gauche de l'implication linéaire, il nous suffit de noter par quelle règle chaque atome est produit et par quelle règle chaque atome est consommé. Les atomes initiaux et finaux jouent un rôle un peu différent puisque les premiers ne sont pas produits par une élimination à gauche de l'implication linéaire et les seconds ne sont pas consommés par une telle règle.

La figure 2.2 montre les relations de précédence déduites des deux fragments d'arbre de preuve 2.16 et 2.17. Dans le premier fragment, nous voyons dans les feuilles terminales que l'événement t_1^1 est successeur des événements I^1 et I^3 donc le nœud t_1^1 doit succéder aux nœuds I^1 et I^3. Dans le deuxième fragment, les feuilles terminales nous indiquent que le nœud t_2^1 est successeur des nœuds t_1^1 et I^2. Le fragment de graphe de la figure 2.2 montre les relations de précédence entre les instants de franchissement des transitions déduites de l'arbre de preuve canonique après étiquetage. Le graphe complet de la figure 2.3 montre le scénario menant du marquage initial M_0 au marquage final M_n pour l'arbre de preuve correspondant à l'élimination en séquence de t^1, t^2, t^3, t^4 et t^5. Les événements I^1, I^2 et I^3 correspondent à la mise à disposition initiale des ressources p_1, p_4 et p_7. Il n'y a pas de relation de précédence entre t_3^1 et t_2^1, ni entre t_3^1 et t_4^1. C'est-à-dire que, par exemple,

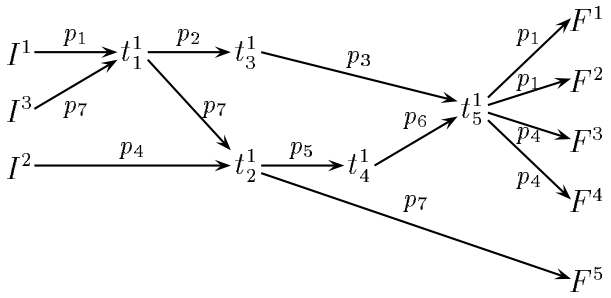

FIG. 2.3: Premier graphe de précédence pour l'arbre canonique complet

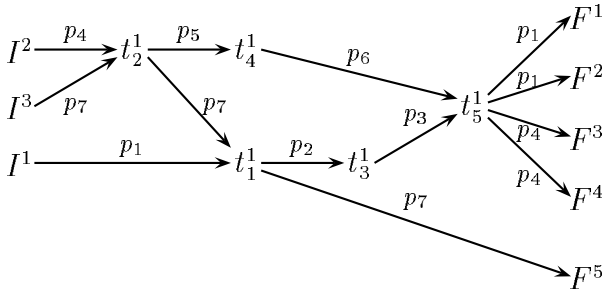

FIG. 2.4: Deuxième graphe de précédence pour l'arbre canonique complet

l'arbre de preuve obtenu par l'élimination en séquence des franchissements de t^1, t^3, t^2, t^4 et t^5 est équivalent au précédent (t^1, t^2, t^3, t^4 et t^5). Leurs feuilles seront identiques même après étiquetage.

Comme les atomes de $Pre(t_1)$ et de $Pre(t_2)$ sont contenus dans la liste $\mathcal{M}_0$, ces deux transitions sont donc franchissables pour le marquage initial. Le problème est que l'atome p_7 appartient à $Pre(t_1)$ et $Pre(t_2)$ et qu'il n'existe qu'en un seul exemplaire dans la liste. Il existe donc un conflit transitions au niveau de cette ressource car nous avons effectivement la possibilité d'utiliser cette ressource pour le franchissement de t_1 et de t_2 au même moment. Le fait de franchir la transition t_1 d'abord génère un arbre de preuve qui nous donne le graphe de précédence complet de la figure 2.3. Le choix de franchir t_2 avant t_1 (qui reste arbitraire) génère un autre arbre de preuve qui nous donne le graphe de précédence de la figure 2.4. Nous obtenons donc deux ordres partiels différents pour le même problème d'accessibilité avec le même marquage initial, le même marquage final et les mêmes transitions. Ce qui différencie ces deux ordres, ce sont les relations de précédence entre les applications des règles. Les arbres de preuve obtenus par les séquences d'applications de règle $t_1; t_2; t_3; t_4; t_5$ et $t_2; t_1; t_3; t_4; t_5$ ne sont pas équivalents. Avant étiquetage leurs feuilles sont identiques, mais après étiquetage ce n'est plus le cas. Par exemple, dans le premier cas nous avons la feuille $p_7(I^3) \vdash p_7(t_1^1)$ et dans le second $p_7(I^3) \vdash p_7(t_2^1)$. Ces deux feuilles s'écrivent $p_7 \vdash p_7$ avant étiquetage.

Nous pouvons remarquer que les deux ordres partiels obtenus par fermeture transitive

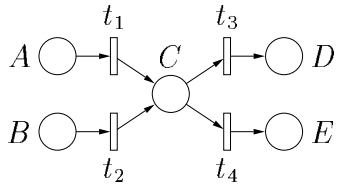

FIG. 2.5: Réseau de Petri avec conflit

à partir des graphes de précédence des figures 2.3 et 2.4 peuvent s'exprimer en utilisant les deux opérateurs " ;" (mise en séquence) et "||" (mise en parallèle). Le premier graphe donne l'expression $t_1^1;(t_3^1 \parallel (t_2^1;t_4^1));t_5^1$ et le second l'expression $t_2^1;(t_4^1 \parallel (t_1^1;t_3^1));t_5^1$. Nous pouvons obtenir des ordres partiels qui ne sont pas du type « série/parallèle », c'est-à-dire qui ne peuvent pas se mettre sous cette forme.

La notion de conflit de transitions correspond à celle existant dans la théorie des réseaux de Petri. Pour notre cadre elle s'exprime de la façon suivante.

Définition 10 (Conflit de transitions) *Soit le séquent d'accessibilité* $\mu_i, l_i \vdash M_n$. *Il y a conflit de transitions s'il existe deux formules de franchissements* $t_{j_1}^{k_1}$ *et* $t_{j_2}^{k_2}$ *appartenant à* l_i *et telles que :*

1. $j_1 \neq j_2$
2. $Pre(t_{j_1}^{k_1}) \sqsubseteq \eta(\mu_i)$ *et* $Pre(t_{j_2}^{k_2}) \sqsubseteq \eta(\mu_i)$
3. *il existe au moins un atome* p *tel que* $p \in Pre(t_{j_1}^{k_1})$ *et* $p \in Pre(t_{j_2}^{k_2})$

D'après le mécanisme d'étiquetage et le principe de l'étape itérative pour la construction de l'arbre de preuve, il est clair que l'arbre obtenu par le franchissement de $t_{j_1}^{k_1}$ avant $t_{j_2}^{k_2}$ et l'arbre obtenu par le franchissement de $t_{j_2}^{k_2}$ avant $t_{j_1}^{k_1}$ ne sont pas équivalents.

Nous allons maintenant illustrer par l'exemple suivant comment il est possible grâce à la méthode d'étiquetage des règles et des atomes de mettre en évidence des décisions cachées en liaison avec l'apparition d'un autre type de conflit que nous appelons « conflit jetons ». Lorsqu'il y a plusieurs jetons dans la même place, l'arbre de preuve ne différencie pas les divers atomes qui leur sont associés puisqu'ils sont tous nommés par le nom de la place du réseau de Petri qui les contient. En effet, l'arbre de preuve travaille en logique propositionnelle. Après étiquetage, les atomes seront différents s'ils ont été produits par des franchissements différents et le choix de l'un ou de l'autre donnera des graphes de précéence différents.

Exemple avec conflit jetons Considérons le réseau de Petri de la figure 2.5. Imaginons qu'il modélise par la place C une opération sur des données, qui peuvent être de types différents suivant qu'elles proviennent de la place A ou de la place B, dont le résultat produit est stocké dans les places D ou E.

Considérons le problème d'accessibilité posé par le séquent :

$$a \otimes b, t_1, t_2, t_3, t_4 \vdash d \otimes e \tag{2.20}$$

$$I^1 \xrightarrow{a} t_1^1 \xrightarrow{c} t_3^1 \xrightarrow{d} F^1$$
$$I^2 \xrightarrow{b} t_2^1 \xrightarrow{c} t_4^1 \xrightarrow{e} F^2$$

FIG. 2.6: Premier ensemble de relations de précédence

Un arbre de preuve canonique de ce séquent sans étiquetage est le suivant (après une première application de la règle d'introdution à gauche de l'opérateur $\otimes_L$) :

$$\dfrac{\dfrac{}{a \vdash a}\,\mathrm{id} \quad \dfrac{\dfrac{}{b \vdash b}\,\mathrm{id} \quad \dfrac{\dfrac{}{c \vdash c}\,\mathrm{id} \quad \dfrac{\dfrac{}{d \vdash d}\,\mathrm{id} \quad \dfrac{}{e \vdash e}\,\mathrm{id}}{d, e \vdash d \otimes e}\otimes_R}{c, d, t_4 \vdash d \otimes e}\multimap_L}{c, c, t_3, t_4 \vdash d \otimes e}\multimap_L}{b, c, t_2, t_3, t_4 \vdash d \otimes e}\multimap_L}{a, b, t_1, t_2, t_3, t_4 \vdash d \otimes e}\multimap_L \tag{2.21}$$

Les annotations correspondant à l'élimination des franchissements de t_1 et de t_2 ne posent pas de problème. Nous obtenons le fragment suivant :

$$\dfrac{\dfrac{}{a(I^1) \vdash a(t_1^1)}\,\mathrm{id} \quad \dfrac{\dfrac{}{b(I^2) \vdash b(t_2^1)}\,\mathrm{id} \quad c(t_1^1), c(t_2^1), t_3, t_4 \vdash d(F^1) \otimes e(F^2)}{b(I^2), c(t_1^1), t_2, t_3, t_4 \vdash d(F^1) \otimes e(F^2)}\multimap_L(t_2^1)}{a(I^1), b(I^2), t_1, t_2, t_3, t_4 \vdash d(F^1) \otimes e(F^2)}\multimap_L(t_1^1) \tag{2.22}$$

Il faut maintenant éliminer le franchissement de t_3 pour prouver le séquent :

$$c(t_1^1), c(t_2^1), \underbrace{c \multimap d}_{t_3}, t_4 \vdash d(F^1) \otimes e(F^2) \tag{2.23}$$

En effet nous devons associer l'atome $c = Pre(t_3)$ soit avec $c(t_1^1)$ soit avec $c(t_2^1)$. Du point de vue strictement logique, il n'y a pas de différence puisque les deux atomes sont identiques (logique propositionelle), alors que lorsque nous voulons mettre en évidence les relations de précédence, les atomes sont différents.

Nous pouvons donc poursuivre les arbres de deux façons suivant le choix de l'atome. La première consiste à choisir $c(t_1^1)$ pour t_3 et donc $c(t_2^1)$ pour t_4. Nous obtenons le fragment :

$$\dfrac{\dfrac{}{c(t_1^1) \vdash c(t_3^1)}\,\mathrm{id} \quad \dfrac{\dfrac{}{c(t_2^1) \vdash c(t_4^1)}\,\mathrm{id} \quad \dfrac{\dfrac{}{d(t_3^1) \vdash d(F^1)}\,\mathrm{id} \quad \dfrac{}{e(t_4^1) \vdash e(F^2)}\,\mathrm{id}}{d(t_3^1), e(t_4^1) \vdash d(F^1) \otimes e(F^2)}\otimes_R}{c(t_2^1), d(t_3^1), t_4 \vdash d(F^1) \otimes e(F^2)}\multimap_L(t_4^1)}{c(t_1^1), c(t_2^1), t_3, t_4 \vdash d(F^1) \otimes e(F^2)}\multimap_L(t_3^1) \tag{2.24}$$

Ce choix donne le graphe de précédence de la figure 2.6.

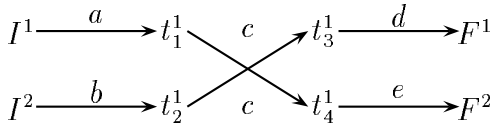

FIG. 2.7: Deuxième ensemble de relations de précédence

La deuxième façon de poursuivre consiste à choisir $c(t_2^1)$ pour le franchissement de t_3 et $c(t_1^1)$ pour celui de t_4. Nous obtenons ainsi :

$$\dfrac{\dfrac{}{c(t_2^1) \vdash c(t_2^1)}\,\mathrm{id} \quad \dfrac{\dfrac{}{c(t_1^1) \vdash c(t_1^1)}\,\mathrm{id} \quad \dfrac{\dfrac{}{d(t_3^1) \vdash d(F^1)}\,\mathrm{id} \quad \dfrac{}{e(t_4^1) \vdash e(F^2)}\,\mathrm{id}}{d(t_3^1), e(t_4^1) \vdash d(F^1) \otimes e(F^2)}\otimes_R}{c(t_1^1), d(t_3^1), t_4 \vdash d(F^1) \otimes e(F^2)} \multimap_L(t_4^1)}{c(t_1^1), c(t_2^1), t_3, t_4 \vdash d(F^1) \otimes e(F^2)} \multimap_L(t_3^1) \qquad (2.25)$$

Cela donne le graphe de précédence de la figure 2.7.

Quel atome choisir parmi deux sachant qu'ils paraissent identiques (ils portent le même nom) mais qu'ils ne proviennent pas de la même source ? Nous remarquons grâce à cet exemple que l'étiquetage que nous avons mis en place durant la construction d'une preuve a permis de résoudre ce problème en différenciant ces atomes. En effet nous avons la possibilité d'utiliser deux ressources identiques *dans la forme* mais différentes par leur origine et leur âge. Seule une étiquette nous permet de faire cette différence. Elle permet de mettre en évidence le choix fait dans l'utilisation de ressources pour le franchissement de transitions. Choix qui n'apparaît pas explicitemment dans le cadre logique propositionnelle car il n'y a pas de différence entre les ressources d'une même place.

Définition 11 (Conflit de jetons) *Soit le séquent d'accessibilité étiqueté* $\mu_i, l_i \vdash M_n$. *Il y a conflit de jetons s'il existe au moins une formule de franchissement de transition* t_j^k *dans la liste* l_i *et une place* p *tels que :*

1. $Pre(t_j^k) \sqsubseteq \eta(\mu_i)$
2. *le degré de* p *dans* $Pre(t_j^k)$ *est inférieur au degré de* p *dans* $\eta(\mu_i)$
3. *il existe deux sous-listes de* μ_i, *soient* Γ_1 *et* Γ_2, *uniquement formées d'atomes* p *telles que :*
   (a) *le degré de* p *dans* $\eta(\Gamma_1)$ *et* $\eta(\Gamma_2)$ *est égal au degré de* p *dans* $Pre(t_j^k)$
   (b) $\exists p(t_{x_1}^{y_1}) \in \Gamma_1$ *tel que* $p(t_{x_1}^{y_1}) \notin \Gamma_2$ *et* $\nexists p(t_{x_2}^{y_1}) \in \Gamma_2$ *et* $\notin \Gamma_1$ *avec* $x_1 \neq x_2$.

La dernière condition vient du fait que nous ne souhaitons pas différencier deux graphes de précédence qui seraient identiques à une numérotation des franchissements de transition près. Nous supposons implicitement que t_i^j précède toujours t_i^k si $j < k$.

2.6 Conclusion

Nous avons présenté dans ce chapitre la méthode que nous utilisons pour traduire un problème d'accessibilité dans un réseau de Petri en logique linéaire. Cette méthode est fondée sur l'équivalence entre le problème de l'accessibilité dans un réseau de Petri et la prouvabilité du séquent de logique linéaire associé. L'originalité de cette méthode est de ne pas raisonner sur un graphe d'états mais sur des événements. Nous avons donc une meilleure repésentation du parallélisme car il n'est pas traité par l'entrelacement.

La méthode de construction des arbres de preuve est canonique par application de manière itérative de la règle d'introduction à gauche de l'implication linéaire $\multimap_L$. Cette règle étant assimilée à un franchissement de transition, elle exprime la relation de causalité entre les jetons consommés et produits. Chaque jeton produit puis consommé pendant la preuve correspond à une relation de précédence entre deux applications de la règle $\multimap_L$ (celle qui a produit le jeton et celle qui l'a consommé). L'ensemble des relations de précédence déduit de l'arbre de preuve canonique définit un ordre partiel parmi l'ensemble des franchissements de transitions qui correspond à un ordre partiel des applications de la règle $\multimap_L$.

Pour différencier les applications de cette règle, nous avons étiqueté chaque application dans l'arbre de preuve par le nom de la transition franchie et le nombre de fois que cette transition a été franchie. Les atomes produits et consommés par l'application de cette règle sont étiquetés par l'étiquette de la règle et les feuilles de l'arbre de preuve définissent alors un ensemble de relations de précedence devant être vérifiées.

Étiqueter les atomes de l'arbre de preuve est utile et nous permet d'introduire un nouveau type de conflit : les conflits jetons. Cela permet de différencier les atomes afin de faire le choix adéquat de l'atome à utiliser pour le franchissement d'une certaine transition. Cela met en valeur des décisions prises qui n'apparaissent pas explicitement sans annotation. Ce n'est plus un atome quelconque qui est produit ou consommé dorénavant mais un atome bien spécifique qui possède une « histoire » liée aux relations de précédence entre les transitions.

À partir de ces résultats, nous sommes capables de décrire précisément les ordres partiels associés aux arbres de preuve. Comme chaque jeton possède une étiquette qui est une caractérisation temporelle composée d'un instant de production et d'un instant de consommation, nous pouvons raisonner qualitativement sur le temps.

Auparavant, il nous faut résoudre un autre problème, celui de la complétude. La question est la suivante : les ordres partiels entre les franchissements que nous obtenons à partir des arbres de preuve sont-ils tous les ordres partiels possibles ? Rappelons en effet que les ordres partiels obtenus par L.A. Künzle dans sa thèse étaient les seuls ordres partiels « série-parallèle » et qu'il avait donné un exemple d'un ensemble de relations de précédence ne s'exprimant pas sous cette forme.

Chapitre 3

Une équivalence entre preuve canonique et processus de réseau de Petri

3.1 Processus de branchement et autres dépliages

Nous avons vu dans le chapitre précédent que nous pouvions obtenir une représentation de l'ordre partiel des événements d'un système (franchissements de transitions d'un réseau de Petri) sous la forme d'un graphe de précédence. Ce graphe est obtenu par construction d'un arbre de preuve en logique linéaire.

Cependant il existe d'autres techniques qui permettent de mettre en évidence le parallélisme et d'obtenir une représentation de cet ordre partiel : les techniques des processus de branchement. Ces techniques sont basées sur le dépliage d'un réseau de Petri. La méthode du dépliage a été introduite par K.L. McMillan [35]. Inversement à notre méthode présentée dans le chapitre précédent, dans les techniques basées sur le dépliage la vérification de l'accessibilité est effectuée après avoir obtenu l'ordre partiel.

Comme pour un graphe d'accessibilité, le but initial était de caractériser tous les marquages accessibles ainsi que les événements sensibilisés à partir de chacun d'eux. Mais plutôt que de représenter les entrelacements de tirs de transitions, l'idée de base est de représenter les tirs de transitions indépendants par des transitions indépendantes.

Les dépliages et les processus de réseaux de Petri constituent une des principales sémantiques *réellement* parallèles des réseaux de Petri [40, 23, 4, 38, 25, 17]. Les processus et les dépliages d'un réseau de Petri pour un marquage initial donné sont des graphes acycliques, finis ou infinis, composés de places et de transitions. En fait, dans de tels réseaux dénommés *réseaux d'occurrence*, les places symbolisent les jetons et les transitions symbolisent les franchissements de transitions. L'unique transition d'entrée d'une place correspond au tir qui a produit le jeton correspondant. La différence avec un dépliage est qu'un processus ne contient pas de conflit. Chaque place possède une seule transition de sortie représentant le tir qui a consommé le jeton correspondant.

Comme précédemment, nous travaillons avec un marquage initial et un marquage final. Le dépliage effectué à partir du réseau de Petri initial sera donc fini. Étant donné que nous ne représentons qu'un comportement donné parmi les dépliages, nous ne considérerons que les processus de réseaux de Petri. En somme nous représenterons les ordres partiels de franchissements de transitions par des processus de réseaux de Petri finis.

Le but de ce chapitre est de montrer que les ordres partiels sur les franchissements de transition obtenus par annotation d'un arbre de preuve en logique linéaire sont exactement les ordres partiels associés aux processus finis du réseau de Petri (voir [18]. Cela démontre d'une certaine manière la cohérence et la complétude de la démarche exposée au chapitre précédent.

3.2 Processus de réseaux de Petri

3.2.1 Processus et réseaux causaux

Un réseau causal [48] ou réseau d'occurrences déterministe [40, 4] est un réseau de Petri (étendu, car éventuellement de taille infinie) acyclique *sauf* (1-borné) où chaque place possède au plus une transition en entrée et en sortie.

Un processus de réseau de Petri R est un réseau causal O associé à un morphisme de graphes biparties de O vers R, c'est-à-dire une application des places de O vers les places de R, des transitions de O vers les transitions de R, et préservant les relations exprimées par les arcs.

Dans notre travail, nous ne considèrerons que les processus *finis*. Formellement, les réseaux causaux et les processus sont définis comme suit.

Définition 12 (Réseau causal) *Un réseau causal est un réseau de Petri $O = (B, E, F)$ tel que :*

1. $F : B \times E \cup E \times B \longrightarrow \{0, 1\}$ *: les transitions ne peuvent produire (respectivement consommer) que des jetons uniques.*
2. $\forall e \in E, {}^\bullet e \neq \emptyset \neq e^\bullet$*, c'est-à-dire toute transition possède au moins une place d'entrée et une place de sortie.*
3. $\forall b \in B,\ |{}^\bullet b| \leq 1$ *et* $\forall b \in B,\ |b^\bullet| \leq 1$ *: les places sont des entrées et des sorties d'au plus une seule transition.*
4. F^+ *est acyclique où* $F^+ \subseteq B \cup E \times B \cup E$ *est la fermeture transitive de* F*.*

Définition 13 (Processus de réseau de Petri) *Un processus de réseau de Petri $R = (P, T, W)$ est une paire (O, λ) où le triplet $O = (B, E, F)$ est un réseau causal et λ est une application $\lambda : B \cup E \longrightarrow P \cup T$ telle que :*

1. $\lambda(B) \subseteq P$, $\lambda(E) \subseteq T$ *: λ associe les transitions (respectivement les places) de O aux transitions (respectivement aux places) de R.*

2. $\forall e \in E : \lambda({}^\bullet e) = {}^\bullet\lambda(e)$, $\lambda(e^\bullet) = \lambda(e)^\bullet$: *λ préserve l'ensemble des places d'entrée et l'ensemble des places de sortie des transitions.*
3. $\forall e \in E$, $\forall p \in P : W(p, \lambda(e)) = |\lambda^{-1}(p) \cap {}^\bullet e| \bigwedge W(\lambda(e), p) = |\lambda^{-1}(p) \cap e^\bullet|$: *λ préserve les arités d'entrée et de sortie des transitions.*
4. *Soit M_{in} un marquage de R, alors (O, λ) est un processus du réseau marqué (R, M_{in}) seulement si $\forall p \in P$, $M_{in}(p) = |\{b \in B : {}^\bullet b = \emptyset \bigwedge \lambda(b) = p\}|$. Cela signifie que les places initiales de O correspondent de manière bijective aux jetons du marquage initial M_{in}.*

Par la suite, (O, λ) est un processus de réseau marqué (R, M_{in}) avec $R = (P, T, W)$ et $O = (B, E, F)$. Une transition $e \in E$ est appelé un *événement* et peut être vu comme (une occurrence de) un tir de la transition $\lambda(e) \in T$. Une place $b \in B$ modélise un jeton dans la place $\lambda(b) \in P$.

Remarque 1 Un processus $((B, E, F), \lambda)$ est défini « à isomorphisme près » : les ensembles B et E des places et des transitions peuvent être substitués par n'importe quels ensembles B' et E' d'intersection vide au moyen de bijections $B \longrightarrow B'$ et $E \longrightarrow E'$. Ces bijections induisent de façon unique une relation F' image de F et un morphisme λ' compatible avec le morphisme λ. Le processus $((B', E', F'), \lambda')$ obtenu est le même que celui de départ « à isomorphisme près ». En particulier dans une représentation canonique d'un processus, B pourrait être défini par $e^\bullet = \cup_{p \in t^\bullet}\{(e, p)\} \times [W(t, p)]$, pour tout $e \in E$ avec $\lambda(e) = t$. Pour tout entier k, nous notons $[k]$ l'ensemble $[k] = \{1, 2, \ldots, k\}$.

3.2.2 B-coupe, ordre partiel étiqueté et marquage associé

Soit F^* la fermeture transitive-réflexive de F. Comme F^+ est acyclique, F^* (respectivement F^+) est un ordre partiel (respectivement un ordre partiel strict) que nous notons $\leq_F$ (respectivement $<_F$).

Définition 14 (B-coupe) *Une B-coupe est un ensemble de places $C \subseteq B$ tel que deux places de C sont incomparables pour $\leq_F$ (propriété Cut1 ci-dessous) et est un sous-ensemble maximal de B pour cette propriété (propriété Cut2).*

Nous notons par $Cut(O)$ l'ensemble des B-coupes de O et ainsi si $C \subseteq B$:

$$C \in Cut(O) \Longleftrightarrow \begin{cases} \textbf{Cut1} & : b, b' \in C \Longrightarrow \neg(b <_F b' \bigvee b' <_F b) \\ \textbf{Cut2} & : b \notin C \Longrightarrow \exists b' \in C : (b <_F b' \bigvee b' <_F b) \end{cases}$$

Un ensemble de places qui ne satisfait que la condition Cut1 est appelé une **antichaîne**.

Une place sans transition d'entrée (respectivement transition de sortie) est appelée minimale (respectivement maximale). Les ensembles des places minimales et maximales sont notés $Min(O)$ et $Max(O)$:

$$Min(O) = \{b, {}^\bullet b = \emptyset\}, Max(O) = \{b,\ b^\bullet = \emptyset\}$$

$Min(O)$ et $Max(O)$ sont des exemples de B-coupes : $Min(O) \in Cut(O)$ et $Max(O) \in Cut(O)$.

Le passé d'une B-coupe C est le réseau d'occurrences $\downarrow C = (B_C, E_C, F_C)$ avec $B_C = \{b \in B | \exists b' \in C, b \leq_F b'\}$, $E_C = \{e \in E | \exists b \in C, e \leq_F b\}$ et $F_C = F/_{B_C \cup E_C}$. Avec l'application $\lambda_C = \lambda/_{B_C \cup E_C}$, la paire $(\downarrow C, \lambda_C)$ est un processus de R.

Ordre partiel étiqueté associé à un processus

À chaque processus (O, λ) nous associons l'ordre partiel étiqueté par les transitions $Op(O, \lambda) = (E, \leq_F /_{E \times E}, \lambda/_E)$ induit par les restrictions de $\leq_F$ et de λ à l'ensemble des événements. Nous notons aussi cet ordre partiel par $Op(O)$ lorsqu'il n'y a aucune ambiguïté. L'ensemble des extensions linéaires (ou linéarisations) de $Op(O, \lambda)$ est noté $Lin(Op(O, \lambda))$, formellement $Lin(Op(O, \lambda))$ est l'ensemble des mots de T défini par :

$$Lin(Op(O, \lambda)) = \{\lambda(e_1)...\lambda(e_i)...\lambda(e_j)...\lambda(e_n) \in T^*,\ E = \{e_1, ..., e_n\} \bigwedge (e_i <_F e_j \Longrightarrow i < j)\}$$

Marquages associés aux B-coupes

Soit $C \in Cut(O)$, nous associons à C un marquage de R (un multi-ensemble de places de R) noté $\mu(C)$, avec $\mu(C) : P \to \mathbb{N}$, défini par :

$$\mu(C)(p) = |\lambda^{-1}(p) \cap C|$$

Cela signifie que les places de $\lambda^{-1}(p) \cap C$ correspondent de manière bijective aux jetons de $\mu(C)$ dans la place p.

La condition 4 de la définition 13 d'un processus, qui dit que $M_{in}(p) = |\{b \in Min(O) : \lambda(b) = p\}|$, peut être reformulée par $\mu(Min(O)) = M_{in}$. Ce qui signifie que le marquage correspondant à la coupe minimale du réseau causal O est le marquage initial du réseau marqué (R, M_{in}). Notons que le multi-ensemble $\mu(D)$ peut être défini pour tout sous-ensemble $D \subset B$ même si D n'est pas une coupe, en particulier si D est une antichaîne de O.

Une propriété importante des B-coupes est que les marquages associés sont accessibles dans le réseau de Petri [4] : si (O, λ) est un processus de (R, M_{in}) et que C est une B-coupe alors le marquage $\mu(C)$ est accessible dans (R, M_{in}). De plus, $\mu(C)$ est accessible depuis M_{in} pour toute séquence σ de la linéarisation de l'ordre partiel étiqueté du réseau d'occurrences $Lin(Op(\downarrow C))$. Ce qui se traduit par :

$$\forall C \in Cut(O), \forall \sigma \in Lin(Op(\downarrow C)) : M_{in}[\sigma > \mu(C)$$

Tout marquage accessible d'un réseau de Petri et toute séquence de tir y menant peuvent ainsi être extraits d'un processus. Nous utilisons la propriété générale associant des séquences de tirs à des processus [4] :

$$\forall \sigma \in T^* : M[\sigma > M' \Longleftrightarrow \exists (O, \lambda) : \mu(Min(O)) = M, \mu(Max(O)) = M', \sigma \in Lin(Op(O, \lambda))$$

Ce qui signifie que les séquences de tirs correspondent à des linéarisations de processus. D'autres propriétés importantes des processus les lient aux séquences de pas et aux franchissements d'ordres partiels (voir [4]).

3.3 Construction d'un processus fini

3.3.1 Ajout d'occurrence de transition

La construction des arbres de preuve se fait de façon itérative. Pour montrer l'équivalence, nous allons nous appuyer sur une construction itérative équivalente des processus de réseau de Petri.

À chaque pas de la construction de la preuve canonique d'un séquent de logique linéaire (règle d'introduction à gauche de l'implication linéaire), nous rajoutons une transition à l'ordre partiel courant. De la même manière, la construction d'un processus fini est basée sur l'ajout itératif d'une occurrence de transition au processus courant.

Définition 15 (Ajout d'occurrence de transition) *Soit $t \in T$ et $D \subseteq Max(O)$ un sous-ensemble de places maximales tel que $\mu(D) = Pre(t)$, ou de manière équivalente $\lambda(D) = {}^\bullet t \bigwedge(\forall p \in {}^\bullet t, |\lambda^{-1}(p) \cap D| = W(p,t))$. Dans ce cas, nous pouvons « ajouter une occurrence de t à (O,λ) après D », ce qui donne un nouveau processus de R noté $(O',\lambda') = (O,\lambda).(D,t)$ où $O' = (B', E', F')$ et λ' sont définis par :*

1. *$E' = E \uplus \{e\}$, où e est un nouvel événement ($\uplus$ représente l'union disjointe).*
2. *$B' = B \uplus B_e$ avec une bijection $r : B_e \to \cup_{p \in t^\bullet} \{p\} \times [W(t,p)]$*
3. *$F' = F \cup \{(b,e), b \in D\} \cup \{(e,b), b \in B_e\}$*
4. *$\lambda'/B \cup E = \lambda$, $\lambda'(e) = t$, $\lambda'(b) = p$ si $b \in B_e$ et $r(b) = (p,i)$.*

L'introduction de B_e et de r est un artifice qui assure que $B \cap B_e = \emptyset$ et qui permet de définir λ' sur B_e dans le point 4 ci-dessus. Le lemme suivant est une conséquence directe des définitions :

Lemme 1 *Le couple $(O',\lambda') = (O,\lambda).(D,t)$ défini ci-dessus est un processus de R qui satisfait les points suivants :*

1. *B_e est une antichaîne telle que $\mu(B_e) = Post(t)$.*
2. *$Max(O') = (Max(O) - D) \uplus B_e$.*
3. *$\mu(Max(O')) = \mu(Max(O)) - Pre(t) + Post(t)$.*

Notons que $Min(O') = Min(O)$.

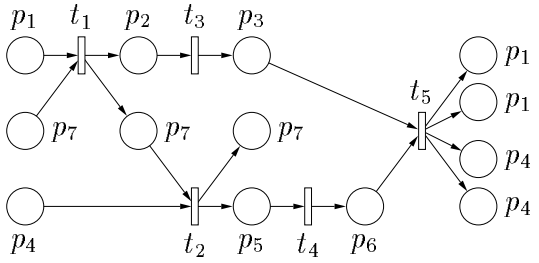

FIG. 3.1: Exemple de process de réseau de Petri

3.3.2 Algorithme de construction

Un processus de réseau de Petri peut être construit en ajoutant itérativement les transitions comme nous l'avons défini précédemment.

Soit (O, λ) un processus de R où $O = (B, E, F)$, soit $n = |E|$ et soit $e_1 ... e_n$ une énumération d'éléments de E telle que $e_i <_F e_j \Longrightarrow i < j$. Clairement $\sigma = \lambda(e_1)..\lambda(e_n)$ est une linéarisation de l'ordre partiel $Op(O, \lambda)$: $\sigma \in Lin(Op(O, \lambda))$.

Soit $O_i = (B_i, E_i, F_i)$ $(i = 1..n)$ la séquence de réseaux causaux définie par $E_i = \{e_1, ...e_{i-1}, e_i\}$, $B_i = {}^\bullet E_i^\bullet$, $F_i = F/_{B_i \cup E_i}$, et soit $\lambda_i = \lambda/_{B_i \cup E_i}$, alors le lemme suivant est une conséquence directe des définitions :

Lemme 2 *Pout tout $i = 1...n$, (O_i, λ_i) est un processus de R qui peut être défini par ajout de la transition $\lambda(e_i)$ à (O_{i-1}, λ_{i-1}) après ${}^\bullet e_i$: $(O_i, \lambda_i) = (O_{i-1}, \lambda_{i-1}).({}^\bullet e_i, \lambda(e_i))$. Qui plus est, $(O, \lambda) = (O_n, \lambda_n)$.*

Le processus fini (O, λ) de la figure 3.1 a été construit à partir du réseau de Petri de la figure 2.1 avec comme marquage initial $\mu(Min(O)) = p_1, p_4, p_7$, comme marquage final $\mu(Max(O)) = p_1, p_1, p_4, p_4, p_7$ et comme séquence de franchissements de transitions $\sigma(\tau) \in Lin(Op(O, \lambda)) = t_1; t_2; t_3; t_4; t_5$.

3.4 Des arbres de preuves canoniques aux processus finis

Nous avons vu précédemment que les algorithmes de construction des arbres de preuves canoniques et des processus finis de réseaux de Petri étaient basés sur une étape itérative de franchissement de transition. Nous allons montrer comment nous obtenons un processus de réseau de Petri fini à partir d'un arbre de preuve canonique. Nous partons d'un arbre de preuve canonique et allons construire un processus en suivant pas à pas la méthode itérative de preuve.

Dans un arbre de preuve canonique τ prouvant un séquent $M_0, l_0 \vdash M_n$, les étapes initiales et finales sont uniques et entièrement définies par M_0 et M_n (respectivement marquages initial et final). L'arbre de preuve est alors caractérisé par l'ordre dans lequel

les franchissements de transitions de la liste l_0 sont éliminés de manière itérative. La séquence de transitions correspondante à la séquence d'éliminations dans l'arbre de preuve canonique τ se note $\sigma(\tau)$.

Nous associons un processus à chaque arbre de preuve canonique τ d'un séquent décrivant un problème d'accessibilité tel que le séquent et le processus ont les mêmes marquages inital et final, et tel que $\sigma(\tau)$ est une linéarisation de l'ordre partiel étiqueté associé au processus.

La construction se déroule comme suit : soit τ l'arbre de preuve du séquent $M_0, l_0 \vdash M_n$, à chaque pas i $(1 \leq i \leq n)$ de la construction de l'arbre de preuve, caractérisé par un séquent courant $\mathcal{M}_i, l_i \vdash M_n$ et un fragment courant de l'arbre τ_i, nous définissons un processus (O_i, λ_i) tel que $\mu(Min(O_i)) = M_0$, $\mu(Max(O_i)) = \eta(\mathcal{M}_i)$ et tel que la séquence de transitions $\sigma(\tau_i)$ est une linéarisation de l'ordre partiel $Op(O_i, \lambda_i)$.

Processus initial : Démarrons avec le séquent $\mathcal{M}_0, l_0 \vdash M_n$. Le processus (O_0, λ_0) est défini par : $O_0 = (B_0, \emptyset, \emptyset)$ avec B_0 et λ_0 tels que $\mu(B_0) = M_0$.

Pas itératif $i+1$: Nous avons construit un processus (O_i, λ_i) tel que $\mu(Min(O_i)) = M_0$, $\mu(Max(O_i)) = \eta(\mathcal{M}_i)$ et $\sigma(\tau_i)$ est une linéarisation de l'ordre partiel étiqueté $Op(O_i, \lambda_i)$. Soit t_{i+1} la transition éliminée au pas $i+1$ de la preuve. Cela signifie que $Pre(t_{i+1}) \sqsubseteq \eta(\mathcal{M}_i)$. Puisque $\mu(Max(O_i)) = \eta(\mathcal{M}_i)$, nous pouvons trouver (au moins, voir remarque 3 ci-après) une antichaîne $C_{i+1} \subseteq Max(O_i)$, telle que $\mu(C_{i+1}) = Pre(t_{i+1})$. Les hypothèses de la construction du paragraphe 3.3.1 sont satisfaites et nous pouvons donc ajouter une occurrence de t_{i+1} à (O_i, λ_i) après C_{i+1}. Le processus obtenu est le processus construit au pas courant : $(O_{i+1}, \lambda_{i+1}) = (O_i, \lambda_i).(C_{i+1}, t_{i+1})$.

Suite au lemme 1 nous obtenons $\mu(Max(O_{i+1})) = \mu(Max(O_i)) - Pre(t_{i+1}) + Post(t_{i+1})$. Soit $\mathcal{M}_{i+1}, l_{i+1} \vdash M_n$ le séquent après avoir éliminé t_{i+1}, suite à la règle 2.12, nous savons que $\eta(\mathcal{M}_{i+1}) = \eta(\mathcal{M}_i) - Pre(t_{i+1}) + Post(t_{i+1})$. Nous avons prouvé que $\mu(Max(O_{i+1})) = \eta(\mathcal{M}_{i+1})$. De plus l_{i+1} est déduit de l_i en enlevant un franchissement de t_{i+1}. Soit τ_{i+1} le nouveau fragment de l'arbre de preuve canonique, comme $\sigma(\tau_i)$ est une linéarisation de l'ordre partiel étiqueté $Op(O_i, \lambda_i)$, $\sigma(\tau_{i+1})$, par construction $\sigma(\tau_{i+1})$ est une linéarisation de l'ordre partiel étiqueté $Op(O_i, \lambda_{i+1})$.

Processus final : Après le pas n, nous avons construit un processus (O_n, λ_n) tel que : $\mu(Max(O_n)) = M_n$ et $\sigma(\tau) \in Lin(Op(O_n, \lambda_n))$.

Nous avons donc prouvé le théorème suivant :

***Théorème* 1** *Soit $M_0, l_0 \vdash M_n$ un séquent exprimant l'accessibilité d'un marquage M_n depuis un marquage M_0 dans un réseau de Petri R. Soit τ une preuve canonique de ce séquent et soit $\sigma(\tau)$) la séquence d'étiquettes de transitions caractérisant la séquence d'élimination de tir dans l'arbre de preuve. Il est alors possible de construire au moins*

un processus (O, λ) *de* R *tel que :* $\mu(Min(O)) = M_0$, $\mu(Max(O)) = M_n$ *et* $\sigma(\tau) \in Lin(Op(O, \lambda))$.

Remarque 2 Quand il y a plus d'un jeton dans une place, il peut y avoir un conflit jetons si la place possède en sortie plusieurs transitions. Le processus peut donc être prolongé de différentes façons qui ne sont pas toujours équivalentes suivant le jeton utilisé. C'est pourquoi à un arbre de preuve, il est parfois possible d'associer plus d'un processus.

Exemple Pour le réseau de Petri de la figure 2.1, l'arbre de preuve canonique du séquent 2.4 peut être prouvé en éliminant successivement (après le pas initial) les tirs de t_1,t_2, t_3, t_4 et t_5. La preuve se termine par le pas final. Pour un des arbres de preuve possibles, τ', et la séquence de tir associée $\sigma(\tau') = t_1; t_3; t_2; t_4; t_5$, nous obtenons le processus qui est représenté sur la figure 3.1.

Ce processus est unique car c'est seulement pour le marquage final que certaines places contiennent plus d'un jeton. Pour chaque pas de la preuve, il n'y a donc aucune ambiguïté dans la construction du processus : il n'y a qu'une seule façon d'ajouter une occurrence de tir de la transition considérée.

Pour le marquage initial les deux transitions t_1 et t_2 sont en conflit. En sélectionnant dans la preuve l'élimination du tir de t_1 avant celle de t_2, ou le contraire, nous aboutissons à deux arbres de preuve canoniques caractérisés par deux séquences différentes σ. C'est la même chose pour les processus. En construisant le processus de manière itérative guidée par σ, nous construisons en effet les processus correspondant à la même résolution du conflit entre les transitions.

3.5 Des processus finis aux arbres de preuves canoniques

Dans le paragraphe précédent (3.4), nous avons montré qu'à partir de n'importe quel arbre de preuve canonique nous pouvions construire un processus. Dans ce paragraphe, nous démontrons l'inverse : à partir de n'importe quel processus fini, nous pouvons construire un arbre de preuve canonique.

Soit (O, λ) un processus de R où $O = (B, E, F)$, soient $M_0 = \mu(Min(O))$ et $M_n = \mu(Max(O))$. Soient $e_1 ... e_n$ une énumération d'éléments de E telle que $e_i <_F e_j \Longrightarrow i < j$, et soit (O_i, λ_i) avec $i = 1, n$ la séquence associée des processus définie dans le paragraphe 3.3.2. Soit $\sigma = \lambda(e_1)..\lambda(e_n)$ ($\sigma \in Lin(Op(O, \lambda))$). Soit l_0 la liste non ordonnée des éléments de σ. Nous construisons itérativement un arbre de preuve τ du séquent $M_0, l_0 \vdash M_n$ tel que $\sigma(\tau) = \sigma$.

Fragment initial de l'arbre de preuve : Il est formé par l'étape initiale (paragraphe 2.4.3) et il est complètement déterminé par $Min(O)$.

Pas itératif $i+1$ **:** Nous avons construit les i premiers pas itératifs de l'arbre de preuve. Soit τ_i ce fragment. Le séquent de droite en haut de l'arbre τ_i est $\mathcal{M}_i, l_i \vdash M_n$. Les hypothèses d'induction sont $\eta(\mathcal{M}_i) = \mu(Max(O_i))$, $\sigma(\tau_i) = \lambda(e_1)..\lambda(e_i)$ et l_i est la liste non ordonnée des éléments de $\lambda(e_{i+1})..\lambda(e_n)$.

Suite au lemme 2, nous obtenons $(O_{i+1}, \lambda_{i+1}) = (O_i, \lambda_i).(^\bullet e_{i+1}, \lambda(e_{i+1}))$. Soit $t_{i+1} = \lambda(e_{i+1})$, par construction nous obtenons $^\bullet e_{i+1} \subseteq Max(O_i)$ et $\mu(^\bullet e_{i+1}) = Pre(t_{i+1})$. Par conséquent $Pre(t_{i+1}) \sqsubseteq \mu(Max(O_i))$ et $Pre(t_{i+1}) \sqsubseteq \eta(\mathcal{M}_i)$. Nous pouvons ajouter au fragment de l'arbre de preuve τ_i la règle 2.12 pour la transition t_{i+1}. Le nouveau séquent $\mathcal{M}_{i+1}, l_{i+1} \vdash M_n$ au sommet de l'arbre est tel que les hypothèses d'induction sont vérifiées pour $i+1$ (les calculs de $\eta(\mathcal{M}_{i+1})$ et $\mu(Max(O_{i+1}))$ sont identiques à ceux du paragraphe 3.4).

Arbre final : Quand la séquence $e_1 \ldots e_n$ a été complètement explorée, nous procédons à l'étape finale (paragraphe 2.4.3) pour obtenir un arbre de preuve canonique τ tel que $\sigma(\tau) = \sigma$. Pour chaque linéarisation de $Op(O, \lambda)$, un arbre de preuve canonique différent peut être obtenu. Pour une linéarisation donnée, l'arbre de preuve canonique est unique car les pas itératifs sont complètement définis par les noms des transitions.

Théorème 2 *Pour chaque processus fini (O, λ) tel que $\mu(Min(O)) = M_0$, $\mu(Max(O)) = M_n$ et pour chaque $\sigma \in Lin(Op(O, \lambda))$, il existe exactement un arbre de preuve canonique τ du séquent $M_0, l_0 \vdash M_n$.*

Exemple Pour le réseau de Petri de la figure 2.1, un processus fini démarrant avec un marquage initial composé d'un jeton dans les places s_1, s_4 et s_7 et terminant avec un marquage final composé de deux jetons dans chacune des places s_1 et s_4 et de un jeton dans la place s_7 est représenté figure 3.1. C'est un processus pour lequel la transition t_1 est franchie avant t_2. Différents arbres de preuve canonique peuvent lui être associés : par exemple l'arbre τ' avec $\sigma(\tau') = t_1; t_3; t_2; t_4; t_5$ ou l'arbre τ" avec $\sigma(\tau") = t_1; t_2; t_3; t_4; t_5$. Ils sont tous équivalents quant au processus et à l'ordre partiel généré.

3.6 Des relations de précédence entre les règles aux processus de réseau de Petri

Nous venons de montrer qu'à partir d'un processus, nous pouvions obtenir au moins un arbre de preuve par linéarisation de l'ordre partiel associé au processus. Au chapitre précédent, nous avions montré que par un processus d'étiquetage, nous pouvions, à partir d'un arbre de preuve, obtenir un ensemble de relations de précédence entre les applications de la règle d'élimination de l'implication linéaire à gauche. La question qu'il est possible de se poser maintenant est la suivante : les relations d'ordre obtenues à partir des relations

de précédence sont-elles exactement les mêmes que celles qui sont obtenues à partir des processus ?

La réponse est « oui ». Cela découle du fait que la démarche de construction itérative des processus à partir d'un arbre de preuve est en fait identique à celle de l'arbre de preuve annoté. Au préalable, il nous faut remarquer que dans la représentation d'un processus O, chaque place ($b \in B$) ayant au plus une transition ($e_i \in E$) en entrée et une transition ($e_j \in E$) en sortie, elles peuvent être remplacées par des arcs reliant directement la transition e_i à la transition e_j. Les places de $Min(O)$ peuvent être supposées reliées à des nœuds initiaux et les places de $Max(O)$ à des nœuds finaux comme nous l'avons fait lors de la construction des relations de précédence entre les applications des règles.

Pour identifier un processus à un graphe de précédence, il suffit alors de nommer chaque élément e de E par $\lambda(e)$ en ajoutant en exposant le nombre de fois que la transition $\lambda(e)$ a été franchie auparavant dans la séquence σ, linéarisation de l'ordre partiel utilisée pour la construction du processus. Nous ajoutons alors les événements initiaux (production de $Min(O)$) et finaux (consommation de $Max(O)$) pour compléter le graphe et les deux représentations seront identiques. Il est alors possible de considérer que les éléments de B sont nommés par les événements qui les produisent. Nous utiliserons d'ailleurs cette idée au pragraphe suivant.

Nous venons de montrer qu'il est possible de donner aux processus de réseaux de Petri la même forme que les graphes de précédence entre les applications des règles. Mais il subsiste une difficulté. En effet, à partir d'un arbre de preuve, il est possible d'obtenir plusieurs arbres de preuve étiquetés lorsque des confits jetons apparaissent. Nous en avons donné un exemple au chapitre précédent. Lorsqu'il s'agit de construire un graphe de précédence entre les applications des règles, un conflit jetons apparaît lorsqu'il y a plusieurs façons d'extraire les atomes de $Pre(t_{i+1})$ de la liste μ_i pour un franchissement d'une transition t_{i+1} donnée. Pour la construction du processus, il y a plusieurs possibilités de prolonger un processus donné avec un franchissement d'une transition t_{i+1} donnée lorsqu'il y a plusieurs façons d'extraire $Pre(t_{i+1})$ de $Max(O_i)$. Comme $\mu(Max(O_i)) = \eta(\mu_i)$, il est clair qu'il s'agit de deux visions du même mécanisme. Il y a donc bien identité entre l'ensemble des graphes de précédence et l'ensemble des processus finis. Et si nous prenons les processus de réseaux de Petri comme l'outil de référence décrivant les ordres partiels entre les franchissements de transition, ce résultat montre la cohérence et la complétude de la démarche présentée au chapitre précédent. Nous rappelons que pour les travaux de Gunter et Gehlot étendus par L.A. Künzle, il n'y avait cohérence et complétude que pour la classe des processus factorisables sous la forme « série-parallèle ».

3.7 Synthèse sur l'équivalence

Nous venons de montrer dans les sections 3.4, 3.5 et 3.6 précédentes que les ensembles de relations de précédence obtenus par les preuves canoniques de séquents sont des pro-

cessus finis de réseau de Petri et que réciproquement, à partir d'un processus fini, un arbre de preuve canonique peut être construit. Ceci justifie bien la correspondance entre des applications de la règle $\multimap_L$ dans les arbres de preuve et les tirs de transitions d'un réseau de Petri. C'est aussi une preuve que les preuves canoniques sont complètes, c'est-à-dire qu'elles génèrent tous les ordres partiels possibles parmi les tirs de transitions.

Les résultats donnent donc une autre approche pour construire les processus de réseau de Petri qui est similaire à celle basée sur les dépliages de réseau de Petri. Quelles sont donc les différences ? La plus importante est que l'approche basée sur la logique linéaire produit des processus *finis* entre deux marquages prédéfinis. Comme nous ne travaillons qu'avec des processus finis, cette approche est très utile pour prouver des propriétés d'accessibilité plutôt que de sûreté. Par exemple, elle n'est pas adaptée pour prouver qu'un état est inacessible car le fait que nous ne prouvions pas un séquent ne signifie pas qu'il ne soit pas prouvable. Par contre, il est possible de prouver qu'un système peut être contrôlé afin d'avoir un certain comportement prédéfini. Le fait que seuls les processus finis soient concernés est la raison pour laquelle l'approche est exactement la même, que le réseau de Petri soit sauf ou non, ou même qu'il soit non borné. L'accessibilité n'est étudiée que dans le cas d'une liste finie et prédéfinie de franchissements de transitions. Avoir des transitions *sources*, c'est-à-dire des transitions sans place d'entrée, dans la liste n'est pas un problème car le nombre de fois qu'elles seront franchies est défini par le séquent d'accessibilité. Nous devons simplement utiliser l'élément neutre « $\mathbf{1}$ » associé au connecteur $\otimes$.

Tous ces points sont des conséquences directes de la décidabilité du fragment multiplicatif de la logique linéaire parce que nous avons choisi une traduction des réseaux de Petri dans ce fragment sans avoir ajouté d'axiomes. L'élimination des coupures est ainsi préservée.

3.8 Séquent caractéristique d'un ordre partiel

Nous venons de voir qu'à un arbre de preuve pouvaient être associés plusieurs processus. Cela signifie que plusieurs ensembles de relations de précédence entre les applications des règles de la logique linéaire peuvent être associés à un arbre de preuve. Le processus d'étiquetage de l'arbre de preuve n'est donc pas unique comme cela a été montré avec l'exemple sur le conflit jetons dans le paragraphe 2.5.2.

Est-il possible en modifiant l'écriture du séquent de n'avoir qu'un ensemble de relations de précédence ? Cela permettrait d'avoir une notation, sous la forme d'un séquent en logique linéaire, pour un ordre partiel donné. La procédure est la suivante.

Une fois un arbre de preuve canonique étiqueté construit, nous conservons les étiquettes de l'arbre de façon à lever toute ambiguïté concernant le choix des jetons. Les ambiguïtés seront effectivement levées en considérant ensuite que les étiquettes font partie du nom. Pour cela, nous remplaçons les parenthèses par des crochets. Ainsi l'atome $c[t_i^j]$ produit par le j^e franchissement de t_i sera différent de l'atome $c[t_k^l]$ produit dans la même place

c, mais par le l^e franchissement de t_k. La règle identité ne s'appliquera pas entre deux atomes produits par deux franchissements différents.

Pour pouvoir travailler de cette façon, tous les atomes seront étiquetés par le franchissement de transition qui les a produits. Il n'est bien évidemment possible d'inscrire ces annotations qu'après la construction d'un arbre de preuve complet annoté. Nous annoterons directement les marquages initiaux M_0 ainsi que les atomes des formules $Post(t)$. Les atomes finaux M_n et ceux des formules $Pre(t)$ seront annotés par les étiquettes des atomes avec lesquels ils ont été unifiés dans l'arbre de preuve au moment de l'application de la règle identité.

Considérons l'exemple du conflit jetons sur la figure 2.5 exposé au paragraphe 2.5.2. Nous obtenons, pour les deux ensembles de relations de précédence des figures 2.6 et 2.7, les deux séquents annotés suivants :

$$a[I^1]\otimes b[I^2], \underbrace{a[I^1]\multimap c[t_1^1]}_{t_1^1}, \underbrace{b[I^2]\multimap c[t_2^1]}_{t_2^1}, \underbrace{c[t_1^1]\multimap d[t_3^1]}_{t_3^1}, \underbrace{c[t_2^1]\multimap e[t_4^1]}_{t_4^1} \vdash d[t_3^1]\otimes e[t_4^1] \tag{3.1}$$

$$a[I^1]\otimes b[I^2], \underbrace{a[I^1]\multimap c[t_1^1]}_{t_1^1}, \underbrace{b[I^2]\multimap c[t_2^1]}_{t_2^1}, \underbrace{c[t_2^1]\multimap d[t_3^1]}_{t_3^1}, \underbrace{c[t_1^1]\multimap e[t_4^1]}_{t_4^1} \vdash d[t_3^1]\otimes e[t_4^1] \tag{3.2}$$

Nous pouvons remarquer que les relations de précédence peuvent être déduites directement du séquent. Elles sont données par la liste des franchissements. En effet, l'étiquette du franchissement est portée par les atomes produits (ceux de $Post(t)$) et chaque atome consommé (ceux de $Pre(t)$) porte l'étiquette de l'événement qui l'a produit. Nous avons donc tous les arcs entrants dans le nœud événement. Les arcs aboutissant aux nœuds terminaux sont donnés par les étiquettes du marquage final. Il suffit de rajouter un nœud terminal par atome du marquage final.

À partir de l'étiquetage (vu au chapitre précedent) et du graphe de précédence, nous pouvons ainsi définir un *séquent caractéristique* d'un ordre partiel c'est-à-dire d'un ensemble de relations de précédence. Les séquents 3.1 et 3.2 sont les séquents caractéristiques des deux ordres partiels pour un scénario donné. Comme nous l'avons vu précédemment, les deux correspondent à un arbre de preuve non annoté unique.

Définition 16 (Séquent caractéristique) *Le séquent caractéristique d'un ensemble de relations de précédence est obtenu en concaténant aux noms des places les noms des événements ayant produit le jeton dans la place.*

Lors de la preuve d'un séquent caractéristique et de la construction des relations de précédence associées, nous aurons deux annotations. L'annotation entre crochets est une extension du nom de l'atome et doit être prise en compte lors de l'application des règles du calcul des séquents. L'annotation entre parenthèses est celle qui a été définie au chapitre 1 et qui permet de générer les relations de précédence.

Ainsi, par exemple, la preuve du séquent 3.1 va donner la feuille

$$\frac{}{c[t_1^1](t_1^1) \vdash c[t_1^1](t_3^1)}\ \mathrm{id}$$

qui induit une relation de précédence entre t_1^1 et t_3^1 car l'atome $c[t_1^1]$ doit être produit par t_1^1 et consommé par t_3^1.

Par contre le séquent $c[t_1^1](t_1^1) \vdash c[t_2^1](t_4^1)$ ne peut être prouvé car $c[t_2^1]$ est différentié de $c[t_1^1]$. Nous voyons donc que pour un séquent caractéristique, toutes les relations de précédence sont bien figées.

3.9 Composition de processus

Nous venons, dans la section précédente, d'introduire une notation sous la forme d'un séquent de logique linéaire pour un processus fini de réseau de Petri. Les règles binaires (ayant deux séquents comme prémisses) du calcul des séquents permettent la composition de deux séquents pour prouver un séquent complexe (ou la décomposition d'un séquent complexe en deux sous-séquents). Il est donc naturel d'utiliser cette approche pour composer (ou décomposer) des processus finis de réseau de Petri. C'est ce que nous allons faire dans cette section en introduisant trois règles de composition. Auparavant nous allons revenir sur le travail de L.A. Künzle car, même si nos règles de composition parallèle et de composition séquentielle sont très proches des siennes, grâce aux annotations des séquents caractéristiques, nous évitons d'introduire des relations de précédence parasites et nous ne nous restreignons pas aux ordres « série-parallèle ».

3.9.1 Retour sur les travaux de L.A. Künzle

Dans ses travaux, L.A. Künzle [32] a entrepris une caractérisation du parallélisme maximal à l'aide d'opérateurs séquence et parallèle. Il a étendu le travail de V. Gehlot qui s'appliquait à transformer des déductions en logique linéaire en réduisant l'utilisation des coupures dans le calcul des séquents. Mais il ne considérait que la structure du réseau pour caractériser le parallélisme maximal pour les franchissements de transitions. L'extension de Künzle consista donc à prendre en compte le parallélisme résultant des marquages. Il caractérisa ainsi ce qu'il appela le parallélisme dynamique.

Il a dû construire pour cela deux règles de calcul des séquents SEQ et SYNC (pour la composition de règles de base) exprimant respectivement l'exécution en séquence et l'exécution en parallèle de deux sous-scénarios de franchissements de transitions. Cependant, les relations SEQ et SYNC ne permettent d'extraire que des ordres partiels de type « série-parallèle ». En effet, la règle SEQ entre deux scénarios introduit globalement une relation de précédence entre tous les franchissements du premier scénario et tous les franchissements du deuxième.

Un point important à souligner est le fait que cette approche est compositionnelle. La composition de deux sous-scénarios par les deux relations est possible et une preuve valable pour un scénario reste valable après composition. Ce résultat est valable dans le cas structurel mais aussi dans le cas dynamique. La règle SEQ permet de composer deux scénarios en séquence et la règle SYNC permet de composer deux scénarios en parallèle. Au cœur de la règle SEQ se trouve la règle de coupure (Cut) et au cœur de la règle SYNC les règles d'élimination à droite et à gauche du connecteur fois ($\otimes_L$ et $\otimes_R$).

Les compositions se font toujours dans le cadre du fragment MILL. Nous n'avons pas utilisé la règle de coupure jusqu'à présent et les deux autres règles d'élimination de $\otimes_L$ et $\otimes_R$ ont été utilisées dans des cas particuliers pour la construction des arbres de preuve canoniques. Ici, nous allons nous appuyer sur ces trois règles pour introduire deux des trois compositions de processus que nous envisageons. La première composition est une composition en série avec un franchissement de transition réunissant les deux fragments et ne nécessite pas de nouvelle introduction de connecteur. La deuxième est une composition par mise en parallèle de deux processus. La troisième est une composition par mise en série de deux processus.

3.9.2 Composition par franchissement de transition

Le principe de cette composition est de montrer que deux preuves canoniques peuvent être composées par l'intermédiaire du franchissement d'une transition. Ceci est possible si la partie droite (conclusion) du séquent initial de l'une des preuves est exactement le marquage sensibilisant la transition voulue ($Pre(t)$) et si le marquage produit par le franchissement ($Post(t)$) est contenu dans la liste des atomes initiaux du séquent initial de l'autre preuve. Nous appellerons séquent *amont* le séquent contenant $Pre(t)$ et séquent *aval* celui contenant $Post(t)$. En effet, pour que la transition t soit franchie, il est nécessaire qu'elle soit sensibilisée. La preuve du séquent amont prouve que t peut être sensibilisée. Si nous voulons composer les deux séquents $\mathcal{M}_{1\_0}, l_1 \vdash M_{1\_n}$ et $\mathcal{M}_{2\_0}, l_2 \vdash M_{2\_n}$ par le franchissement d'une transition t, il est nécessaire que $M_{1\_n} = Pre(t)$ et que $Post(t) \subseteq \eta(\mathcal{M}_{2\_0})$.

Cette composition est fondée sur l'utilisation de la règle d'élimination à gauche de l'implication linéaire dans un cas plus général que celui que nous utilisons pour la construction des arbres de preuve canoniques. En effet, dans un arbre de preuve canonique comme ceux que nous avons présentés au chapitre précédent, le séquent prémisse gauche produit est un séquent qui ne comprend aucune formule implicative. Il ne comporte que les jetons consommés pour franchir la transition. Ce séquent comporte à gauche une liste d'atomes et à droite un monôme en $\otimes$ des mêmes atomes ($A, B \vdash A \otimes B$).

Nous nous proposons donc ici de l'utiliser dans une autre forme, où chaque séquent

prémisse est un séquent décrivant un problème d'accesssibilité, qui est la suivante :

$$\frac{\mathcal{M}_1, l_1 \vdash Pre(t) \quad \mathcal{M}_2, Post(t), l_2 \vdash M_3}{\mathcal{M}_1, \mathcal{M}_2, l_1, l_2, \underbrace{Pre(t) \multimap Post(t)}_{t} \vdash M_3} \multimap_L \tag{3.3}$$

Supposons que nous ayons construit au préalable un arbre de preuve étiqueté pour chacun des séquents prémisses :

$$\mathcal{M}_1, l_1 \vdash Pre(t) \quad \text{(séquent amont)} \tag{3.4}$$

et

$$\mathcal{M}_2, Post(t), l_2 \vdash M_3 \quad \text{(séquent aval)} \tag{3.5}$$

Nous allons définir comment un ensemble de relations de précédence peut être obtenu par composition pour le séquent suivant :

$$\underbrace{\mathcal{M}_1, \mathcal{M}_2}_{\mathcal{M}_0}, \underbrace{l_1, l_2, Pre(t) \multimap Post(t)}_{l_0} \vdash \underbrace{M_3}_{M_n} \tag{3.6}$$

Pour cela nous commençons par écrire les séquents caractéristiques correspondants aux arbres de preuves étiquetés construits pour les séquents 3.4 et 3.5.

L'étape suivante est un processus de réécriture des étiquettes associées aux atomes.

- Soit n le nombre d'atomes de $\mathcal{M}_1$, pour tous les atomes de $\mathcal{M}_2$, remplacer l'étiquette I^j par I^{j+n}. Cela permet de distinguer toutes les étiquettes initiales.
- Soit m le nombre de franchissements d'une transition t donnée dans la liste l_1. Le franchissement de t exprimé par la formule $Pre(t) \multimap Post(t)$ dans 3.6 sera étiqueté par t^{m+1}. On remplace donc toutes les étiquettes de $Post(t)$ par t^{m+1}. Chaque franchissement t^j de t dans 3.5 sera réétiqueté t^{j+m+1}.
- Soit k le nombre de franchissements de t_i ($t_i \neq t$) dans l_1, chaque franchissement t_i^j de t_i dans 3.5 sera ré-étiqueté t_i^{j+k}.

La réécriture ci-dessus ne change pas les relations de précédence entre les franchissements de transition. Elle confond les événements initaux de création des jetons de $Post(t)$ en un seul événement et évite que des étiquettes soient identiques dans les deux séquents 3.4 et 3.5.

Une fois ce réétiquetage effectué, on applique la règle 3.3 pour obtenir un séquent 3.6 étiqueté qui est un séquent caractéristique de l'ordre partiel composé. Comme nous l'avons remarqué au paragraphe 3.9, le graphe de précédence est obtenu directement.

Si nous avons par exemple deux séquents caractéristiques pour le séquent 3.4 et trois pour le séquent 3.5, nous obtiendrons six séquents caractéristiques et donc six ordres partiels pour le séquent global. La décomposition est donc une façon de lutter contre l'explosion combinatoire du nombre de processus solutions d'un problème d'accessibilité.

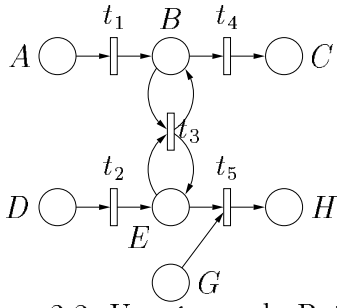

FIG. 3.2: Un réseau de Petri

Exemple de composition par franchissement de transition

Considérons le réseau de Petri de la figure 3.2. Un premier problème d'accessibilité peut être décrit par le séquent caractéristique suivant :

$$a[I^1], d[I^2], \underbrace{a[I^1] \multimap b[t_1^1]}_{t_1}, \underbrace{d[I^2] \multimap e[t_2^1]}_{t_2}, \underbrace{b[t_1^1] \otimes e[t_2^1] \multimap b[t_3^1] \otimes e[t_3^1]}_{t_3} \vdash \underbrace{b[t_3^1] \otimes e[t_3^1]}_{Pre(t_3)} \tag{3.7}$$

Un deuxième problème d'accessibilité peut être représenté par le séquent caractéristique suivant :

$$g[I^1], \underbrace{b[I^2] \otimes e[I^3]}_{Post(t_3]}, \underbrace{b[I^2] \multimap c[t_4^1]}_{t_4}, \underbrace{e[I^3] \otimes g[I^1] \multimap h[t_5^1]}_{t_5} \vdash c[t_4^1] \otimes h[t_5^1] \tag{3.8}$$

Nous voyons que ces deux séquents ont la structure adéquate pour les composer autour d'un franchissement de la transition t_3. Il faut d'abord réécrire les étiquettes du séquent 3.8. Il faut renommer l'étiquette associée à l'atome g pour qu'elle soit différente de celle de a dans 3.7 ainsi que celles des atomes de $Pre(t_3)$ et répercuter ces changements dans les formules associées à t_4 et t_5. Cela donne :

$$g[I^3], \underbrace{b[t_3^2] \otimes e[t_3^2]}_{Post(t_3)}, \underbrace{b[t_3^2] \multimap c[t_4^1]}_{t_4}, \underbrace{e[t_3^2] \otimes g[I^3] \multimap h[t_5^1]}_{t_5} \vdash c[t_4^1] \otimes h[t_5^1] \tag{3.9}$$

Nous pouvons remarquer que nous avons particularisé les relations de précédence du deuxième problème puisque maintenant les jetons b et e sont produits par le même événement (t_3^2).

Nous obtenons alors le séquent global suivant :

$$\begin{aligned} &a[I^1], d[I^2], g[I^3], \underbrace{a[I^1] \multimap b[t_1^1]}_{t_1}, \underbrace{d[I^2] \multimap e[t_2^1]}_{t_2}, \underbrace{b[t_1^1] \otimes e[t_2^1] \multimap b[t_3^1] \otimes e[t_3^1]}_{t_3}, \\ &\underbrace{b[t_3^1] \otimes e[t_3^1] \multimap b[t_3^2] \otimes e[t_3^2]}_{t_3}, \\ &\underbrace{b[t_3^2] \multimap c[t_4^1]}_{t_4}, \underbrace{e[t_3^2] \otimes g[I^3] \multimap h[t_5^1]}_{t_5} \vdash c[t_4^1] \otimes h[t_5^1] \end{aligned} \tag{3.10}$$

3.9.3 Composition séquentielle

Une autre règle binaire du fragment MILL est la règle de coupure. Nous n'utilisons pas cette règle dans la construction de l'arbre canonique car elle introduit une formule dans les séquents supérieurs qui n'existait pas dans le séquent inférieur, ce qui peut entraîner des difficultés lors de la construction des arbres de preuve (quelle formule introduire pour être efficace ?) et cela donnerait une infinité de façons de construire un arbre de preuve.

Par contre, la règle de coupure (2.6), utile pour permettre l'enchaînement de deux séquents linéaires, a été utilisée par L. A. Künzle afin de caractériser un scénario où deux transitions sont franchies en séquence : cela a donné la règle SEQ. La règle SEQ peut aussi être utilisée pour composer séquentiellement deux scénarios représentés par deux séquents.

Nous nous basons sur le même principe pour composer séquentiellement deux séquents. Ici la formule introduite par la règle de coupure va représenter un marquage partiel par lequel nous souhaitons passer. Ce marquage est le membre conclusion du séquent prémisse gauche. Soit M_4 ce marquage nous avons :

$$\frac{\mathcal{M}_1, l_1 \vdash M_4 \qquad M_4, \mathcal{M}_2, l_2 \vdash M_3}{\mathcal{M}_1, \mathcal{M}_2, l_1, l_2 \vdash M_3}\ \text{cut} \tag{3.11}$$

Nous procédons de la même manière que précédemment en construisant des séquents caractéristiques pour les deux séquents supérieurs, puis en effectuant un renommage des étiquettes pour le séquent supérieur de droite. Le séquent de droite n'a pas la forme habituelle de nos séquents puisque les jetons initialement disponibles n'apparaissent pas sous la forme d'une liste, mais sont structurés sous la forme d'une formule (monôme en $\otimes$ M_4) et d'une liste ($\mathcal{M}_2$). Comme la transformation du connecteur $\otimes$ en méta-connecteur « , » est réversible, cela ne pose pas de problème.

- Soit n le nombre d'atomes de $\mathcal{M}_1$ et soit m le nombre d'atomes de M_4. Pour tous les atomes de $\mathcal{M}_2$, il faut remplacer l'étiquette I^j par I^{j-m+n}. Cela permet de distinguer toutes les étiquettes initiales sans avoir de trou dans la numération.
- Soit k le nombre de franchissements de t_i dans l_1, chaque franchissement t_i^j dans l_2 sera re-étiqueté t_i^{j+k}.
- On reporte enfin dans le monôme M_4 du séquent de droite, et bien sûr dans les formules implicatives de l_2 où apparaissent des atomes de M_4, les étiquettes des atomes correspondant du monôme M_4 du séquent de gauche.

En appliquant la règle 3.11, nous obtenons un séquent caractéristique global.

La troisième étape de renommage consiste simplement à dire que les atomes consommés dans le deuxième séquent (ceux de M_4) sont les atomes produits dans le premier. Cette dernière approche est similaire à la première (composition par un franchissement de transition), mais elle est plus générale puisque le marquage M_4 ne correspond pas nécessairement à la précondition de franchissement d'une transition.

3.9.4 Composition parallèle

Le principe de cette composition est de montrer que deux preuves canoniques, dont les parties hypothèse et conclusion des séquents prouvés sont indépendantes, peuvent être composées afin de donner une preuve canonique.

Cette composition est basée sur la règle d'élimination à droite du connecteur *fois* ($\otimes$). Cette règle est utilisée dans un cas simple dans l'arbre de preuve canonique. En effet, nous ne l'utilisons que pour décomposer un séquent formé à gauche d'une liste d'atomes et à droite d'un monôme composé des mêmes atomes en une suite de séquents identité. Ici nous allons l'utiliser pour faire éclater un séquent d'accessibilité en deux séquents d'accessiblité :

$$\frac{\mathcal{M}_1, l_1 \vdash M_3 \quad \mathcal{M}_2, l_2 \vdash M_4}{\mathcal{M}_1, \mathcal{M}_2, l_1, l_2 \vdash M_3 \otimes M_4} \otimes_R \tag{3.12}$$

Cela revient à dire que de façon indépendante, nous pouvons produire le marquage partiel M_3 à partir de la liste de jetons $\mathcal{M}_1$ et de la liste de franchissements l_1 et le marquage M_4 à partir de la liste des jetons $\mathcal{M}_2$ et de la liste de franchissements l_2.

Comme dans le cas précédent, une fois que nous avons obtenu un séquent caractéristique d'un ensemble de relations de précédence (c'est-à-dire d'un processus fini) pour $\mathcal{M}_1, l_1 \vdash M_3$ et pour $\mathcal{M}_2, l_2 \vdash M_4$, il faut réécrire les étiquettes du deuxième séquent pour qu'elles soient différentes du premier.

- Soit n le nombre d'atomes de $\mathcal{M}_1$, pour tous les atomes de $\mathcal{M}_2$, remplacer l'étiquette I^j par I^{j+n}. Cela permet de distinguer toutes les étiquettes initiales.
- Soit k le nombre de franchissements de t_i dans l_1, chaque franchissement t_i^j dans l_2 sera re-étiqueté t_i^{j+k}.

Ensuite on compose les deux pour obtenir un séquent caractéristique d'un ordre partiel déduit du séquent $\mathcal{M}_1, \mathcal{M}_2, l_1, l_2 \vdash M_3 \otimes M_4$ par la règle 3.12. On peut remarquer qu'il s'agit d'une juxtaposition de deux processus car il n'y aura aucun nœud en commun.

3.9.5 Un exemple de composition *série/parallèle*

Comme nous l'avons déjà mentionné auparavant, les démarches de caractérisation des ordres partiels fondées sur les ordres « série-parallèle » (ou des *pas* de franchissement) ne sont ni cohérentes ni complètes. En effet, si elles sont cohérentes et complètes vis-à-vis de l'ensemble des linéarisations des ordres partiels (elles préservent ainsi l'accessibilité), elles peuvent introduire des relations d'orde parasites. Le réseau de Petri de la figure 3.3 (tiré de [32]) donne un exemple de processus non factorisable avec les opérateurs série (« ; ») et parallèle (« $\|$ ») entre les marquages A et K. Nous allons le prendre pour exemple illustratif afin de montrer que les compositions série et parallèle que nous venons d'introduire ne possèdent pas cet inconvénient. Cela vient du fait que notre composition séquentielle de

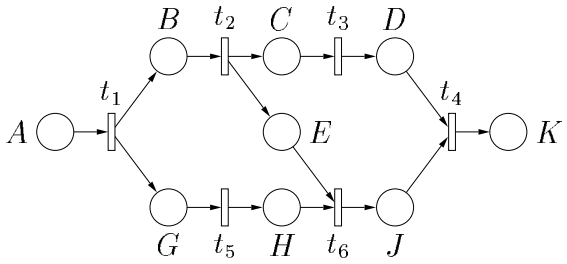

FIG. 3.3: Un réseau de Petri en Z inversé

$\mu_1, l_1 \vdash M_4$ avec $M_4, \mu_2, l_2 \vdash M_3$ n'entraîne pas le fait que dans l'ordre partiel généré tous les éléments de l_1 vont précéder tous les éléments de l_2. Si nous considérons une place p particulière de M_4, les seuls successeurs de p dans l_2 deviendront des successeurs des seuls prédécesseurs de p dans l_1.

Soit le séquent (dans la figure 3.3) :

$$a, t_1, t_2, t_3, t_4, t_5, t_6 \vdash k \tag{3.13}$$

Il génère un processus qui est moins contraint que, par exemple, la factorisation

$$t_1 \,;\, ((t_2 \,;\, t_3) \parallel t_5) \,;\, t_6 \,;\, t_4 \tag{3.14}$$

Nous allons maintenant remplacer les opérateurs série et parallèle par nos règles de composition. Considérons le séquent correspondant au franchissement de t_1 (comme chaque transition n'est franchie qu'une fois, nous omettons les indices correspondant au nombre de franchissements) :

$$a[I^1], \underbrace{a[I^1] \multimap b[t_1] \otimes g[t_1]}_{t_1} \vdash b[t_1] \otimes g[t_1] \tag{3.15}$$

Celui correspondant au franchissement en séquence de t_2 et t_3 ($t_2 \,;\, t_3$) est :

$$b[I^1], \underbrace{b[I^1] \multimap c[t_2] \otimes e[t_2]}_{t_2}, \underbrace{c[t_2] \multimap d[t_3]}_{t_3} \vdash d[t_3] \otimes e[t_2] \tag{3.16}$$

Celui correspondant au franchissement de t_5 est :

$$g[I^1], \underbrace{g[I^1] \multimap h[t_5]}_{t_5} \vdash h[t_5] \tag{3.17}$$

Enfin celui de la séquence $t_6 \,;\, t_4$ est :

$$d[I^1], e[I^2], h[I^3], \underbrace{e[I^2] \otimes h[I^3] \multimap j[t_6]}_{t_6}, \underbrace{d[I^1] \otimes j[t_6] \multimap k[t_4]}_{t_4} \vdash k[t_4] \tag{3.18}$$

Après réécriture, le séquent du franchissement de t_5 (3.17) devient :

$$g[I^2], \underbrace{g[I^2] \multimap h[t_5]}_{t_5} \vdash h[t_5] \tag{3.19}$$

La composition $(t_2\,;\,t_3) \parallel t_5$ s'exprime par :

$$\frac{b[I^1], \overbrace{b[I^1] \multimap c[t_2] \otimes e[t_2]}^{t_2}, \overbrace{c[t_2] \multimap d[t_3]}^{t_3} \vdash d[t_3] \otimes e[t_2] \qquad g[I^2], \overbrace{g[I^2] \multimap h[t_5]}^{t_5} \vdash h[t_5]}{b[I^1], g[I^2], \underbrace{b[I^1] \multimap c[t_2] \otimes e[t_2]}_{t_2}, \underbrace{c[t_2] \multimap d[t_3]}_{t_3}, \underbrace{g[I^2] \multimap h[t_5]}_{t_5} \vdash d[t_3] \otimes e[t_2] \otimes h[t_5]} \otimes_R \tag{3.20}$$

Après réécriture, le séquent du franchissement de $t_6\,;\,t_4$ (3.18) devient (nous avons déjà fait apparaître le monôme M_4) :

$$d[t_3] \otimes e[t_2] \otimes h[t_5] \underbrace{e[t_2] \otimes h[t_5] \multimap j[t_6]}_{t_6}, \underbrace{d[t_3] \otimes j[t_6] \multimap k[t_4]}_{t_4} \vdash k[t_4] \tag{3.21}$$

La composition en série nous donne (nous ne détaillons pas le contenu des transitions qui est bien entendu celui des séquents 3.20 et 3.21) :

$$\frac{b[I^1], g[I^2], t_2, t_3, t_5 \vdash d[t_3] \otimes e[t_2] \otimes h[t_5] \qquad d[t_3] \otimes e[t_2] \otimes h[t_5], t_6, t_4 \vdash k[t_4]}{b[I^1], g[I^2], t_2, t_3, t_5, t_6, t_4 \vdash k[t_4]}\ \text{cut} \tag{3.22}$$

Il ne reste plus qu'à mettre le franchissement de t_1 avant. Dans la réécriture du séquent conclusion du fragment d'arbre 3.22, il faut remplacer l'atome $b[I^1]$ par $b[t_1]$ et l'atome $g[I^2]$ par $g[t_1]$, ce qui modifie les expressions des franchissements des transitions t_2 et t_5 comme suit :

$$t_2 \quad : \quad b[t_1] \multimap c[t_2] \otimes e[t_2] \tag{3.23}$$

$$t_5 \quad : \quad g[t_1] \multimap h[t_5] \tag{3.24}$$

Nous obtenons :

$$\frac{a[I^1], t_1 \vdash b[t_1] \otimes g[t_1] \qquad b[t_1] \otimes g[t_1], t_2, t_3, t_4, t_5, t_6 \vdash k[t_4]}{a[I^1], t_1, t_2, t_3, t_4, t_5, t_6 \vdash k[t_4]}\ \text{coupure} \tag{3.25}$$

L'examen de la forme des transitions t_1 dans 3.15, t_2 dans 3.23, t_3 dans 3.20, t_4 dans 3.21, t_5 dans 3.24 et t_6 dans 3.21, nous montre que les relations de précédence sont correctement définies. Il n'ya pas de raltion de précédence parasite comme, par exemple, dans l'expression $t_1\,;\,((t_2\,;\,t_3) \parallel t_5)\,;\,t_6\,;\,t_4$. En effet, cette expression inclut une relation de précédence entre le franchissement de t_3 et celui de t_6 alors qu'il n'en est rien lorsque nous regardons le réseau. Ces deux transitions sont structurellement indépendantes et peuvent être franchies en parallèle.

3.10 Conclusion

Dans ce chapitre, nous avons montré que les relations de précédence entre les applications des règles pour la construction d'un arbre canonique étaient équivalentes aux processus finis de réseaux de Petri entre les mêmes marquages et pour les mêmes franchissements de transition. Cette équivalence justifie notre approche : l'élimination d'une formule implicative dans un séquent d'accessibilité peut bien être associé à l'événement correspondant au franchissement de cette transition.

Nous avions toutefois un problème : à un arbre de preuve canonique peut correspondre plusieurs processus lorsqu'il y a des conflits jetons.

À partir de l'annotation des jetons d'un arbre de preuve, nous avons vu que nous pouvions réécrire le séquent initial en un séquent caractéristique qui caractérisait un seul ordre partiel entre les franchissements de transitions.

Le séquent caractéristique pour un scénario donné décrit :

- l'état partiel initial à partir duquel le scénario peut s'exécuter,
- l'état partiel final résultant de l'exécution,
- l'ensemble des événements composant le scénario,
- le nombre de fois que chaque événement a eu lieu,
- les relations de précédence qui relient ces événements.

Le séquent caractéristisque est l'élément de base pour faire de l'analyse temporelle car il permet de construire l'ordre partiel entre les événements. Un séquent caractéristique est équivalent à un processus fini de réseau de Petri entre deux marquages. Il présente l'avantage d'avoir une notation logique, sous la forme d'un séquent, au lieu d'une représentation graphique. Nous avons vu à la fin de ce chapitre que cette notation logique permettait l'élaboration de règles de composition simples sous la forme de déductions.

Chapitre 4

Analyse temporelle qualitative et quantitative

4.1 Introduction

L'analyse des systèmes distribués implique de vérifier le respect des contraintes logiques de synchronisation mais aussi celui des échéances temporelles quantitatives. Pour pouvoir expliciter les contraintes temporelles quantitatives entre les événements possibles, nous devons d'abord générer les scénarios de franchissements de transitions.

Nous avons vu précédemment qu'à un arbre de preuve canonique de logique linéaire pouvaient être associés plusieurs processus. Cela signifie que nous pouvons associer à un arbre de preuve plusieurs ensembles de relations de précédence entre les applications des règles. Un ensemble de relations de précédence est équivalent à un graphe de précédence. S'il existe plusieurs graphes de précédence alors plusieurs ordonnancements sont possibles. Il a été montré d'autre part qu'un séquent caractéristique représentait un seul ordre partiel. Il correspond à un scénario de franchissements de transitions contenant un marquage initial et un marquage final donnés, une liste d'événements et un ensemble de relations de précédence. Nous n'avons ainsi qu'un seul ordonnancement possible pour un séquent caractéristique donné.

Dans le cadre des systèmes à temps contraint, ordonnancer de façon optimale un ensemble de tâches c'est trouver le meilleur ordre partiel parmi les tirs de transition correspondant aux tâches à exécuter mais aussi leur date optimale de tir. Le modèle réseau de Petri doit être complété afin de prendre en compte les contraintes temporelles (durées des tâches, dates de fin d'exécution etc). Ces contraintes temporelles sont associées aux places, aux transitions ou aux arcs d'un réseau de Petri (suivant le modèle utilisé) sous la forme d'intervalles temporels ou de durées précises.

Les relations de précédence que nous obtenons à partir d'un arbre de preuve en logique linéaire correspondent à des relations de causalité entre les événements. Ces relations de

causalité sont en fait des relations de cause à effet immédiates. Si « e_i précède e_j » cela veut dire que l'un des jetons produits par e_i est consommé par e_j. Nous allons supposer dans ce chapitre que les contraintes temporelles quantitatives sont uniquement liées aux relations de cause à effet immédiates exprimées par le réseau de Petri. Le problème de l'ordonnancement des tâches se ramène donc à celui de l'ordonnancement des dates de franchissement des transitions soumises à un ensemble de contraintes temporelles. La meilleure façon de représenter ces contraintes est d' utiliser un graphe de contraintes temporelles.

Pour raisonner sur un graphe de contraintes temporelles, nous allons donc réécrire les contraintes temporelles sous la forme d'un TCSP [8] (temporal constraint satisfaction problem), représenté par un système d'inéquations linéaires, que nous résoudrons par des méthodes classiques utilisées dans les approches par contraintes.

Nous illustrerons le principe de cette méthode par un exemple portant sur la présentation de documents multimedia.

4.2 Du temps qualitatif au temps quantitatif

4.2.1 Contraintes sur les relations de précédence définies par réseau de Petri

Une relation de précédence exprime une causalité entre deux événements successifs qui doit être obligatoirement vérifiée pour qu'un système modélisé ait le comportement souhaité. Pour trouver ces relations, nous avons réécrit les problèmes d'accessibilité dans un réseau de Petri sous la forme de preuves de séquents de logique linéaire. La preuve de ces séquents se fait par la construction d'un arbre de preuve canonique. Les relations de précédence sont obtenues par un processus d'annotation de l'arbre de preuve qui lie les ressources produites et consommées aux événements associés aux tirs de transitions.

À partir de cette annotation, nous avons réécrit le séquent initial prouvé en ajoutant aux atomes leur instant de production (correspondant au tir de transition) à la fois dans le marquage initial, dans le final et dans la liste des tirs de transitions (voir chapitre précédent). Cette réécriture du séquent fige les relations de précédence. Nous obtenons ainsi un séquent caractéristique d'un seul ordre partiel. De plus, la liste des tirs de transition du séquent caractéristique initial définit un ordonnancement strict des franchissements de transition.

Les relations de précédence entre les événements doivent être vérifiées par tout comportement correct. Elles sont donc assimilées à des contraintes *qualitatives* car elles ne contiennent aucune information quantitative. Chacune d'entre elles exprime une précédence entre deux événements successifs ce qui signifie que la contrainte est *binaire* [33] : soit la relation de précédence entre les deux événements est vraie soit elle est fausse.

Pour analyser ces contraintes binaires dans les systèmes à événements discrets, il existe

des outils qui ont été développés dans un cadre formel : les problèmes de satisfaction de contraintes (CSP, voir [39]). où les variables sont booléennes (domaines de valeurs égaux à $\{0,1\}$).

En résumé, un séquent caractéristique définit un ensemble de relations de précédence qui peuvent être représentées sous une forme graphique appelée *graphe de contraintes*. Les nœuds de ce graphe sont les dates des franchissements. Ce sont les variables du problème de satisfaction de contraintes.

4.2.2 Modélisation des contraintes temporelles

La prise en compte simultanée des contraintes de précédence et de durées entre les événements (franchissements de transition) se fait tout naturellement en se plaçant dans le cadre d'un type de CSP particulier : les problèmes de satisfaction de contraintes temporelles (TCSP). Dans les TCSP, les contraintes entre les variables sont binaires et les paramètres utilisés dans les contraintes prennent des valeurs sur des domaines qui peuvent être des durées ou des intervalles. Ces valeurs se retrouvent associées aux arcs des graphes.

Dans la définition originelle des TCSP, les variables peuvent avoir pour domaine de valeurs, une union d'intervalles. Cependant, il existe une classe particulière plus simple de ces TCSP qui ne considère qu'un seul intervalle par domaine de valeurs possibles pour les variables : les problèmes temporels simples (STP).

Mathématiquement, les TCSP et les STP sont définis de la manière suivante.

Définition 17 (TCSP) *Un problème à satisfaction de contraintes temporelles est un triplet* (X, D, C). $X = \{x_1, ..., x_n\}$ *est un ensemble de n variables (dates) ayant des domaines continus de valeurs.* $D = \{d_1, ..., d_n\}$ *est l'ensemble des domaines (intervalles temporels) des variables.* $C = \{c_1, ..., c_m\}$ *est un ensemble de m contraintes. Chaque contrainte est représentée par un ensemble d'intervalles* $C \stackrel{def}{=} \{i_1, ..., i_n\} = \{[i_{1min}, i_{1max}], ..., [i_{nmin}, i_{nmax}]\}$. *Une contrainte binaire* c_{ij} *entre* x_i *et* x_j *restreint les valeurs possibles pour la distance* $x_j - x_i$ *et est représentée par* $c_{ij} \stackrel{def}{=} (i_{1min} \leq x_j - x_i \leq i_{1max}) \cup ... \cup (i_{nmin} \leq x_j - x_i \leq i_{nmax})$.

Un n-uplet $X = (x_1, ..., x_n)$ est appelé *solution* si $\{X_1 = x_1, ..., X_n = x_n\}$ satisfait toutes les contraintes. Notre problème est *cohérent* si et seulement s'il existe au moins une solution.

Les problèmes temporels simples communément appelés par leur abréviation anglaise STP(simple temporal problems) sont définis de la façon suivante.

Définition 18 (Problème temporel simple) *Un problème temporel simple est un TCSP où les domaines des variables de* X *et les contraintes* c_{ij} *de* C *ne comportent qu'un seul intervalle temporel* $[a_{ij}, b_{ij}]$ *tel que* $a_{ij} \leq x_j - x_i \leq b_{ij}$.

Un réseau de contraintes binaires temporelles associé à un TCSP ou à un STP est constitué de l'ensemble des variables X et d'un ensemble de constraintes. Il peut être représenté sous la forme d'un graphe tel que les nœuds représentent les variables et qu'un

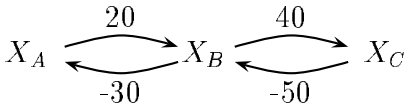

FIG. 4.1: Exemple de STP

arc $i \to j$, étiqueté par l'ensemble des intervalles, indique qu'une contrainte c_{ij} est spécifiée (elle existe). Les TCSP et les STP sont souvent utilisés pour l'analyse de contraintes temporelles liées à des activités. Dans la communauté scientifique de l'ordonnancement exploitant les TCSP, il a été pris pour habitude d'utiliser des graphes où les activités sont liées aux arcs (*activity-on-arc graph* en anglais, graphe AOA pour l'abréviation).

Définition 19 (Graphe AOA) *Un graphe AOA [12] est un graphe tel que les nœuds représentent des événements de début ou de fin d'un ensemble d'activités et tel que la longueur d'un arc représente la durée de l'activité associée (c'est une contrainte temporelle entre deux événements).*

Un graphe AOA peut être vu comme une représentation graphique d'un STP. La borne minimale a_{ij} de la contrainte c_{ij} est associée à l' activité reliant l'événement x_i à l'événement x_j. Si la borne maximale b_{ij} est différente de l'infini, une activité virtuelle est rajoutée entre x_j et x_i de longueur $-b_{ij}$.

Exemple : Considérons que nous voulons nous déplacer d'un point A à un point C en passant par un point B. Sachant que la durée du déplacement entre les points A et B varie entre 20 et 30 minutes et que la durée entre B et C varie entre 40 et 50 minutes, voici une représentation graphique du problème (figure 4.1). X_A représente l'instant de départ de A, X_B représente l'instant d'arrivée depuis A mais aussi l'instant de départ de B et X_C représente l'instant d'arrivée depuis B.

Ainsi les deux contraintes

$$\begin{array}{lrcl} C_{A\_B} & 20 \leq & x_B - x_A & \leq 30 \\ C_{B\_C} & 40 \leq & x_C - x_B & \leq 50 \end{array}$$

sont représentées sous la forme de quatre activités

$$\begin{array}{lrl} C_{A\_B} & 20 & \leq x_B - x_A \\ C_{B\_A} & -30 & \leq x_A - x_B \\ C_{B\_C} & 40 & \leq x_C - x_B \\ C_{C\_B} & -50 & \leq x_B - x_C \end{array}$$

4.2.3 Extensions temporelles de réseaux de Petri

Pour prendre en compte les contraintes quantitatives, nous avons vu qu'il existait des extensions temporelles au modèle réseau de Petri : le temps est associé aux transitions,

ou aux places ou aux arcs. Le temps est associé de deux manières : soit c'est une durée associée à un type de nœud et nous parlons de réseau « temporisé », soit c'est un intervalle temporel et nous parlons alors de réseau « temporel ».

Nous allons présenter les différentes extensions temporelles [1] les plus répandues avec leur graphe d'activité associé.

Les réseaux de Petri p-temporisés [57]

Définition 20 (Réseau de Petri p-temporisé) *Un réseau de Petri p-temporisé est une paire* $< N, \Theta_{fp} >$ *où* N *est un réseau de Petri marqué et* Θ_{fp} *une fonction* $\Theta_{fp} : P \rightarrow \mathbb{Q}^+$ *associant une durée* d_i *à chaque place* p_i *du réseau.*

Chaque jeton doit rester au moins d_i unités de temps dans un état non disponible quand il arrive dans la place p_i avant d'être consommé par un franchissement de transition.

Nous allons voir comment les contraintes temporelles quantitatives introduites par cette approche peuvent être prises en compte dans le cadre de l'analyse des relations de causalité que nous avons présentée au chapitre précédent.

Le séquent caractéristique déterminé à partir d'un étiquetage d'un arbre de preuve caractérise un seul ordre partiel entre les événements associés aux franchissements des transitions. Cet ordre partiel peut être représenté par un graphe de précédence où les nœuds sont les franchissements de transition et les arcs sont les correpondent aux relations de précédence entre deux franchissements.

Si nous considérons que les places d'un réseau de Petri p-temporisé sont des activités, de façon évidente le graphe de précédence peut être transformé en un graphe AOA. Les nœuds annotés par les franchissements de transition t_i^j sont remplacés par des variables x_i associées aux dates des franchissements et les arcs annotés par les noms des places p_i sont remplacés par les durées d_i associées aux activités correspondantes. Nous pouvons ainsi traduire l'ensemble des contraintes générées par le réseau de Petri p-temporisé en un ensemble de graphes AOA, chaque graphe étant associé à un séquent caractéristique.

La structure de ces graphes est typiquement celle d'un ordre partiel. Ils sont acycliques car nous allons d'un marquage donné à un autre. Il est donc toujours possible de trouver une solution à un problème de satisfaction de contraintes temporelles basé sur un réseau de Petri p-temporisé. En effet, un résultat fondamental sur les graphes AOA [33] nous indique qu'il est cohérent si et seulement si il ne comprend aucun cycle de longueur positive. Cependant, ce modèle ne permet de travailler que sur contraintes « au plus tôt » sans possibilité d'introduire des contraintes « au plus tard ». Il n'y a aucune notion d'urgence.

L'exemple de la figure 4.3 montre le graphe AOA associé au réseau de Petri de la figure 4.2 pour l'ordre partiel où le franchissement de transition t_1^1 a lieu avant le fran-

[1] Nous ne ferons pas un comparatif des différentes extensions temporelles. Nous préférons pour cela renvoyer le lecteur à la thèse de M. Boyer [6]

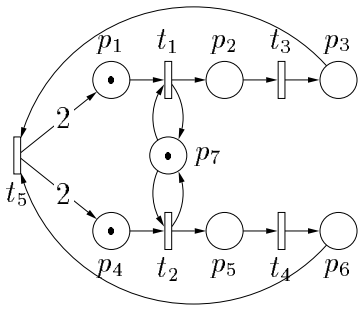

FIG. 4.2: Exemple de réseau de Petri

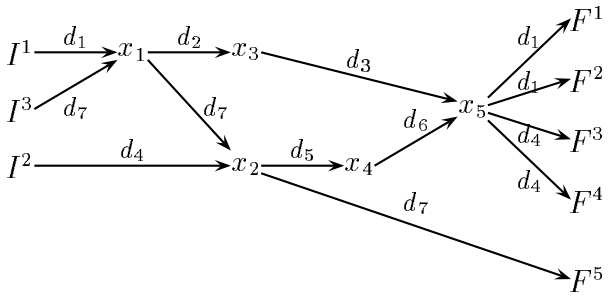

FIG. 4.3: Graphe AOA associé au réseau de Petri p-temporisé

chissement t_2^1. Nous associons respectivement aux places p_i les durées d_i pour obtenir un réseau de Petri p-temporisé. Les varaibles x_i correspondent aux dates des franchissements t_i^1. Ce graphe est structurellement identique au graphe des relations de précédence de la figure 3.1. Les variables I_k correspondent aux dates de début des activités initiales et les variables F_k à celles de fin des activités finales.

Les réseaux de Petri p-temporels [30, 31, 29]

Définition 21 (Réseau de Petri p-temporel) *Un réseau de Petri p-temporel est une paire* $< N, IS_p >$ *où* N *est un réseau de Petri marqué et* IS_p *une fonction d'intervalle statique* $IS_p : P \rightarrow \mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \infty)$ *associant une durée minimale* d_{imin} *et une durée maximale* d_{imax} *à chaque place* p_i *du réseau.*

Les réseaux de Petri p-temporels ont été créés à partir des réseaux de Petri t-temporels. Chaque jeton doit rester au moins d_{imin} unités de temps dans la place p_i et doit être consommé par un franchissement de transition avant d_{imax} au plus tard.

La durée pendant laquelle le jeton restera dans la place peut prendre n'importe quelle valeur du domaine temporel $[d_{imin}, d_{imax}]$. Avant d_{imin}, le jeton est indisponible. Une transition de sortie de la palce p_i peut être franchie en consommant ce jeton à partir du moment où la durée écoulée est supérieure ou égale à d_{imin}. Elle ne sera plus franchissable en consommant ce jeton lorsque la durée écoulée est supérieure à d_{imax} : le jeton est dit *mort*. La notion de *mort* d'un jeton est liée à la notion de violation de contrainte

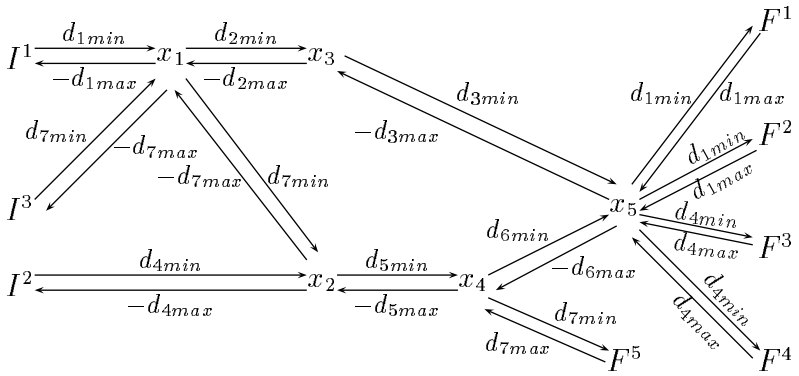

FIG. 4.4: Graphe AOA associé au réseau de Petri p-temporel

temporelle. Il y a en fait une horloge par jeton du marquage courant.

Comme dans le cas précédent, voyons comment ces contraintes quantitatives vont se traduire sur le graphe des relations de précédence associé à un séquent caractéristique. Comme précédemment les places peuvent être vues comme des activités. En fait, c'est un jeton dans une place qui correspond à l'instanciation d'une activité d'un certain type. Mais, comme dans cette approche il y a simultanément une contrainte « au plus tôt » et une contrainte « au plus tard » associées à chaque activité, ce sont deux contraintes qui sont associées à chaque relation de précédence. Une contrainte représente la durée minimale et l'autre la durée maximale. La figure 4.4 montre le graphe AOA associé au réseau de Petri de la figure 4.2 où nous associons à chaque place p_i un intervalle $[d_{imin}, d_{imax}]$ pour obtenir un réseau de Petri p-temporel. Ce graphe est valable pour l'ordre partiel où le franchissement de transition t_1^1 a lieu avant le franchissement t_2^1.

Chaque arc du graphe de précédence étant remplacé par deux arcs (un dans chaque sens), le graphe AOA obtenu n'est plus acyclique. Il est donc tout à fait possible qu'un ensemble de contraintes temporelles exprimées sous la forme p-temporelle soit contradictoire avec les contraintes logiques des relations de précédence exprimées sous la forme d'un séquent caractéristique.

Par exemple, le graphe AOA de la figure 4.4 comporte deux cycles qui donnent les inéquations suivantes (la longueur du cycle ne peut être positive) :

$$d_{7min} + d_{5min} + d_{6min} - d_{3max} - d_{2max} \leq 0 \quad (4.1)$$

$$d_{2min} + d_{3min} - d_{6max} - d_{5max} - d_{7max} \leq 0 \quad (4.2)$$

Nous ne considérons que ces deux cycles car nous supposons que pour tout $i, d_{imax} \geq d_{imin}$.

La différence fondamentale de cette approche avec, par exemple, celle du graphe des classes est que les contraintes temporelles sont exprimées sous la forme de paramètres (les d_{imin} et les d_{imax}) et non de valeurs. Au lieu de vérifier pour un jeu de valeurs donné qu'une séquence de franchissements de transitions est possible ou non, nous obtenons un

ensemble de contraintes sur les paramètres (les équations (4.1) et (4.2) ci-dessus) pour qu'un certain comportement soit possible. Nous reviendrons sur ce point à la fin de ce chapitre.

Les réseaux de Petri p-temporels sont capables de prendre en compte les échéances des activités. De plus, ils sont plus généraux que les réseaux de Petri p-temporisés car ils permettent de décrire certains mécanismes (chiens de garde par exemple) non modélisables avec les réseaux p-temporisés. De plus il existe déjà des méthodes de programmation linéaire qui permettent d'effectuer une recherche de solution (voir thèse de P. Bonhomme [5]).

La différence entre l'approche développée dans la thèse de P. Bonhomme et la notre est que nous ne nous appuyons pas sur une séquence de franchissements obtenue par la construction initiale d'un graphe des classes pour obtenir un système d'inéquations. Nous l'obtenons directement à partir d'un graphe de précédence exprimant les relations de causalité strictement nécessaires, sans risque d'introduire des relations de précédence parasites, conséquences d'une linéarisation particulière de l'ordre partiel induite par le graphe des classes (sémantique d'entrelacement).

Les réseaux de Petri t-temporisés [47]

Définition 22 (Réseau de Petri t-temporisé) *Un réseau de Petri t-temporisé est une paire* $< N, \Theta_{ft} >$ *où* N *est un réseau de Petri marqué et* Θ_{ft} *une fonction* $\Theta_{ft} : T \rightarrow \mathbb{Q}^+$ *associant une durée* d_i *à chaque transition* t_i *du réseau.*

Quand une transition t_i est franchie, un temps d_i s'écoule entre la consommation des jetons en entrée de la transition ($Pre(t_i)$) et la production des jetons en sortie ($Post(t_i)$).

Avant de renommer les nœuds et les arcs, il est impératif de transformer le graphe de précédence car les transitions ne sont plus assimilées à des événements. En effet, le fait d'associer le temps aux transitions lie implicitement ces dernières aux activités qui ne sont donc plus liées aux arcs du graphe des relations de précédence. Chaque nœud du graphe est ainsi transformé en un arc associé à la durée de franchissement de la transition correspondante. Désormais, les nœuds de notre graphe sont des arcs dont les origines représentent les dates de débuts de franchissements de transition et les fins représentent les dates de fins de franchissements de transition alors que précédemment les nœuds étaient directement interprétés comme étant des dates de franchissement.

Cette modification est représentée par le graphe de la figure 4.5 avec pour exemple le même ordre partiel que précédemment où le franchissement de transition t_1^1 a lieu avant le franchissement de transition t_2^1.

Chaque place p_i est ensuite remplacée par la valeur 0 étant donné que le temps n'est plus associé aux places et qu'elles ne représentent plus qu'une relation de précédence. Et chaque franchissement de transition t_i^j est remplacé par deux nœuds x_i^j et y_i^j associés à la transition t_i dans le réseau de Petri. Par contre, nous ne remplaçons pas les instants initiaux (I^1, I^2, I^3) et finaux (F^1, F^2, F^3, F^4, F^5), car ils restent associés aux événements

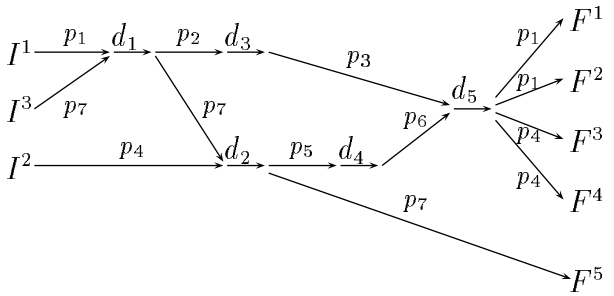

FIG. 4.5: Graphe associé au réseau de Petri t-temporisé

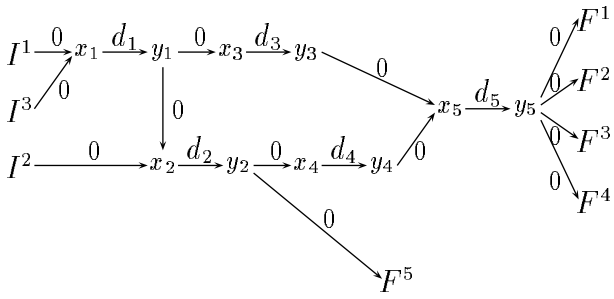

FIG. 4.6: Graphe AOA associé au réseau de Petri t-temporisé

de production initiale et de consommation finale des jetons. Nous obtenons alors le graphe AOA de la figure 4.6. Le graphe est toujours acyclique et par conséquent il existe toujours une solution à l'ensemble de contraintes.

Les réseaux de Petri t-temporels [37]

Définition 23 (Réseau de Petri t-temporel) *Un réseau de Petri t-temporel est une paire* $< N, IS_t >$ *où* N *est un réseau de Petri marqué et* IS_t *une fonction d'intervalle statique* $IS_t : T \rightarrow \mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \infty)$ *associant une durée minimale* d_{imin} *et une durée maximale* d_{imax} *à chaque transition* t_i *du réseau.*

La transition t_i doit rester sensibilisée durant au moins d_{imin} unités de temps et au plus d_{imax} unités de temps avant d'être franchie.

Nous nous plaçons dans la sémantique opérationnelle multiserveur, c'est-à-dire que nous avons une horloge associée à chaque n-uplet de jetons sensibilisant une transition. Il n'y a pas d'effet mémoire c'est-à-dire que l'horloge est remise à zéro dès qu'un jeton du n-uplet est consommé par un franchissement, qu'il s'agisse ou non du franchissement de la transition concernée, et que le jeton retourne ou non dans sa place après le franchissement. Enfin, nous nous plaçons également dans le cadre de la sémantique « faible », c'est-à-dire que lorsque plusieurs transitions sont en conflit, c'est une décision *externe* qui choisit la transition à franchir indépendamment des contraintes temporelles. Les contraintes tem-

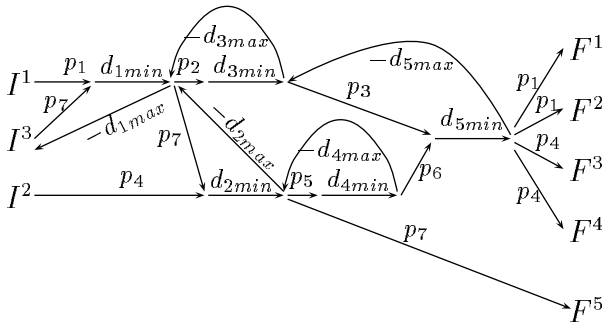

FIG. 4.7: Graphe AOA associé au réseau de Petri t-temporel

porelles associées au franchissement d'une transition dans le cadre d'un scénario ne sont jamais modifiées par les conditions de franchissement d'une autre transition, même si cette dernière est en conflit avec la transition considérée. En effet, dans le cas contraire il y aurait une contradiction avec la linéarité de la logique linéaire qui suppose que si un comportement est possible, il le reste lorsque des jetons sont rajoutés.

Comme pour les réseaux de Petri t-temporisés, les activités sont liées aux transitions. Les nœuds du graphe de précédence (franchissements de transition) sont donc transformés en arcs correspondant aux durées de franchissement des transitions, ce qui au premier abord nous donne un graphe identique à celui de la figure 4.5.

Étant donné que c'est un intervalle temporel qui est associé aux transitions, nous avons besoin de prendre en compte les deux contraintes dans le graphe. Néanmoins, cette fois la première date est celle de la sensibilisation de la transition et la seconde celle du franchissement, supposé instantané. Soit x_i la première et y_i la seconde. La contrainte minimale correspondant à une durée de sensibilisation s'exprime directement en mettant un arc de longueur d_{imin} entre x_i et y_i.

Par contre la contrainte d'attente maximale pose un problème car la variable x_i correspond en fait à la borne minimale de la date de sensibilisation. Si nous considérons par exemple le franchissement de t_5, l'instant de sensibilisation coïncide avec l'arrivée du dernier jeton dans l'une de ses deux places d'entrée p_3 ou p_6. Supposons qu'il s'agit de p_3. La date de sensibilisation de t_5 coïncide alors avec la date du franchissement de t_3 (y_3, nœud final de l'arc d_{3min}). Donc l'arc de longueur $-d_{5max}$ relie alors le nœud final de l'arc d_{5min} au nœud final de d_{3min}.

Le graphe AOA correspondant au scénario étudié en supposant que t_1 est sensibilisée lors de l'arrivée du jeton dans p_7, t_2 est sensibilisée lors de la deuxième arrivée d'un jeton en p_7 et t_5 lors de l'arrivée d'un jeton en p_3 est représenté figure 4.7 ; pour augmenter la visibilité, les variables x_i et y_i ont été omises et les noms des places associés aux arcs maintenus au lieu de l'information de leur longueur nulle. Mais pour un autre séquencement nous obtiendrons un autre graphe. Comme les graphes AOA ne nous permettent pas de représenter un ensemble de contraintes disjonctives (reliées par un « ou »), nous n'avons

pas donc pas de représentation simple des contraintes quantitatives associées à un réseau de Petri t-temporel. Une expression simple n'est possible que si nous nous ramenons à une séquence impliquant un ordre total entre les franchissements des transitions.

Toutefois nous pouvons énoncer la proposition suivante.

***Proposition* 1** *Si le marquage* $[M_0 + (Pre - Post).\overline{\sigma}]$ *est accessible dans R, alors l'ensemble des contraintes associées à chaque réseau de Petri t-temporel construit à partir de R est toujours cohérent pour le marquage initial* M_0 *et la liste de franchissements de transition* σ*.*

Cette proposition est déduite du fait qu'il n'existe pas de circuit positif dans les graphes AOA correspondants. En effet, quand la traduction d'un réseau de Petri t-temporel en un graphe AOA est faite, les circuits générés par les transitions qui ont une unique place d'entrée ne peuvent pas être positifs sinon cela veut dire que les bornes données sont incohérentes.

Exemple L'aspect le plus intéressant apparaît lorsque des circuits sont générés à partir de réseaux de Petri contenant deux branches parallèles. Considérons pour cela notre exemple de la figure 4.2 dans le cadre d'une extension t-temporelle. Un graphe AOA correspondant est celui de la figure 4.7.

Dans ce graphe AOA, il existe le circuit $[d_{2min}, d_{4min}, d_{5min}, -d_{5max}, -d_{3max}]$. L'arc attaché à la longueur négative $-d_{5max}$ relie le nœud correspondant à la fin du franchissement de la transition t_5 au nœud correspondant à la fin du franchissement de t_3. Ceci n'est valable que dans le cas où la transition t_3 a produit un jeton après la transition t_4. Dans le cas où la transition t_4 aurait produit un jeton après la transition t_3, nous aurions eu l'arc $-d_{5max}$ reliant la fin du franchissement de t_5 à la fin du franchissement de t_4.

Soit θ_3 la date de production du jeton dans la place p_3 et θ_6 la date de production du jeton dans la place p_6. Étant donné que p_6 est produit avant p_3, nous avons la relation : $\theta_6 \leq \theta_3$.

Soit θ_1 la date de franchissement de la transition t_1. Nous savons que $\theta_3 \in [\theta_1 + d_{3min}; \theta_1 + d_{3min}]$ et que $\theta_6 \in [\theta_1 + d_{2min} + d_{4min}; \theta_1 + d_{2max} + d_{4max}]$. Nous pouvons en déduire que la condition nécessaire est : $d_{3max} \geq d_{2min} + d_{4min}$. Nous avons vu précédemment qu'étant donné que le temps était associé aux transitions, les arcs associés aux places dans le graphe AOA représentaient une durée nulle. La longueur du circuit qui nous intéresse dans le graphe AOA est donc : $0 + d_{2min} + 0 + d_{4min} + 0 + d_{5min} - d_{5max} - d_{3max}$. Sachant que $d_{5min} \leq d_{5max}$ et que $d_{2min} + d_{4min} \leq d_{3max}$ alors la longueur de ce circuit est inférieure ou égale à zéro.

Après avoir montré qu'il était impossible de trouver des contraintes quantitatives contradictoires avec les contraintes de précédence si t_4 est franchie avant t_3, il est possible de faire un raisonnement analogue dans le cas contraire.

Ce résultat provient du fait que pour un réseau t-temporel en sémantique faible, il est toujours possible de franchir les transitions dans l'intervalle spécifié puisque les horloges

ne sont activées que lorsque les transitions sont sensibilisées, et donc franchissables.

4.2.4 Raisonnement sur les variables : contrôlabilité et non contrôlabilité

Les graphes AOA permettent de représenter des contraintes de précédence et de durées entre événements. Un grand nombre de résultats ont été obtenus dans cette communauté et plus généralement dans celle des TCSP [54, 55]. Les travaux les plus récents distinguent, de façon réaliste deux types de contraintes : les contraintes *contingentes* et les contraintes *contrôlables* [59, 27].

Définition 24 (Contrainte contingente) *Une contrainte est contingente si la durée effective entre les instants de début et de fin d'activité liée à la contrainte est incertaine et ne sera connue qu'en cours d'exécution.*

Dans le domaine de l'ordonnancement et de la plannification, le terme de contrainte *non contrôlable* est aussi utilisé comme synonyme de contrainte contingente.

Définition 25 (Contrainte contrôlable) *Une contrainte est contrôlable si la durée effective entre les instants de début et de fin d'activité liée à la contrainte peut être restreinte ou instanciée.*

La « contrôlabilité » des contraintes temporelles dépend fortement du type d'application à ordonnancer. En effet, suivant la criticité de l'application, le concepteur est amené à prendre plus ou moins en compte les contraintes liées aux exécutions situées dans le futur afin d'agir en temps réel. Pour une application qui requiert un respect total des échéances temporelles, il sera impératif de prendre en compte toutes les exécutions possibles futures. Par contre, pour une application qui peut se permettre de ne pas toujours respecter les échéances temporelles, prendre en compte toutes les possibilités d'exécutions futures n'est pas forcément toujours le bon choix à faire. Le tout est de savoir ce que le concepteur souhaite privilégier : l'exécution de l'application en temps borné ou l'exécution complète de l'application. Il est évident que vouloir exécuter une application en temps borné et connu à l'avance amène assez souvent le concepteur à faire des sacrifices sur le temps imparti à l'exécution des tâches qui composent l'application.

Thierry Vidal a caractérisé dans un article ([59]) plusieurs degrés de contrôlabilité en fonction de la prise en compte ou non d'observations situées dans le futur pour certaines décisions. Nous allons illustrer par un exemple, composé de deux branches entre une transition *fork* et une transition *join*, que dans notre cas ce choix a une influence sur le modèle temporel choisi pour le réseau de Petri.

Exemple Considérons le réseau de Petri p-temporel de la figure 4.8 et son graphe AOA correspondant (figure 4.9). Dans un réseau de Petri p-temporel, tout jeton doit quitter la place à laquelle il appartient une fois la borne maximale de l'intervalle temporel associé

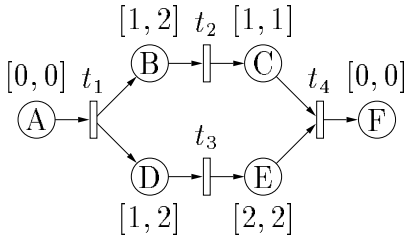

FIG. 4.8: Réseau de Petri p-temporel

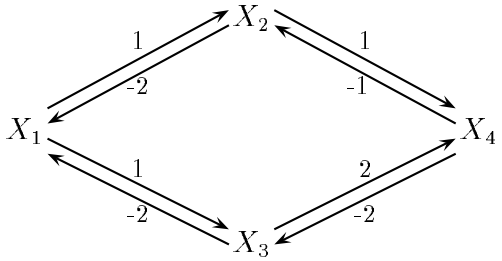

FIG. 4.9: Graphe AOA associé au réseau de Petri p-temporel

atteinte, sinon le jeton est dit *mort* (inutilisable). Ce modèle est très intéressant pour vérifier si la synchronisation de branches parallèles s'effectue bien en temps voulu. Dans l'exemple, nous voulons vérifier si les spécifications temporelles, données par le concepteur, des tâches représentées par les places B, C, D et E sont cohérentes.

Pour cela, nous construisons un système d'inéquations linéaires à l'aide des variables X_1, X_2, X_3 et X_4 qui représentent, respectivement, les dates de franchissement des transitions t_1^1, t_2^1, t_3^1 et t_4^1 avec comme paramètre $X_1 = 0$. Voici toutes les inéquations obtenues à partir du graphe AOA (elles représentent toutes les contraintes entre deux variables qu'il est possible de déduire) :

$$\begin{array}{l|l|l|l} X_2 - X_1 \geq 1 & X_3 - X_1 \geq 1 & X_4 - X_1 \geq 2 & X_3 - X_2 \geq -1 \\ X_1 - X_2 \geq -2 & X_1 - X_3 \geq -2 & X_1 - X_4 \geq -3 & X_2 - X_3 \geq -1 \\ X_4 - X_2 \geq 1 & X_4 - X_3 \geq 2 & X_4 - X_1 \geq 3 & X_3 - X_2 \geq -1 \\ X_2 - X_4 \geq -1 & X_3 - X_4 \geq -2 & X_1 - X_4 \geq -4 & X_2 - X_3 \geq 1 \end{array}$$

Considérons les inéquations des deux colonnes de droite en ne gardant que les inéquations où les contraintes sont les plus fortes :

$$(a) X_3 - X_2 \geq -1 \quad (b) X_2 - X_3 \geq 1 \quad (c) X_4 - X_1 \geq 3 \quad (d) X_1 - X_4 \geq -3$$

Raisonnons avec les valeurs des bornes des intervalles des places B et D. Comme $X_1 = 0$, nous avons : $X_{3min} = 1$, $X_{3max} = 2$, $X_{2min} = 1$ et $X_{2max} = 2$. Pour les quatre combinaisons

des valeurs du couple (X_2, X_3), une seule solution vérifie les inéquations (a) et (b) ci-dessus : $X_2 = 2$ et $X_3 = 1$. De plus, d'après les inéquations (c) et (d), nous avons $X_4 \geq 3$ et $X_4 \leq 3$. Il est donc impossible que la transition t_2 puisse être franchie à la date $X_2 = 1$ car ça nous donnerait $X_4 = 2$, ce qui est incohérent. De même, il est impossible que la transition t_3 puisse être franchie à la date $X_3 = 2$ car nous aurions $X_4 = 4$, ce qui est aussi incohérent. Il nous faut modifier les valeurs des intervalles temporels des places B et D si nous voulons avoir un modèle cohérent. Les nouveaux intervalles temporels sont $[2, 2]$ pour la place B et $[1, 1]$ pour la place D.

Dans cet exemple, nous avons pu montrer que les intervalles temporels des places C et E avaient eu une influence sur les intervalles temporels des places B et D par le biais du STP et de son système d'inéquations linéaires. Certaines contraintes apparaissent ainsi plus contrôlables que d'autres. Ici, X_1 et X_4 ont été prises comme contraintes contingentes et X_2 et X_3 comme contraintes contrôlables. Nous aurions pu modifier ces valeurs en procédant à un calcul de dates basé sur les combinaisons de toutes les valeurs possibles. Cela dit, nous aurions peut être modifié d'autres intervalles que les intervalles des places B et D. Plus les modifications des intervalles interviennent tard dans le réseau (proche de la fin), plus les répercussions sur les intervalles antérieurs peuvent être importantes. L'utilisation des méthodes de résolution des réseaux de contraintes (propagation de contraintes) nous permet de trouver une solution cohérente sans avoir à modifier beaucoup de valeurs. L'intérêt est quand même d'apporter un minimum de modifications sinon cela devient trop contraignant pour le système.

4.3 Exemple : une application multimedia

Dans cette partie, nous allons présenter un exemple de présentation de documents multimédia grâce auquel nous allons mettre en œuvre toutes les étapes de notre méthode en partant du modèle réseau de Petri jusqu'à l'obtention soit d'un ordonnancement temporel vérifiant les contraintes temporelles soit de la délimitation des domaines des dates de franchissement des transitions.

Nous avons repris un exemple présenté dans les travaux effectués au LAAS-CNRS par P.N.M. Sampaio [53, 52] qui utilisait une autre approche formelle décrite dans sa thèse [51].

Pour illustrer notre méthode, nous allons considérer le scénario multimedia qui décrit la présentation de deux médias successifs (M_1 et M_2) avec en parallèle un média interactif M_3. La présentation du média M_1 a une durée de présentation indéterminée $[0, +\infty[$, en unités de temps. Le média M_2, qui démarre lorsque le média M_1 se termine, possède une durée appartenant à l'intervalle $[10, 20]$. M_3 est un média interactif qui dure au maximum 40 unités de temps. L'aspect interactif vient du fait que l'utilisateur a la possibilité d'interrompre ce média entre 30 et 40 unités de temps après son début sans quoi sa présentation se terminera automatiquement lorsque son échéance maximale (40)

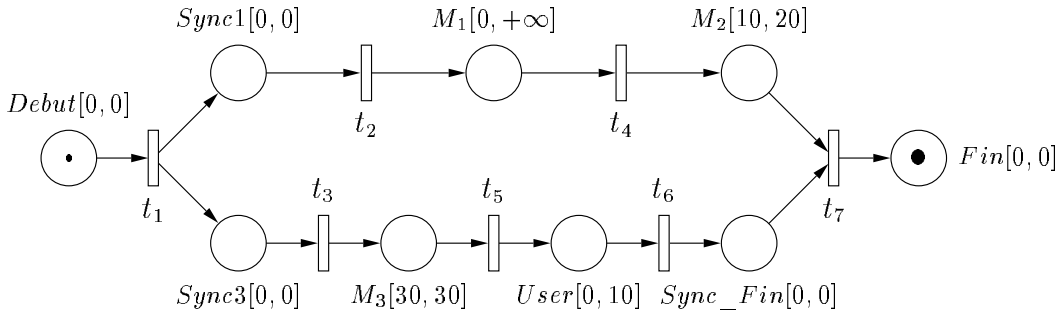

FIG. 4.10: Réseau de Petri p-temporel de l'application multimédia

sera atteinte. Il y a deux synchronisations dans ce scénario multimedia : une au lancement des présentations des médias M_1 et M_3 et une à l'arrêt des présentations des médias M_2 et M_3. D'après ces spécifications nous devons donc trouver une durée maximale de présentation du média M_1, initialement indéterminée, afin de pouvoir satisfaire toutes les contraintes de synchronisation de notre scénario multimedia.

À partir de ces spécifications, nous déterminons un modèle réseau de Petri p-temporel en portant les contraintes temporelles sur les places (figure 4.10). La branche composée des places $Sync1$, M_1 et M_2 traduit les présentations successives des deux médias précédées par une place montrant le démarrage synchronisé des deux branches parallèles du réseau. La branche composée des places $Sync3$, M_3, $User$ et $Sync\_Fin$ traduit la présentation du média M_3 avec l'interaction possible de l'utilisateur modélisée par la place $Sync\_Fin$. En effet, si la transition t_6 est franchie avant que la durée liée à la place $User$ n'atteigne sa valeur maximale (10 unités de temps), il faut immédiatement sensibiliser la transition t_7 pour pouvoir arrêter la présentation multimedia. Cela explique pourquoi nous avons associé un intervalle $[0,0]$ car nous ne souhaitons pas qu'il y ait d'attente dans cette place-là ($Sync\_Fin$). Cette branche débute comme l'autre par une place ($Sync3$) qui montre le démarrage synchronisé et instantané du média M_3 avec l'autre.

À partir du réseau de la figure 4.10, nous allons d'abord obtenir les ordres partiels. Pour cela nous allons construire un arbre de preuve en logique linéaire correspondant au problème d'accessibilité entre le marquage initial $M_0 = \{Debut\}$ et le marquage final $M_n = \{Fin\}$ pour le réseau de Petri composé des franchissements de transitions $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$. Ce problème d'accessibilité se traduit par le séquent : $\mathcal{M}_0, l_0 \vdash M_n$ avec $l_0 = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$.

La construction de la preuve de ce séquent nous permet de vérifier que M_n est bien accessible à partir de M_0 pour une séquence de franchissements donnée l_0. Tout au long de la construction de la preuve, nous étiquetons les franchissements de transition ainsi que les atomes consommés et produits par la méthode proposée au chapitre 2.

Plusieurs ensembles de relations de précédence peuvent être associés à un arbre de preuve construit. Dans notre exemple, comme il n'y a ni conflit de transitions ni conflit de

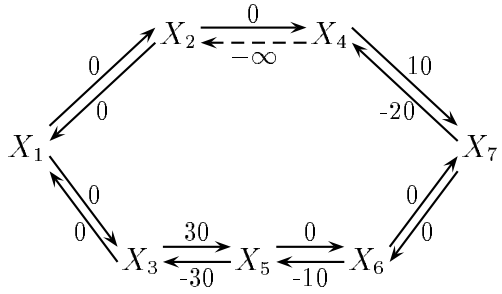

FIG. 4.11: Graphe AOA du scénario multimedia

jetons, il existe un seul ensemble de relations de précédence qui peut se traduire par un séquent caractéristique dont les atomes des marquages et les franchissements de transition sont étiquetés comme suit :

$Debut[I^1], t_1^1, t_2^1, t_3^1, t_4^1, t_5^1, t_6^1, t_7^1 \vdash Fin[t_7^1]$

avec :

$$\begin{array}{l|l} t_1^1 : Debut[I^1] \multimap Sync1[t_1^1] \otimes Sync3[t_1^1] & t_2^1 : Sync1[t_1^1] \multimap M_1[t_2^1] \\ t_3^1 : Sync3[t_1^1] \multimap M_3[t_3^1] & t_4^1 : M_1[t_2^1] \multimap M_2[t_4^1] \\ t_5^1 : M_3[t_3^1] \multimap User[t_5^1] & t_6^1 : User[t_5^1] \multimap Sync\_Fin[t_6^1] \\ t_7^1 : M_2[t_4^1] \otimes Sync\_Fin[t_6^1] \multimap Fin[t_7^1] & \end{array}$$

Nous pouvons désormais construire le graphe de précédence à partir de ce séquent caractéristique. Un graphe AOA représente un STP de contraintes binaires et temporelles. Nous pouvons donc construire directement le graphe AOA de la figure 4.11 associé à notre modèle p-temporel en respectant les relations de précédence du séquent caractéristique.

Une particularité de ce graphe AOA est la présence de l'arc étiqueté par $-\infty$ représentant la contrainte entre X_4 et X_2. Cet arc peut être éliminé. En effet, une contrainte de ce type sera toujours satisfaite. C'est pourquoi nous l'avons indiqué en traits pointillés.

Dans cet exemple, nous avons une contrainte *contrôlable* entre X_4 et X_2 qui correspond à la durée du média M_1 qui peut être restreinte ou instanciée pour respecter la cohérence du STP. En effet, la durée de ce média va dépendre de la durée des autres médias et surtout de l'interaction de l'utilisateur qui peut arriver n'importe quand entre la date $X_5 = 30$ et la date $X_6 = 40$. Ici, nous pouvons parler de date car l'origine temporelle de l'application (date $X_1 = 0$) est son lancement. La date de fin d'exécution du média M_3 étant égale à $X_1 + 30$ unités de temps. Les valeurs des variables X de notre STP sont d'ailleurs des dates et non des durées car les variables représentent les instants de franchissement de transition. La différence entre deux variables correspond à la durée d'exécution des tâches séparant les variables considérées. $X_6 = 40$ est la date maximale possible pour la fin du média M_3 mais elle peut varier en fonction de l'interaction de l'utilisateur. Nous avons donc une contrainte *contingente* entre X_6 et X_5 dûe à l'incertitude sur l'interaction de l'utilisateur, incertitude qui ne sera levée qu'au cours de l'exécution de l'application.

Il reste une dernière contrainte entre X_7 et X_4 qui correspond à la durée du média M_2 et qui elle aussi ne sera connue qu'en cours d'exécution de l'application. C'est donc une contrainte contingente. Cependant, elle n'est pas libre de choisir sa valeur sur son intervalle, c'est la contrainte entre X_6 et X_5 qui fixera sa valeur sans toutefois restreindre l'intervalle associé à M_2. C'est l'inverse de la contrainte entre X_4 et X_2 pour laquelle nous pouvons définir un intervalle de manière statique, c'est-à-dire avant l'exécution. Au cours de l'exécution, elle pourra prendre n'importe quelle valeur sur l'intervalle défini statiquement.

Pour résoudre ce problème temporel, nous commençons par énumérer toutes les inéquations du système linéaire correspondant au graphe AOA de la figure 4.11.

Le système initial suivant comporte toutes les inéquations entre deux variables successives pour tous les circuits possibles :

$X_2 - X_1 \geq 0$	$X_3 - X_1 \geq 0$	$X_6 - X_5 \geq 0$
$X_1 - X_2 \geq 0$	$X_1 - X_3 \geq 0$	$X_5 - X_6 \geq -10$
$X_2 - X_1 \geq -\infty$	$X_3 - X_1 \geq -30$	$X_6 - X_5 \geq -20$
$X_1 - X_2 \geq -30$	$X_1 - X_3 \geq -\infty$	$X_5 - X_6 \geq -\infty$
$X_4 - X_2 \geq 0$	$X_5 - X_3 \geq 30$	$X_7 - X_6 \geq 0$
$X_2 - X_4 \geq -\infty$	$X_3 - X_5 \geq -30$	$X_6 - X_7 \geq 0$
$X_4 - X_2 \geq 10$	$X_5 - X_3 \geq 0$	$X_7 - X_6 \geq -30$
$X_2 - X_4 \geq -30$	$X_3 - X_5 \geq -\infty$	$X_6 - X_7 \geq -\infty$
$X_7 - X_4 \geq 10$		
$X_4 - X_7 \geq -20$		
$X_7 - X_4 \geq -\infty$		
$X_4 - X_7 \geq -40$		

À partir de ce sytème, nous pouvons éliminer certaines inéquations afin de ne conserver que les celles exprimant les contraintes les plus fortes entre deux variables. Ainsi, par exemple, nous éliminons les inéquations $X_2 - X_1 \geq -\infty$ et $X_1 - X_2 \geq -30$ car elles sont moins contraignantes (plus faciles à satisfaire) que les inéquations $X_2 - X_1 \geq 0$ et $X_1 - X_2 \geq 0$. Le premier système réduit sera :

$X_2 - X_1 \geq 0$	$X_3 - X_1 \geq 0$	$X_6 - X_5 \geq 0$
$X_1 - X_2 \geq 0$	$X_1 - X_3 \geq 0$	$X_5 - X_6 \geq -10$
$X_4 - X_2 \geq 10$	$X_5 - X_3 \geq 30$	$X_7 - X_6 \geq 0$
$X_2 - X_4 \geq -30$	$X_3 - X_5 \geq -30$	$X_6 - X_7 \geq 0$
$X_7 - X_4 \geq 10$		
$X_4 - X_7 \geq -20$		

D'après ces inéquations, des simplifications peuvent être effectuées. En effet, la variable X_1 correspondant au franchissement de t_1 (première transition franchie dans le réseau)

est nulle car l'intervalle temporel associé à la place qui la sensibilise est $[0, 0]$. Nous en déduisons que :

$$X_2 = X_1 = 0$$
$$X_3 = X_1 = 0$$

Nous posons X_{123} comme la variable qui représente indifféremment le nœud X_1 ou X_2 ou X_3 donc $X_{123} = 0$. De même nous posons X_{67} comme la variable unique pour X_6 et X_7. Nous obtenons le nouveau système réduit suivant :

$$\begin{array}{rcr|rcr} X_{123} & = & 0 & X_5 - X_{123} & \geq & 30 \\ X_4 - X_{123} & \geq & 10 & X_{123} - X_5 & \geq & -30 \\ X_{123} - X_4 & \geq & -30 & X_{67} - X_5 & \geq & 0 \\ X_{67} - X_4 & \geq & 10 & X_5 - X_{67} & \geq & -10 \\ X_4 - X_{67} & \geq & -20 & & & \end{array}$$

Nous remplaçons X_{123} par sa valeur et simplifions ainsi le système précédent :

$$\begin{array}{rcr|rcr} X_4 & \geq & 10 & X_5 & = & 30 \\ X_4 & \leq & 30 & X_{67} - X_5 & \geq & 0 \\ X_{67} - X_4 & \geq & 10 & X_5 - X_{67} & \geq & -10 \\ X_4 - X_{67} & \geq & -20 & & & \end{array}$$

Cela nous amène au système minimal suivant en remplaçant X_5 par sa valeur numérique :

$$X_{67} - X_4 \geq 10 \tag{4.3}$$
$$X_4 - X_{67} \geq -20 \tag{4.4}$$
$$\text{avec} \quad 30 \leq X_{67} \leq 40 \tag{4.5}$$
$$\text{et} \quad 10 \leq X_4 \leq 30 \tag{4.6}$$

D'après ce dernier système nous obtenons le nouveau graphe AOA de la figure 4.12.

À partir des inéquations (4.1) et (4.2), nous pouvons vérifier si les encadrements obtenus en (4.3) et (4.4) sont cohérents. Pour cela, nous allons remplacer les variables dans les inéquations par les valeurs aux bornes de l'encadrement. Nous avons donc quatre combinaisons de valeurs possibles :

- $X_{67} = 30$ et $X_4 = 10$,
- $X_{67} = 40$ et $X_4 = 10$,
- $X_{67} = 30$ et $X_4 = 30$,
- $X_{67} = 40$ et $X_4 = 30$.

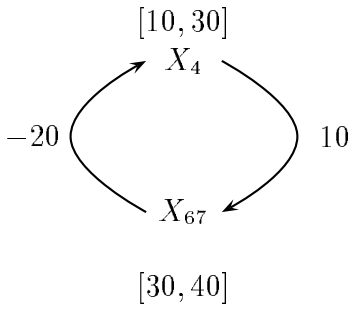

FIG. 4.12: Graphe AOA réduit

En remplaçant les variables dans les inéquations, nous remarquons que pour chacun des deux couples ($X_{67} = 40, X_4 = 10$) et ($X_{67} = 30, X_4 = 30$), une seule des deux inéquations est vérifiée. Cela indique que les bornes des encadrements des variables X_{67} et X_4 ne sont pas toujours adéquates. Pour que les bornes soient adéquates, il faut modifier la valeur de certaines.

Sachant que X_{67} est une variable liée à la contrainte contingente entre X_{67} et X_5, et que X_4 est une variable liée à la contrainte contrôlable entre X_4 et X_2, nous devons raisonner de manière à satisfaire *prioritairement* la contrainte contingente et donc à satisfaire tout d'abord les valeurs de X_{67}.

A partir des inéquations (4.1), (4.2) et du domaine incertain de X_{67}, nous avons deux cas possibles de valeurs pour X_4 :

- 1^{er} cas : nous pouvons trouver un domaine de valeurs pour X_4 pour lequel il existe au moins une solution possible pour X_{67}.
- $2^{è}$ cas : nous pouvons trouver au moins une valeur pour X_4 pour laquelle toutes les valeurs de X_{67} sont possibles.

Pour raisonner, nous allons travailler avec les bornes de X_{67}.

En utilisant l'inéquation (4.1) et en posant $X_{67} = 30$, nous nous plaçons dans le cas le plus contraint pour les valeurs possibles de X_4. Nous cherchons ainsi à trouver la plus petite borne maximale de X_4 : nous obtenons $X_4 \leq 20$. En utilisant l'inéquation (4.1) et en posant $X_{67} = 40$, nous nous plaçons dans le cas le moins contraint pour les valeurs possibles de X_4. Nous cherchons ainsi à trouver la plus grande borne maximale de X_4 : nous obtenons $X_4 \leq 30$.

Maintenant, en utilisant l'inéquation (4.2) et en posant $X_{67} = 30$, nous nous plaçons dans le cas le moins contraint pour les valeurs possibles de X_4. Nous cherchons ainsi à trouver la plus petite borne minimale de X_4 : nous obtenons $X_4 \geq 10$. En utilisant l'inéquation (4.2) et en posant $X_{67} = 40$, nous nous plaçons dans le cas le plus contraint pour les valeurs possibles de X_4. Nous cherchons ainsi à trouver la plus grande borne minimale de X_4 : nous obtenons $X_4 \leq 20$.

Dans le cas le plus contraint, nous trouvons l'encadrement $20 \leq X_4 \leq 20$ et dans le cas le moins contraint, nous trouvons $10 \leq X_4 \leq 30$. Le cas le plus contraint caractérise l'ensemble des valeurs de la variable X_4 pour lequel toutes les valeurs de X_{67} satisfont les contraintes de synchronisation. Par contre, l'encadrement $10 \leq X_4 \leq 30$, qui correspond au cas le moins contraint, caractérise l'ensemble des valeurs de X_4 pour lequel il existe au moins une valeur de X_{67} qui satisfasse les contraintes de synchronisation du scénario. En effet, si nous faisons une vérification par le calcul, nous voyons bien qu'en prenant X_4 égale à 10 et X_{67} égale à 40, nous ne vérifions pas les deux inéquations (4.1) et (4.2).

L'intervalle temporel de la place M_1 du réseau p-temporel de la figure 4.10 ne doit donc plus être $[0, \infty]$ mais l'intervalle de l'un des deux cas caractérisés ci-dessus suivant la flexibilité que le concepteur veut laisser à l'utilisateur pour interagir. Pour que l'utilisateur puisse interagir n'importe quand entre la date 30 et 40, il choisira $[20, 20]$ sinon il choisira $[10, 30]$. Dans ce dernier cas, l'utilisateur sera contraint à ne pouvoir éventuellement interagir dans la présentation multimedia qu'en un instant donné.

4.4 Conclusion

Dans ce chapitre, nous avons montré que le graphe de précédence obtenu par le biais d'un arbre de preuve en logique linéaire pouvait se traduire sous la forme d'un STP. Pour cela, il a fallu effectuer deux transformations :

- les franchissements de transition, représentés par les nœuds de notre graphe de précédence, en variables temporelles,
- les places de notre réseau de Petri représentés par les arcs de notre graphe de précédence, en contraintes temporelles.

Suivant le type de modèle utilisé, nous avons un ou deux arcs pour la contrainte temporelle. Pour un modèle type « temporisé », il y a un seul arc pour la durée associée à la transition ou à la place. Pour un modèle type « temporel », il y a deux arcs : un pour la contrainte minimale et un pour la contrainte maximale.

Il est plus *naturel* de faire une traduction à partir des réseaux de Petri p-temporisés et p-temporels car les activités sont associées aux places et les événements aux transitions. Cela paraît moins naturel avec les modèles t-temporisés et t-temporels car les contraintes temporelles sont représentées par les franchissements de transition habituellement utilisées pour représenter des événements, des changements d'état. Le graphe AOA généré par un modèle où le temps est associé aux places paraît ainsi plus simple.

Nous avons de plus rencontré une difficulté sérieuse avec les réseaux de Petri t-temporels car la détermination du graphe AOA nécessite de connaître quel est le dernier jeton arrivé pour chaque transition ayant plus d'une place en entrée. Cette information n'est clairement accessible que si nous considérons des séquences de franchissement et non des ordres partiels entre les franchissements. Ceci vient du fait que les graphes AOA ne peuvent pas exprimer des contraintes disjonctives.

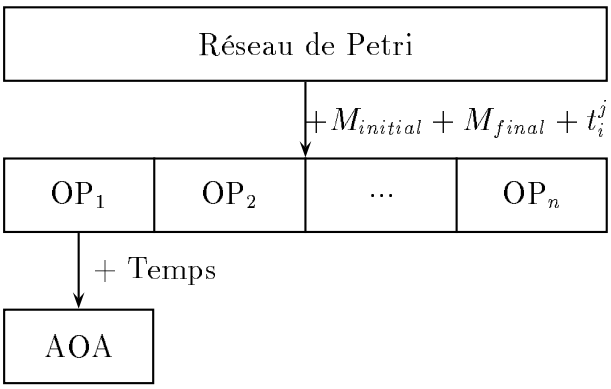

FIG. 4.13: Méthode globale

Nous avons ainsi montrer que l'utilisation conjointe des réseaux de Petri p-temporels, des graphes AOA et des STP nous permettait de trouver au moins une solution satisfaisante, s'il en existe une, à un problème d'ordonnancement en temps contraint pour chaque ordre partiel sans avoir à construire le graphe de tous les états possibles de notre modèle. La solution obtenue n'est bien sûr valable que pour l'ordre partiel généré par l'utilisation de la logique linéaire et dans lequel nous avons propagé les contraintes temporelles. Dans le cas où un conflit apparaît, autant d'ordres partiels que de chemins différents devront être générés pour lesquels nous pouvons trouver une solution qui sera différente pour chaque ordre partiel. La figure 4.13 montre bien comment il est possible de générer à partir de n'importe quel réseau de Petri un ensemble de graphes AOA pour un problème d'accessibilité donné.

Chapitre 5

Analyse temporelle symbolique et quantitative

5.1 Introduction

Dans un réseau de Petri t-temporel [37], les intervalles temporels associés aux transitions ne sont pris en compte qu'à partir de leur sensibilisation, c'est-à-dire que le compteur lié à la transition ne commence à compter le temps que lorsque tous les jetons de $Pre(t)$ sont présents.

L'intervalle de temps associé aux transitions correspond à une durée de sensibilisation imprécise. Ainsi, une transition t_i dont l'intervalle temporel est $[d_{imin}, d_{imax}]$ signifie que la transition sera franchie au plus tôt d_{imin} unités de temps après sa sensibilisation et au plus tard d_{imax} unités de temps après celle-ci. La transition est franchie instantanément entre ces deux bornes. Formellement, une durée de sensibilisation d_i est associée à chaque transition t_i du réseau. Le domaine de valeurs possibles pour la durée d_i est $\Delta_i = [d_{imin}, d_{imax}]$.

Nous avons vu au chapitre précédent que cela posait un problème pour exprimer les contraintes de type « date au plus tard » car elles ne s'expriment pas comme une conjonction de contraintes linéaires, c'est-à-dire entre deux événements de type franchissement de transition.

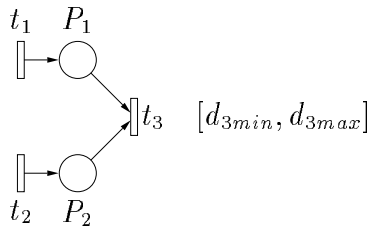

FIG. 5.1: Exemple de réseau t-temporel

En effet, considérons la figure 5.1. Supposons que la date de franchissement de t_1 soit délimitée par $[D_{1min}, D_{1max}]$ et celle de t_2 par $[D_{2min}, D_{2max}]$. La date de sensibilisation de t_3 au plus tôt est $max(D_{1min}, D_{2min})$ et la date D_3 de franchissement de t_3 doit vérifier les deux contraintes binaires suivantes de façon conjonctive :

$$\begin{aligned} D_3 &\geq D_{1min} + d_{3min} \\ D_3 &\geq D_{2min} + d_{3min} \end{aligned}$$

La date de sensibilisation de t_3 au plus tard est $max(D_{1max}, D_{2max})$ et la contrainte de franchissement au plus tard est : $D_3 \leq max(D_{1max}, D_{2max}) + d_{3max}$. Cette contrainte n'est pas équivalente à la conjonction de :

$$\begin{aligned} D_3 &\leq D_{1max} + d_{3max} \\ D_3 &\leq D_{2max} + d_{3max} \end{aligned}$$

qui est équivalente à : $D_3 \leq min(D_{1max}, D_{2max}) + d_{3max}$. C'est pour cette raison que l'ensemble des contraintes temporelles pour un scénario correspondant à un séquent caractéristique ne peut pas se mettre sous la forme d'un TCSP (nous rappelons que ce dernier représente un ensemble de contraintes binaires). Nous voyons que par contre, à l'aide des opérateurs *max* et $+$, nous pouvons écrire les contraintes sous une forme algébrique.

D'autre part, nous avons également vu au chapitre précédent que l' ensemble des contraintes temporelles définies par un réseau de Petri t-temporel était toujours cohérent à l'intérieur d'un scénario. Cela veut dire que le choix d'une date de franchissement pour une transition t_i ne peut pas rendre la contrainte concernant une transition t_j postérieure à t_i impossible à vérifier.

Étant donné que nous n'avons pas besoin de prendre en compte les contraintes finales, il n'est pas nécessaire de générer *a priori* les scénarios de franchissements de transitions. Les dates de début et fin de tâche peuvent ainsi être calculées *à la volée*.

Un algorithme permettant de calculer des dates de franchissement de transitions à la volée basé sur les réseaux de Petri t-temporisés [47] à partir de scénarios complètement scpécifiés, obtenus après la résolution de problèmes d'accessibilité de marquages, a été présenté dans [46]. Cet algorithme est basé sur la construction d'un arbre de preuve en logique linéaire dans le fragment MILL car comme le travail décrit dans le chapitre précédent, il s'appuie sur une recherche des relations de causalité. Étant donné que les durées sont associées aux transitions, les dates sont associées durant la construction aux atomes produits. Des dates « symboliques » sont utilisées afin de conserver toutes les relations de causalité entre les atomes consommés et les atomes produits par les franchissements de transition.

Dans ce chapitre nous étendons cette approche aux réseaux de Petri t-temporels car ils sont plus généraux que les t-temporisés. De plus les résultats obtenus avec les t-temporisés sont conservés avec les t-temporels à savoir la caractérisation des durées de scénario.

Nous nous plaçons d'abord dans un cadre de sémantique « faible » puis de sémantique « forte ». En effet, une accessibilité de marquage peut être vraie en sémantique faible et fausse en sémantique forte. Cette extension a été initiée par B. Pradin-Chézalviel et Robert Valette [45] et développée dans [50].

5.2 Les réseaux de Petri t-temporels : une approche symbolique

5.2.1 Sémantique forte et faible

Dans les réseaux t-temporisés, la valeur associée à chaque transition caractérise la durée de franchissement de cette transition. Pendant cette durée, les jetons ne sont ni dans les places d'entrée des transitions (ils sont réservés) ni dans les places de sortie. Les conflits ne peuvent apparaître qu'avant le début du franchissement de l'une des deux transitions. Les conflits sont résolus par des décisions extérieures et ils ne remettent pas en cause les durées quantitatives associées aux transitions.

Par contre, il existe deux sémantiques pour traiter les conflits dans les réseaux de Petri t-temporels : la sémantique faible et la sémantique forte. La sémantique forte est basée sur le fait qu'une transition sensibilisée t_i doit *nécessairement* être franchie *au plus tard* quand sa durée de sensibilisation atteint la borne maximale d_{imax} de l'intervalle. Elle ne peut rester sensibilisée au-delà, même si elle est en conflit avec une autre. Ainsi les conflits peuvent être résolus par les contraintes temporelles liées aux transitions. En sémantique faible, nous supposons que la durée de sensibilisation d_i peut prendre sa valeur sur tout le domaine Δ_i lors du franchissement de la transition t_i. Il est toutefois possible qu'une transition reste sensibilisée plus longtemps que d_{imax} si elle n'est pas tirée, dans ce cas elle ne restera pas franchissable.

Pour qu'une séquence de franchissements de transitions soit valide, il est nécessaire de considérer que les durées de sensibilisation appartiennent aux intervalles $[d_{imin}, d_{imax}]$ que ce soit dans le cas de la sémantique faible ou de la sémantique forte. Par contre, en présence d'un conflit, le traitement est résolu de manières radicalement différentes en fonction de la sémantique choisie.

Pour cela, considérons l'exemple de la figure 5.2 où nous avons un réseau de Petri t-temporel avec les transitions t_1 et t_2 en conflit. Nous pouvons avoir deux situations de marquage différentes.

Soit la situation où les dates de production des jetons sont égales à 0 pour les places P_A et P_B, et celle de P_C est égale à 2. Si nous nous plaçons dans le cadre de la sémantique forte, nous sommes obligés de franchir la transition t_1 et ce dans l'intervalle $[1, 3]$ car t_2 ne pourra être franchie que dans l'intervalle $[4, 7]$ dont toute valeur est supérieure à la borne maximale de l'intervalle associé à t_1. L'intervalle $[4, 7]$ est obtenu en additionnant la date à laquelle la transition t_2 commence à être sensibilisée (la date de production de P_C) aux

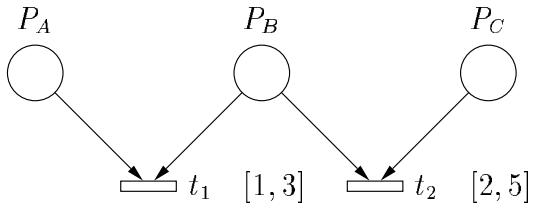

FIG. 5.2: Exemple de transitions en conflit

bornes de $[2,5]$. Par contre, si nous sommes dans le cadre de la sémantique faible, nous avons la possibilité de choisir de franchir t_2 dans l'intervalle $[4,7]$ en supposant qu'une décision externe bloque le franchissement de t_1 pour résoudre le conflit.

Maintenant, considérons la situation où les jetons des places P_A, P_B et P_C sont produits simultanément. Leur date de production est donc 0. Si nous nous plaçons dans le cadre de la sémantique forte, nous avons la possibilité de franchir t_1 dans l'intervalle $[1,3]$. La transition t_2 devra obligatoirement être franchie dans l'intervalle $[2,3]$ car t_1 ne peut être franchie après la date 3. Dans ce cas-là, les intervalles temporels n'ont pas pu résoudre le conflit de transitions même si l'intervalle de t_1 a eu une influence sur celui de t_2. Nous en déduisons que la sémantique forte résout les conflits de transitions, lorsqu'elle le peut, en tenant compte uniquement des valeurs temporelles. De plus, les intervalles réels de franchissement peuvent être modifiés.

Quant à la sémantique faible, les conflits sont toujours résolus par une décision *externe* car les transitions sont considérées indépendamment et sont seulement contraintes par leur propre intervalle temporel. Cela signifie que même en conflits, les transitions n'ont pas leur intervalle temporel modifié par les intervalles des autres transitions. Comme indiqué dans le paragraphe 4.2.3, la sémantique forte est donc en contradiction avec la linéarité de la logique linéaire qui suppose que si une séquence est franchissable depuis un marquage alors elle le reste même si des jetons sont ajoutés au marquage courant. En se replaçant dans l'exemple de la figure 5.2, la présence ou l'absence d'un jeton dans la place P_A n'a aucune influence sur le franchissement de t_2 dans le cadre de la sémantique faible, mais ce n'est pas le cas dans celui de la sémantique forte.

Par la suite, comme nous allons travailler avec des dates symboliques, nous ne pouvons de toute façon pas utiliser la sémantique forte car elle est inévitablement liée à un raisonnement basé sur l'utilisation des valeurs numériques des intervalles de sensibilisation pour résoudre les conflits entre les transitions. Nous nous situons donc dans le cadre de la sémantique faible pour notre approche symbolique. Les durées de sensibilisation d_i peuvent être choisies librement sur leur domaine Δ_i indépendamment des conflits. Comme nous l'avons indiqué au paragraphe 4.2.3, nous nous situons dans une sémantique opérationnelle multiserveur.

5.2.2 Arbres de preuves et dates symboliques

La construction des arbres de preuves dans notre approche symbolique se fait de la même manière que celle que nous avons présentée au chapitre 1. Elle se fait toujours pour des séquents de la forme $M_0, l_0 \vdash M_n$. Une fois qu'un arbre de preuve canonique d'un séquent est construit, nous avons vu qu'il était possible de mettre en évidence l'ordre partiel associé en associant une annotation aux atomes logiques et aux applications de la règle d'élimination à gauche de l'implication linéaire. L'ordre partiel est alors complètement défini par un arbre canonique annoté.

Nous avions choisi d'étiqueter chaque atome par le franchissement de transition qui l'avait produit. Étant donné que les intervalles temporels sont associés aux transitions, dorénavant, nous allons calculer nos dates directement à partir de cet arbre canonique annoté en remplaçant l'étiquette du franchissement de transition associée à l'atome par une étiquette temporelle contenant la date de production et la date de consommation de l'atome.

Pour le calcul symbolique et l'étiquetage associé, nous utilisons les notations suivantes :

- D_i pour les dates (en supposant que D_i est délimitée par $[D_{imin}, D_{imax}]$),
- d_j pour les durées de sensibilisation (d_j est délimitée par$[d_{jmin}, d_{jmax}]$).

Une paire (D_p, D_c) est donc associée à chaque atome de l'arbre de preuve : elle représente les dates de production et de consommation. Comme les franchissements de transitions sont instantanés dans les réseaux t-temporels, nous pouvons donner la définition suivante :

Définition 26 (Dates de production et de consommation) *La date de production D_p d'un atome est égale à la date de franchissement de la transition qui l'a produit et la date de consommation D_c est égale à la date de franchissement de la transition qui l'a consommé.*

Quand la date de consomation n'est pas connue (elle n'a pas encore été calculée), nous remplaçons celle-ci par un point pour l'indiquer : $(D_p, .)$. Ces paires apparaissent dans la partie principale du séquent car cela signifie que l'atome associé n'a pas encore été consommé. Les séquents identités générés dans la partie gauche de la preuve contiennent des atomes dont la paire (D_p, D_c) est complètement définie : ces séquents correspondent aux feuilles des arbres et comportent les informations temporelles des atomes. Cependant, pour garder une certaine cohérence avec l'annotation qualitative du chapitre 1, nous séparons cette paire en deux en mettant la date de production à gauche du séquent identité et la partie consommation à droite comme le montre le séquent identité : $A(D_p, .) \vdash A(., D_c)$.

Le calcul des dates dans l'arbre de preuve canonique est effectué de la façon suivante :

- Une date de production D_i est attribuée à tous les atomes du marquage initial M_0.
- Pour chaque application de la règle d'élimination à gauche de l'implication linéaire $\multimap_L$, la date de franchissement de cette transition est calculée : elle est égale au maximum des dates de production des atomes consommés augmenté de la durée de sensibilisation d_i associée à la transition considérée.

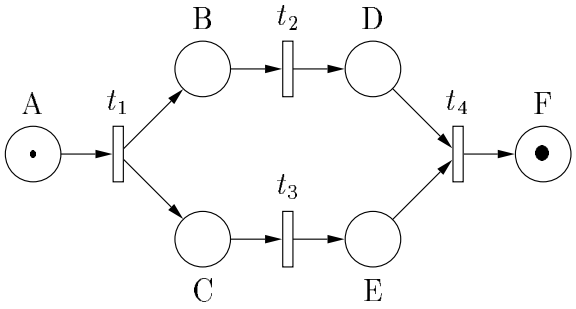

FIG. 5.3: Réseau de Petri exemple

– Une mise à jour des étiquettes temporelles est effectuée pour les atomes consommés et produits.

À partir des dates de production et de consomation des atomes de notre arbre de preuve canonique, nous sommes capables de calculer la durée du scénario correspondant à la preuve et les durées de séjour des jetons dans les places du réseau de Petri t-temporel associé. Nous pouvons donner les deux définitions suivantes :

Définition 27 (Durée de scénario) *La durée d'un scénario est égale à la différence entre le maximum des dates de production du marquage final M_n et la date de production du marquage initial M_0.*

Si les jetons du marquage initial ont des dates de production différentes, nous prendrons la date de production du dernier jeton produit.

Définition 28 (Durée de séjour) *Soit un atome A de la preuve avec une étiquette (D_p, D_c), sa durée de séjour dans la place A est égale à $D_c - D_p$.*

Exemple

Prenons l'exemple de la figure 5.3 dont l'arbre de preuve canonique, donné en annexe C, correspond au problème d'accessibilité traduit en logique linéaire par le séquent pour la séquence $\sigma = t_1; t_2; t_3; t_4$:

$$\underbrace{A}_{M_0}, \underbrace{t_1, t_2, t_3, t_4}_{l_0} \vdash \underbrace{F}_{M_n} \tag{5.1}$$

Soit D_{p_X} la date de production d'un atome X et D_{c_X} sa date de consommation. Nous reprenons l'arbre canonique de l'annexe C et nous réécrivons la preuve en y ajoutant les dates symboliques. Nous ne traiterons l'exemple sur l'arbre de preuve que pour le début de l'arbre de preuve. La suite se trouve dans l'annexe C. Au début de la preuve, le date de production du jeton A du marquage initial a une date initiale D_A qui joue le rôle d'un paramètre.

Nous avons d'abord le franchissement de la transition t_1 représenté par l'utilisation de la règle $\multimap_L$ qui produit deux séquents prémisses. Le séquent prémisse gauche est un séquent identité et le séquent prémisse droit qui est le séquent principal de la preuve.

$$\frac{\dfrac{}{A(D_A,.) \vdash A(.,D_A+d_1)}\,\mathrm{id} \quad B(D_A+d_1,.)\otimes C(D_A+d_1,.), t_2, t_3, t_4 \vdash F}{A(D_A,.), t_1, t_2, t_3, t_4 \vdash F}\multimap_L$$

La construction de la preuve avec les étiquettes de dates symboliques se construit de la même façon et donne la preuve canonique globale de l'annexe. Les expressions des dates symboliques correspondant à cette preuve se trouve en annexe C.

À partir de ces expressions symboliques, nous sommes capables de calculer la durée du scénario et la durée de séjour de chaque atome dans la place associée.

La durée du scénario est :

$$d_{scen} = d_1 + max(d_2, d_3) + d_4$$

Dans le cas de l'atome D, la durée de séjour dans la place est égale à $d_4 + max(d_2, d_3) - d_2$ qui est toujours supérieur ou égal à d_4. C'est cohérent car cela correspond bien à la durée de sensibilisation de la transition qui va consommer D.

Ces calculs de durées montrent bien l'utilité des dates symboliques. En effet, elle permet de comparer des dates mais aussi de tenir compte des relations de causalité en prenant en compte le passé commun inclus dans deux dates différentes. Par exemple la durée du scénario est indépendante de D_A et son évaluation ne sera donc pas entachée par l'imprécision associée à D_A.

5.2.3 Délimitation des domaines temporels

Le raisonnement temporel basé sur du calcul symbolique est lié aux valeurs que les variables symboliques peuvent prendre. Dans notre cas, il est important de délimiter les domaines sur lesquels les dates et les durées prennent leurs valeurs. Dans le cas des réseaux de Petri t-temporels, chaque durée de sensibilisation d_i prend ses valeurs sur un domaine temporel $\Delta_i = [d_{imin}, d_{imax}]$. Comme les dates et les durées calculées dépendent des d_i, leurs domaines seront aussi des intervalles. Dire que le domaine de d_i est l'intervalle temporel $[d_{imin}, d_{imax}]$ et que choisir une valeur de d_i sur ce domaine est *libre* (en effet, nous sommes dans le cadre de la sémantique faible) est équivalent à dire que d_i est une valeur imprécise délimitée par un intervalle disjonctif. L'imprécision est un cas particulier du domaine du *flou* et pour cela nous utilisons les opérations développées par D. Dubois et H. Prade dans [11] pour les intervalles flous afin de délimiter les domaines des valeurs possibles pour les dates et les durées.

Plusieurs opérations sont définies pour deux intervalles $A = [A_{min}, A_{max}]$ et $B = [B_{min}, B_{max}]$:

- $max(A,B) = [max(A_{min}, B_{min}), max(A_{max}, B_{max})]$,
- $min(A,B) = [min(A_{min}, B_{min}), min(A_{max}, B_{max})]$,
- $A + B = [A_{min} + B_{min}, A_{max} + B_{max}]$,
- $A - B = [A_{min} - B_{max}, A_{max} - B_{min}]$.

Considérons l'expression symbolique de la date de production de l'atome F de l'annexe C. Son domaine est l'intervalle temporel :

$$[D_A + d_{1min} + max(d_{2min}, d_{3min}) + d_{4min}, \\ D_A + d_{1,max} + max(d_{2max}, d_{3max}) + d_{4,max}] \tag{5.2}$$

Une première remarque est comme les durées sont des nombres positifs, leur domaine doit être restreint aux valeurs positives. Une seconde remarque est que travailler avec des expressions symboliques réduit l'imprécision. Si des activités sont exécutées en série, l'imprécision est accrue étant donné que l'imprécision de chaque durée de sensibilisation de transition est ajoutée à chaque franchissement. Le fait d'utiliser des expressions symboliques pour le calcul des durées permet d'éliminer les imprécisions passées lorsque nous nous intéressons à des durées et non à des dates. Par exemple, la durée de séjour de l'atome C dans la place est égale à d_3 et prend ses valeurs sur le domaine $[d_{3min}, d_{3max}]$. Dans le calcul de cette durée, nous avons éliminé l'imprécision de d_1. Si nous avions calculé la durée à partir des valeurs numériques des dates de production et de concommation, nous aurions eu une influence provenant de l'imprécision de $d_1(d_{1max} - d_{1min})$. Le domaine aurait été :

$$[d_{3min} + d_{1min} - d_{1max}, d_{3max} + d_{1max} - d_{1min}]$$

La logique linéaire nous permet donc d'avoir une meilleur évaluation des durées de séjours dans les places des réseaux de Petri t-temporels en exploitant au mieux les relations de causalité.

5.3 Raisonnement quantitatif en sémantique faible

5.3.1 Notions de marquages potentiels

Nous venons de voir que des dates symboliques pouvaient être calculées aisément pour chaque atome au cours de la construction de l'arbre de preuve canonique. Cependant, comme nous n'utilisons pas de marquages au cours de la preuve (à l'exception du marquage initial et final), nous n'avons aucune information sur l'accessibilité des marquages intermédiaires. Nous allons maintenant montrer qu'il est possible d'obtenir des informations sur l'accessibilité de ces marquages à partir des dates symboliques des atomes logiques. De plus, si un marquage est accessible, des informations sur ses dates de production et de consommation peuvent être obtenues. Étant donné que nous ne savons pas si ces marquages sont accessibles, nous les appelons *marquages potentiels*.

Définition 29 (Marquage potentiel) *Un marquage potentiel est un ensemble d'atomes. Chaque atome doit apparaître au moins une fois dans une des étapes de l'arbre de preuve.*

Un marquage potentiel est une manière de considérer un ensemble d'atomes afin de pouvoir étudier son accessibilité. Ces atomes n'ont pas nécessairement besoin d'apparaître à la même étape de la preuve. Toutefois, comme pour toute traduction de marquage en logique linéaire nous le noterons comme un monôme en $\otimes$. Un marquage potentiel est accessible si c'est un fragment (marquage partiel) d'un marquage qui appartient à une trajectoire vérifiant les contraintes temporelles entre le marquage initial et final du scénario représenté par le séquent conclusion. Un exemple de marquage potentiel pour notre scénario représenté par le séquent 5.1 est le suivant :

$$\begin{aligned} &B(D_A + d_1, D_A + d_1 + d_2) \otimes \\ &E(D_A + d_1 + d_3, D_A + d_1 + \max(d_2, d_3) + d_4) \end{aligned} \tag{5.3}$$

5.3.2 Dates et durées de séjour des marquages potentiels

La méthode présentée jusqu'ici permet d'attribuer des dates symboliques de production et de consommation à chaque atome utilisé dans le scénario étudié. Cependant elle n'apporte aucune information de ce type sur les marquages car ils ne sont pas manipulés explicitement. Toutefois, il est possible d'obtenir une telle information en utilisant les opérateurs simples *min* et *max* avec les étiquettes des atomes. De plus, le fait q'un marquage potentiel soit accessible ou non pourra être déduit à partir de l'intersection des intervalles temporels des atomes présent dans le marquage.

Définition 30 (Dates d'un marquage potentiel) *La date de production D_{pm} d'un marquage potentiel est égale au maximum des dates de production de tous les atomes qui composent ce marquage. La date de consommation D_{cm} d'un marquage potentiel est égale au minimum des dates de consommation de tous les atomes qui composent ce marquage.*

Considérons le marquage de la formule 5.3. Sa date de production est :

$$\begin{aligned} D_{pm} &= max(D_A + d_1, D_A + d_1 + d_3) \\ &= D_A + d_1 + d_3 \end{aligned} \tag{5.4}$$

et sa date de consommation est :

$$\begin{aligned} D_{cm} &= min(D_A + d_1 + d_2, D_A + d_1 + d_4 + max(d_2, d_3)) \\ &= D_A + d_1 + d_2 \end{aligned} \tag{5.5}$$

À partir de ces dates, il est possible d'obtenir la durée de tels marquages. En effet, comme pour les atomes, nous avons la définition :

Définition 31 (Durée de séjour d'un marquage potentiel) *La durée de séjour d'un marquage potentiel est définie par $D_{cm} - D_{pm}$.*

La différence majeure par rapport à la durée de séjour d'un atome dans une place est que la durée de séjour d'un marquage potentiel n'est pas nécessairement positive. Étant donné que le calcul des dates de production et de consommation des atomes est basé sur un arbre de preuve canonique en logique linéaire nous avons la garantie qu'un atome ne peut être consommé avant d'être produit. Cette garantie n'existe pas pour les marquages potentiels car nous ne sommes par sûrs qu'un marquage soit effectivement accessible pour un jeu de valeurs numériques donné étant donné qu'il repose entièrement sur des expressions symboliques des durées associées aux transitions.

Considérons le marquage potentiel 5.3, à partir des expressions de D_{pm} et de D_{cm} données en 5.4 et en 5.5, nous obtenons une durée de séjour égale à $d_2 - d_3$.

5.3.3 Accessibilité d'un marquage potentiel

Á partir du calcul de la durée de séjour d'un marquage potentiel, nous pouvons énoncer la propriété suivante :

Propriété 1 *Soit un séquent définissant un scénario complètement spécifié. Si la durée de séjour d'un marquage potentiel est strictement positive alors il est accessible dans le scénario.*

En effet dans un tel cas tous les jetons du marquage potentiel seront présents simultanément pendant la durée de séjour du marquage potentiel.

À partir de la définition 31, la durée de séjour d'un marquage potentiel est la différence entre deux dates qui sont des expressions formelles des durées d_i associées aux transitions. Nous pouvons considérer trois cas de figures :

- l'expression symbolique de la durée de séjour peut être prouvée positive, alors le marquage potentiel est structurellement accessible (pour toute valeur de d_i pour tout i),
- l'expression symbolique de la durée de séjour peut être prouvée négative, alors le marquage potentiel est structurellement non accessible (pour toute valeur de d_i pour tout i),
- l'accessibilité du marquage potentiel dépend des valeurs des durées de sensibilisation d_i.

Pour le marquage potentiel 5.3, nous sommes dans le troisième cas : si d_2 est plus grand que d_3 alors il est accessible. L'expression $d_2 - d_3 > 0$ est la condition nécessaire et suffisante d'accessibilité du marquage correspondant.

La durée du marquage potentiel

$$\begin{aligned} &B(D_A + d_1, D_A + d_1 + d_2) \otimes \\ &D(D_A + d_1 + d_2, D_A + d_1 + \max(d_2, d_3) + d_4) \end{aligned} \tag{5.6}$$

est $(D_A + d_1 + d_2) - (D_A + d_1 + d_2) = 0$. Il n'est pas accessible quelle que soit les valeurs des d_i.

Soit le marquage potentiel suivant :

$$D(D_A + d_1 + d_2, D_A + d_1 + \max(d_2, d_3) + d_4) \otimes \\ E(D_A + d_1 + d_3, D_A + d_1 + \max(d_2, d_3) + d_4) \quad (5.7)$$

Sa durée de séjour est égale à d_4 et le marquage potentiel est accessible pour n'importe quelle valeur des d_i.

5.3.4 Étude paramétrique de l'accessibilité des marquages potentiels

Comme pour le cas des dates de production et de consommation des atomes et de leur durée de séjour, nous pouvons utiliser la délimitation des domaines de valeurs des durées d_i afin de délimiter les domaines des dates de production et de consommation des jetons et des durées de séjours des marquages potentiels en fonction de certains paramètres considérés comme des variables commandables du système.

Nous nous sommes attaché à délimiter les durées de séjour car nous sommes intéressés par la caractérisation de l'accessibilité des marquages potentiels. Soit $[d_{pm\_min}, d_{pm\_max}]$ le domaine de la durée de séjour d'un marquage potentiel M_{pm} obtenu en utilisant les règles introduites pour la délimitation du domaine de séjour des atomes en 5.2.3. Nous devons analyser l'impact d'une condition d'accessibilité (ou d'inaccessibilité) sur les bornes du domaine. Pour cela nous considérons que la borne inférieure (ou la borne supérieure) est un paramètre.

Nous retrouvons les trois cas de la section précédente :

- si la borne d_{pm_min} est positive alors le marquage potentiel M_{pm} est nécessairement accessible, et ce quelles que soient les valeurs des durées d_i sur leur domaine Δ_i, et donc quelle que soit la valeur du paramètre.
- si d_{pm_max} est négative alors le marquage potentiel M_{pm} est nécessairement non accessible, et ce quelles que soient les valeurs des durées d_i sur leur domaine Δ_i. En d'autres termes, il n'est pas *possiblement* accessible.
- dans les autres cas, il est possible d'atteindre le marquage potentiel M_{pm}, c'est-à-dire qu'il est possible de trouver des valeurs de d_i dans l'intervalle Δ_i telles que M_{pm} soit accessible et telles que d'autres ne le soient pas. Nous pouvons alors obtenir une condition sur le paramètre (ou les paramètres).

Considérons l'exemple du marquage potentiel 5.3 avec des valeurs spécifiques :

$D_A = 0$, d_1 appartient à $[1, 5]$, d_2 appartient à $[1, \alpha]$ $(\alpha \geq 1)$, d_3 appartient à $[2, \beta]$ $(\beta \geq 2)$ et d_4 appartient à $[2, 5]$.

Comme la durée de séjour du marquage potentiel 5.3 est $d_2 - d_3$, l'accessibilité dépend des valeurs des durées de sensibilisation d_i (pas d'accessibilité structurelle ni d'inacessibilité). Les bornes du domaine de la durée de séjour sont $d_{BE\_min} = 1-\beta$ (toujours négative) et $d_{BE\_max} = \alpha - 2$. Ce marquage est possiblement accessible si $\alpha \geq 2$ et inaccessible si $\alpha < 2$. L'accessibilité est donc indépendante de β.

5.4 Invalidation de scénarios et conflits en sémantique forte

Tout les calculs et résultats précédents ont été effectués dans le cadre de la sémantique faible des réseaux de Petri t-temporels car elle est cohérente avec la monotonie de la logique linéaire : l'ajout de nouveaux jetons n'invalide pas des scénarios possibles. Toutefois, il est fréquemment nécessaire d'avoir recours à la sémantique forte, pour vérifier par exemple si le comportement d'un ensemble de *chiens de garde* est correct. En effet, un chien de garde sert à imposer un choix dans le cas où des contraintes temporelles quantitatives ne seraient pas respectées.

Pour cela, nous allons utiliser les deux sémantiques de manière complémentaire. La sémantique faible sert à construire des scénarios et à calculer les dates et les durées de séjour d'atomes et de marquages potentiels. La sémantique forte est utilisée pour essayer de montrer que certains scénarios sont invalides.

5.4.1 Invalidation potentielle de scénarios

Définition 32 (Invalidation potentielle) *Soient deux scénarios Sc_i et Sc_j. Le scénario Sc_j invalide potentiellement Sc_i si et seulement si :*

- *Sc_i et Sc_j possèdent le même état courant initial,*
- *Sc_j contient au moins une transition t_j qui n'appartient pas à Sc_i,*
- *tous les atomes consommés par le franchissement de la transition t_j (application de la règle $\multimap_L$) apparaissent dans Sc_i comme des atomes non terminaux.*

Prenons l'exemple des deux scénarios suivants pour la figure 5.4 :

$$Sc_1 \quad : \quad A(D_A, .), t_1, t_2, t_3, t_4 \vdash F \tag{5.8}$$

$$Sc_2 \quad : \quad A(D_A, .), t_1, t_3, t_5 \vdash G \tag{5.9}$$

Le scénario Sc_1 invalide potentiellement Sc_2 car la transition t_5 n'appartient pas à Sc_1 mais elle consomme des atomes de ce scénario. En effet, t_5 consomme dans Sc_2 les atomes $B(D_A + d_1, .)$ et $E(D_A + d_1 + d_3, .)$ qui apparaissent aussi dans Sc_1. Dans cet exemple, nous avons de manière symétrique le scénario Sc_2 qui invalide potentiellement Sc_1 car la transition t_2 consomme $B(D_A + d_1, .)$ qui apparaît dans Sc_2. Remarquons que

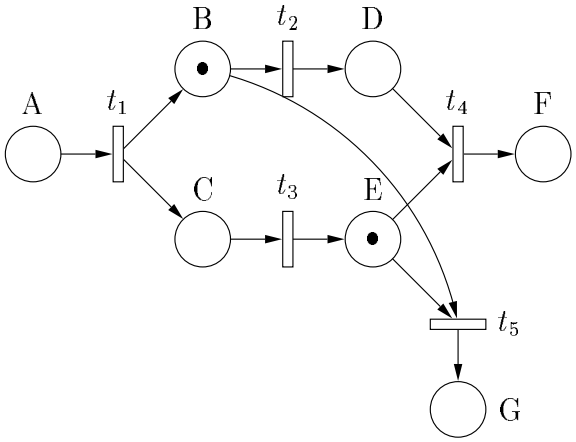

FIG. 5.4: Conflit de transitions d'un réseau de Petri

la transition t_4, qui n'appartient pas à Sc_2, ne joue oas exactement le même rôle que la transition t_2 car t_4 ne consomme qu'un seul atome de Sc_2 : $E(D_A + d_1 + d_3, .)$. Le jeton $D(D_A + d_1 + d_2, .)$ n'appartient pas à Sc_2.

Il est clair que quand un scénario Sc_j invalide potentiellement un scénario Sc_i par l'intermédiaire d'une transition t_j, c'est parce que cette transition est en conflit avec une transition de Sc_i. Pour qu'il y ait une invalidation, il ne faut pas seulement que le conflit soit structurel. En effet, si le marquage qui sensibilise simultanément deux transitions en conflit structurellement n'est pas accessible, il n'y aura aucune décision à prendre concernant le choix de la transition à franchir. Dans le cadre de réseaux de Petri t-temporels, les intervalles liés aux transitions peuvent effectivement rendre ce marquage inaccessible.

Sur l'exemple du réseau de la figure 5.4, les transitions t_5 et t_2 sont en conflits. Nous avons vu précédemment que dans le cadre du scénario Sc_1 le marquage $B \otimes E$ n'était accessible que si nous avions $d_2 - d_3 > 0$. Donc si nous avons $d_{2max} < d_{3min}$, le marquage ne sera jamais atteint et le conflit entre t_5 et t_2 ne sera pas effectif.

Dans notre approche, l'accessibilité est toujours caractérisée pour un scénario, c'est pourquoi nous devons considérer une notion de scénarios en conflit. Le marquage initial des deux scénarios est égal car nous voulons étudier le cas d'un comportement qui dévie du scénario normal en raison de la violation d'une contrainte temporelle à l'instar des chiens de garde. Finallement, nous nous restreignons au cas où tous les atomes consommés par une transition t_j sont des atomes de Sc_i car notre analyse est toujours faite pour un scénario.

5.4.2 Invalidation stricte de scénarios

La sémantique forte impose qu'une transition t_j sensibilisée ne peut le rester une fois que la borne maximale de sa durée de sensibiliation a été atteinte. Nous pouvons poser la

propriété suivante.

Propriété 2 *Considérons deux scénarios Sc_i et Sc_j tels que Sc_j invalide potentiellement Sc_i. Soit M_j le marquage potentiel qui sensibilise la transition t_j de Sc_j. Si la valeur maximale de la durée de sensibilisation de t_j est inférieure à la durée de séjour minimale de M_j dans Sc_i, alors le scénario Sc_j invalide de manière effective Sc_i dans le cadre de la sémantique forte.*

L'invalidation effective de Sc_i par Sc_j signifie que pour toutes les valeurs des durées de sensibiliastion d_k choisies sur leur domaine Δ_k, pour toutes les transitions t_k telles que la condition soit satisfaite, il est impossible de construire une séquence de franchissement correcte correspondant à Sc_i dans le cadre de la sémantique forte. La séquence sera nécessairement interrompue par le franchissement de t_j. La séquence de franchissement correspondra éventuellement à Sc_j mais pas à Sc_i.

Prenons l'exemple des scénarios 5.8 et 5.9. Si le domaine de la durée de sensibilisation de la transition t_5 est $[d_{5min}, d_{5max}]$, alors sa valeur maximale est d_{5max}. La durée de séjour du marquage potentiel $B \otimes E$ dans Sc_1 est égale à $d_2 - d_3$. Le domaine de cette durée est $[d_{2min} - d_{3max}, d_{2max} - d_{3min}]$ et sa valeur minimale est $d_{2min} - d_{3max}$. Par conséquence, si la condition

$$d_{5max} < d_{2min} - d_{3max} \tag{5.10}$$

est vérifiée alors Sc_2 invalide de manière effective Sc_1 et dans le cadre de la sémantique forte, aucune séquence de franchissements comprenant Sc_1 ne peut avoir lieu. La transition t_5 sera toujours franchie avant t_2.

Inversement, le scénario Sc_1 invalide de manière effective le scénario Sc_2 quand la valeur maximale de la durée de sensibiliation de t_2 est inférieure à la durée de séjour minimale du marquage potentiel B dans Sc_2. Ce marquage potentiel est produit à la date $D_A + d_1$ et consommé à la date $D_A + d_1 + d_3 + d_5$ dans le scénario Sc_2 et sa durée de séjour minimale est $d_{3min} + d_{5min}$. En conséquence, la condition est :

$$d_{2max} < d_{3min} + d_{5min} \tag{5.11}$$

Quand la condition 5.10 est vérifiée, le scénario Sc_1 ne peut avoir lieu car il est invalidée de manière effective par Sc_2. Quand les conditions 5.10 et 5.11 sont fausses, les deux scénarios Sc_1 et Sc_2 peuvent avoir lieu pour les transitions de Sc_1 et de Sc_2 en prenant des durées d_k adéquates sur les domaines Δ_k. Quand la condition 5.11 est vraie, ce qui implique mathématiquement que la condition 5.10 soit fausse, alors le scénario Sc_2 ne peut avoir lieu en raison de son invalidation effective par Sc_1.

5.4.3 Caractérisation des scénarios conflictuels

Dans la définition 32, nous avons considéré un cas restreint de conflits entre scénarios. Il se trouve qu'il est possible que le scénario Sc_j contienne une transition t_j dont le

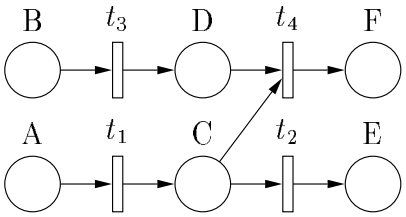

FIG. 5.5: Un réseau de Petri modélisant un chien de garde

franchissement consomme des atomes qui n'appartiennent pas au scénario Sc_i en plus des atomes qui lui appartiennent.

Prenons l'exemple du modèle de la figure 5.5 et les deux scénarios suivants :

$$Sc_3 \quad : \quad A(D_A,.) \otimes B(D_B,.), t_1, t_2 \vdash B \otimes E \tag{5.12}$$

$$Sc_4 \quad : \quad A(D_A,.) \otimes B(D_B,.), t_1, t_3, t_4 \vdash F \tag{5.13}$$

Bien que les deux transitions t_2 et t_4 soient en conflit à cause de leur place d'entrée C partagée, le scénario Sc_4 n'invalide pas de manière potentielle Sc_3 car l'autre place d'entrée de t_4, D, n'appartient pas au scénario Sc_3 (l'atome D n'appartient pas à l'arbre de preuve canonique correspondant à Sc_3).

Il en est de même avec le scénario Sc_5 suivant :

$$Sc_5 : A(D_A,.) \otimes B(D_B,.), t_1, t_2, t_3 \vdash D \otimes E \tag{5.14}$$

En effet, la place d'entrée D de t_4 appartient seulement au marquage final du séquent d'accessibilité correspondant à Sc_5. Elle apparaît sous la forme d'un atome dans une feuille terminal de l'arbre de preuve sans date de consommation. Ce scénario ne peut pas invalider le scénario Sc_3.

Dans les scénarios Sc_3 et Sc_5, étant donné qu'il n'y a pas de date de consommation pour D, il ne sera pas possible d'obtenir une durée de séjour symbolique pour le marquage potentiel $C \otimes D$ et de la comparer avec la durée de sensibilisation de t_4 du scénario Sc_4. Pourtant, il sera toujours possible de choisir une valeur de D_B telle que la transition t_2 soit franchie avant t_4. La résolution de conflit ne peut pas être effectuée localement entre un scénario Sc_i et la durée de sensibilisation de la transition dont le franchissement invalide Sc_i. Cela implique donc l'analyse des deux scénarios Sc_i et Sc_j et une énumération des états qui se rapproche plus de la construction d'un graphe de classes que d'un arbre de preuve. Ceci est dû au fait que la sémantique forte viole la monotonie classique de la théorie des réseaux de Petri : une séquence de franchissements possibles peut être invalidée par l'introduction d'un nouvel atome. La linéarité de la logique linéaire correspond à cette monotonie des réseaux de Petri. Cela explique pourquoi la sémantique forte ne peut être réintroduite dans notre approche que si la résolution du conflit peut être faite dans un scénario.

Cependant nous pouvons remarquer que le scénario Sc_3 invalide potentiellement le scénario Sc_4 au travers de t_2 car C appartient à Sc_4. La relation n'est pas symétrique car Sc_4 n'invalide pas Sc_3 au travers de la transition t_4. Nous pouvons ajouter que le réseau de la figure 5.5 représente un cas typique de *mécanisme à chien de garde* où le scénario Sc_4 représente l'évolution normale avec le franchissement de t_4, avec un jeton qui arrive dans la place D dans un délai raisonnable, et le scénario Sc_3 représente le déclenchement du chien de garde (une alarme en général) en franchissant la transition t_2 si le jeton n'était pas dans la place D en temps voulu. L'analyse typique de ces mécanismes consiste à vérifier que le chien de garde invalide de manière effective le fonctionnement nominal du système modélisé. Dans notre exemple, nous cherchons à savoir quand Sc_3 invalide Sc_4 ce qui correpond bien au cas que nous voulons étudier. D'une manière générale, cela signifie que la possibilité de réintroduire « ponctuellement » la sémantique forte dans notre approche s'avère être d'une utilité pratique.

5.5 Conclusion

Dans ce chapitre, nous avons montré qu'il était possible de donner des informations sur l'accessibilité et les conflits par le biais d'une méthode utilisant les réseaux de Petri t-temporels et la logique linéaire.

Pour cela, nous nous sommes appuyés sur deux approches différentes : la sémantique faible et la sémantique forte. La sémantique faible a été utilisée afin de respecter la linéarité de la logique linéaire et d'être cohérent avec la monotonicité des réseaux de Petri pour pouvoir résoudre le problème de l'accessibilité. En effet, comme travailler avec des valeurs numériques augmente l'imprécision dans le calcul des dates et des durées, nous avons choisi de construire des arbres de preuve canoniques avec des expressions symboliques (attachées aux atomes). De plus cela nous permet de conserver les informations sur les relations de causalité entre les prodcutions et les consommations des atomes. L'approche utilisant directement la sémantique forte aurait imposé l'utilisation des valeurs numériques ce qui aurait forcément accru l'imprécision sur les dates et les durées. Un point important que nous retrouvons et qui est un des avantages de l'utilisation de la logique linéaire est que nous n'avons pas besoin de traiter le parallélisme par entrelacement pour caractériser l'accessibilité d'un marquage en utilisant les dates symboliques dans nos arbres de preuves.

Une fois que nous avons obtenu l'expression temporelle symbolique de chaque atome, nous avons pu effectuer des raisonnements afin de délimiter les domaines des variables de chaque atome de l'arbre de preuve. À l'aide des intervalles temporels imprécis de durée de sensibilisation, nous avons caractérisé chaque variable de manière précise par un domaine : un domaine pour la date de production D_p et un domaine pour la date de consommation D_c. Un domaine caractérisant la durée de séjour de chaque atome peut ainsi être trouvé à partir de ces deux domaines. Ces domaines ont été trouvés pour des atomes et non des marquages dans un premier temps.

Par la suite, nous avons montré qu'il était possible de donner une caractérisation temporelle sur l'accessibilité des marquages en calculant une date de production D_{pm} et une date de consommation D_{cm} pour un marquage potentiel à partir des dates symboliques des atomes qui le compose. Comme pour les atomes, nous trouvons une durée de séjour pour un marquage potentiel. Les marquages potentiels que nous caractérisons sont *effectivement* accessibles si leurs durées de séjour ne sont pas négatives. Ainsi, il est possible de caractériser différents degrés d'accessibilité qui peuvent dépendre ou pas des valeurs numériques des durées d_i de sensibilisation des transitions. L'accessibilité structurelle est la plus forte car elle est vraie quelles que soient les valeurs des intervalles temporels associés aux transitions.

Dans un second temps, nous avons montré comment la sémantique forte pouvait entraîner la résolution de conflits temporels tout en utilisant les dates et les durées obtenues avec la sémantique faible. Pour cela nous avons choisi deux exemples où il y avait plusieurs scénarios d'exécution possibles afin de montrer qu'un scénario pouvait en invalider un autre mais aussi que des conflits entre eux pouvaient apparaître. Pour cela nous avons comparé les durées des marquages potentiels et les durées de sensibilisation des transitions en conflit pour caractériser ces problèmes. Étant donné que le calcul de dates symboliques s'effectue dans le cadre d'un scénario particulier, nous devons d'abord calculer les étiquettes temporelles de chaque scénario en conflit.

L'exemple du mécanisme à chien de garde montre bien que pour mettre en évidence une violation de contrainte temporelle sur une transition dans un réseau de Petri t-temporel, l'existence d'un scénario conflictuel est nécessaire.

Conclusion

Afin d'analyser les contraintes temporelles d'applications temps réel distribuées, nous avons présenté une approche fondée sur les réseaux de Petri, la logique linéaire et les graphes de contraintes temporelles. Cette approche n'a pas pour objectif de simplement montrer qu'une propriété est vraie ou non, elle donne au concepteur des outils d'analyse permettant de préciser les domaines de certains paramètres temporels pour que la propriété soit vraie.

Après un rapide état de l'art des approches existantes, nous avons montré comment une preuve de séquent en logique linéaire pouvait permettre d'obtenir les relations de causalité entre les événements appartenant à un scénario donné. Grâce au chapitre suivant qui relie ces relations de causalité à celles obtenues par dépliage d'un réseau de Petri pour un marquage initial, nous pouvons conclure que ces relations peuvent être vues comme des relations de précédence entre les franchissements des transitions.

Le cœur de l'algorithme de construction de l'arbre de preuve est, en fait, identique à celui de l'algorithme de dépliage lorsque nous recherchons des processus de réseau de Petri finis. La première différence est que la preuve porte sur un séquent spécifiant un marquage initial, un marquage final et une liste de franchissements de transitions alors que le processus de dépliage ne présuppose que la connaissance du marquage initial. Ceci pose un certain nombre de problèmes en particulier dans le cas des réseaux de Petri non bornés : alors que l'arbre de preuve reste borné, le dépliage sera potentiellement infini. La seconde différence est que la notion de séquent caractéristique qui caractérise un seul ensemble de relations de précédence, nous donne une notation simple pour les processus finis et nous permet de définir des règles de composition de processus à partir des règles du calcul des séquents. Une approche compositionnelle peut ainsi être facilement mise en œuvre.

Nous avons ensuite montré comment, pour un certain nombre d'extensions temporelles des réseaux de Petri, il est possible de passer du graphe décrivant les relations de précédence à des graphes de contraintes temporelles exprimant de façon linéaire l'ensemble des contraintes temporelles quantitatives que doivent vérifier les dates des franchissements des transitions dans un scénario (décrit par un séquent caractéristique). Un résultat important de ce chapitre est que cette démarche est complètement cohérente avec les réseaux de Petri p-temporels mais qu'elle est difficilement compatible avec les réseaux de Petri

t-temporels. Les ensembles de contraintes engendrés par les réseaux de Petri t-temporels sont plus complexes que ceux engendrés par les réseaux p-temporels car ils comprennent des disjonctions entre certaines contraintes alors que les seconds ne comprennent que des conjonctions de contraintes binaires (n'impliquant que deux variables). Ce n'est donc que dans le cadre des réseaux de Petri p-temporels qu'il est possible d'exploiter tous les résultats des techniques classiques d' analyse et de propagation de contraintes pour l'étude d'un scénario. Ceci a été illustré par un problème simple d'ordonnancement de documents multimédias.

Les réseaux de Petri t-temporels sont paradoxaux dans le sens où les contraintes temporelles sont complexes mais que malgré tout, du moins dans le cas de la sémantique faible, une séquence de franchissements de transitions franchissable dans le réseau de Petri ordinaire sous-jacent reste franchissable dans le réseau t-temporel. Il sera toujours possible de trouver un ensemble de dates cohérent avec les contraintes engendrées par les durées de sensibilisation associées aux transitions alors que cela est faux de façon évidente pour les réseaux de Petri p-temporels. C'est pourquoi dans le chapitre suivant nous avons repris l'analyse des réseaux t-temporels. L'utilisation de l'opérateur *max* nous permet d'exprimer algébriquement la disjonction d'un ensemble de contraintes binaires et cela nous permet de calculer, en restant sous une forme symbolique, les dates des franchissements et les durées de séjour des jetons dans les places. Nous avons pu également donner des conditions algébriques pour l'accessibilité des marquages à l'intérieur de scénarios spécifiés par des séquents caractéristiques. Toutefois ce travail n'a été possible qu'en se plaçant dans le cadre de la sémantique faible car la sémantique forte est contradictoire avec la linéarité de la logique linéaire. Malgré tout, nous avons montré à la fin du chapitre qu'il était possible dans certains cas de montrer que la sémantique forte allait faire qu'un scénario donné pouvait invalider un autre scénario, les deux scénarios ayant été auparavant calculés dans le cadre de la sémantique faible. C'est une piste pour la vérification du bon fonctionnement de certains chiens de garde dans les systèmes temps réel distribués.

Tous les problèmes sont loin d'avoir été résolus. Tout d'abord nous n'avons considéré que quatre extensions temporelles des réseaux de Petri. Il faudrait compléter ce tour d'horizon et d'abord prendre en considération les réseaux de Petri à arcs temporels [60] qui associent une durée de séjour minimale et maximale aux jetons dans les places, comme les réseaux p-temporels, mais avec des contraintes différentes pour chaque transition de sortie différente d'une place donnée. Cela ne devrait *a priori* pas poser de problème car un ensemble conjonctif de contraintes binaires devrait être obtenu.

Par contre, les réseaux de Petri à flux temporels (RdPFT [56]) risquent de poser des problèmes car les neuf règles de synchronisation pour les rendez-vous temporels impliquent l'ensemble des contraintes à l'entrée d'une transition et elles vont donc sortir du cadre des contraintes binaires n'impliquant que deux événements.

Il est également clair qu'il nous faudra revenir sur les réseaux de Petri t-temporels. En effet, ces derniers permettent une représentation intuitive des chiens de garde et les résultats présentés dans notre dernier chapitre pourront difficilement être utilisés tels

quels pour des systèmes complexes. Si le calcul des dates est dans le cadre de l'algèbre « (max,+) » pour laquelle de nombreux résultats mathématiques existent, le calcul des durées de séjour introduit la négation « - » et pour les dates de consommation des marquages potentiels nous avons utilisé l'opérateur « min ». Faire du calcul symbolique avec les opérateurs « max », « min », « + » et « - » simultanément n'est pas simple. De plus il serait nécessaire d'utiliser la sémantique forte et non la sémantique faible.

Plusieurs pistes sont possibles. Il y a d'abord celle présentée par J. Delatour [9] qui consiste à s'appuyer sur un chemin du graphe des classes pour construire un arbre de preuve et mettre en évidence les ensembles de contraintes temporelles à vérifier pour ce chemin. Cela permet d'analyser un scénario que nous savons cohérent avec la sémantique forte. D'autres associations entre graphes de classes et arbres de preuve sont sans doute possibles.

Pour essayer de contourner la difficulté du calcul symbolique, il est possible d'envisager de combiner l'approche utilisée pour les réseaux p-temporels avec l'approche présentée dans le chapitre 5. En effet, si nous ne considérons que les bornes minimales, nous restons dans le cadre des ensembles conjonctifs de contraintes binaires. Le calcul symbolique n'interviendrait alors que pour spécifier la borne maximale des domaines.

Une autre voie serait de reposer le problème sous la forme d'un problème de modélisation et de voir s'il n'est pas possible de décrire les chiens de garde par des réseaux à arcs temporels. Nous pourrions alors reprendre la démarche présentée à la fin du chapitre 5 concernant l'analyse des conditions pour qu'un scénario invalide un autre scénario. Si dans le cadre des réseaux à arcs temporels les ensembles de contraintes sont bien conjonctifs et binaires, l'analyse pourra à nouveau s'appuyer sur les outils classiques de la programmation linéaire et de la propagation de contraintes.

En ce qui concerne les réseaux de Petri p-temporels, la situation nous semble plus mûre et le travail futur doit plutôt être un travail de mise en œuvre. Il consistera, dans le cadre de l'environnement TINA, à programmer la construction des arbres de preuve canoniques et l'obtention des graphes de contraintes temporelles associés. Un problème reste néanmoins ouvert, celui de l'obtention de tous les ensembles de relations de précédence associés à un scénario sans jamais construire deux fois le même ensemble. Derrière ce problème se trouve l'exploitation optimale des conflits de transitions et des conflits jetons. La réalisation d'un premier outils sous TINA devrait nous aider à mettre en évidence les situations problématiques en traitant un certain nombre d'exemples.

Annexe A

Règles de la logique linéaire du fragment MILL

Soit A un atome, F, G et H des formules, Γ et Δ des blocs de formules (connectés par « , »).

$$\begin{array}{lcl} \langle F \rangle & ::= & A_1 \mid A_2 \mid \ldots \mid A_N \mid 1 \mid \langle F \rangle \otimes \langle F \rangle \mid \langle F \rangle \multimap \langle F \rangle \\ \langle \Gamma \rangle & ::= & . \mid \langle \Gamma \rangle, \langle F \rangle \\ \langle \mathcal{S} \rangle & ::= & \langle \Gamma \rangle \vdash \langle \Gamma \rangle \end{array}$$

Groupe identité

$$\frac{}{A \vdash A}\,\mathrm{id} \qquad \frac{\Gamma \vdash F \quad F, \Delta \vdash H}{\Gamma, \Delta \vdash H}\,\mathrm{cut}$$

Groupe structurel

$$\frac{\Gamma, F, G, \Delta \vdash H}{\Gamma, G, F, \Delta \vdash H}\,\mathrm{X}_L$$

Groupe logique

$$\frac{\Gamma, F, G \vdash H}{\Gamma, F \otimes G \vdash H} \otimes_L \qquad \frac{\Gamma \vdash F \quad \Delta \vdash G}{\Gamma, \Delta \vdash F \otimes G} \otimes_R$$

$$\frac{\Gamma \vdash F \quad \Delta, G \vdash H}{\Gamma, \Delta, F \multimap G \vdash H} \multimap_L$$

Annexe B

Exemple d'arbre canonique

B.1 Séquent initial

$$\underbrace{p_1 \otimes p_4 \otimes p_7}_{M_0}, \underbrace{t_1, t_2, t_3, t_4, t_5}_{l_0} \vdash \underbrace{p_1 \otimes p_1 \otimes p_4 \otimes p_4 \otimes p_7}_{M_n} \tag{B.1}$$

B.2 Arbre de preuve du séquent

Étant donné la taille de l'arbre global, nous allons décomposer cet arbre en plusieurs fragments. Nous utiliserons M_n pour remplacer le marquage final jusqu'à l'étape finale de la construction de l'arbre.

Fragment1

$$\dfrac{\dfrac{\dfrac{\dfrac{}{p_1 \vdash p_1}\,\mathrm{id} \quad \dfrac{}{p_7 \vdash p_7}\,\mathrm{id}}{p_1, p_7 \vdash p_1 \otimes p_7}\otimes_R \quad \dfrac{p_2, p_7, p_4, t_2, t_3, t_4, t_5 \vdash M_n}{p_2 \otimes p_7, p_4, t_2, t_3, t_4, t_5 \vdash M_n}\otimes_L}{p_1, p_4, p_7, t_1, t_2, t_3, t_4, t_5 \vdash M_n}\multimap_L}{p_1, p_4 \otimes p_7, t_1, t_2, t_3, t_4, t_5 \vdash M_n}\otimes_L}{p_1 \otimes p_4 \otimes p_7, t_1, t_2, t_3, t_4, t_5 \vdash M_n}\otimes_L$$

Fragment2

$$\dfrac{\dfrac{\dfrac{}{p_4 \vdash p_4}\,\mathrm{id} \quad \dfrac{}{p_7 \vdash p_7}\,\mathrm{id}}{p_4, p_7 \vdash p_4 \otimes p_7}\otimes_R \quad \dfrac{\dfrac{\dfrac{}{p_2 \vdash p_2}\,\mathrm{id} \quad p_3, p_5, p_7, t_4, t_5 \vdash M_n}{p_5, p_7, p_2, t_3, t_4, t_5 \vdash M_n}\multimap_L}{p_5 \otimes p_7, p_2, t_3, t_4, t_5 \vdash M_n}\otimes_L}{p_2, p_7, p_4, t_2, t_3, t_4, t_5 \vdash M_n}\multimap_L$$

Fragment3

$$\dfrac{\dfrac{}{p_5 \vdash p_5}\,\mathrm{id} \quad \dfrac{\dfrac{\dfrac{}{p_3 \vdash p_3}\,\mathrm{id} \quad \dfrac{}{p_6 \vdash p_6}\,\mathrm{id}}{p_3, p_6 \vdash p_3 \otimes p_6}\otimes_R \quad \dfrac{\dfrac{\dfrac{p_1, p_1, p_4, p_4, p_7 \vdash M_n}{p_1 \otimes p_1, p_4, p_4, p_7 \vdash M_n}\otimes_L}{p_1 \otimes p_1 \otimes p_4, p_4, p_7 \vdash M_n}\otimes_L}{p_1 \otimes p_1 \otimes p_4 \otimes p_4, p_7 \vdash M_n}\otimes_L}{p_6, p_3, p_7, t_5 \vdash M_n}\multimap_L}{p_3, p_5, p_7, t_4, t_5 \vdash M_n}\multimap_L$$

Fragment4

$$\dfrac{\dfrac{}{p_1 \vdash p_1}\,\mathrm{id} \quad \dfrac{\dfrac{}{p_1 \vdash p_1}\,\mathrm{id} \quad \dfrac{\dfrac{}{p_4 \vdash p_4}\,\mathrm{id} \quad \dfrac{\dfrac{}{p_4 \vdash p_4}\,\mathrm{id} \quad \dfrac{}{p_7 \vdash p_7}\,\mathrm{id}}{p_4, p_7 \vdash p_4 \otimes p_7}\otimes_R}{p_4, p_4, p_7 \vdash p_4 \otimes p_4 \otimes p_7}\otimes_R}{p_1, p_4, p_4, p_7 \vdash p_1 \otimes p_4 \otimes p_4 \otimes p_7}\otimes_R}{p_1, p_1, p_4, p_4, p_7 \vdash \underbrace{p_1 \otimes p_1 \otimes p_4 \otimes p_4 \otimes p_7}_{M_n}}\otimes_R$$

Annexe C

Exemple du calcul des dates symboliques

C.1 Arbre de preuve canonique

$$\dfrac{\dfrac{\dfrac{}{E \vdash E}\,\mathrm{id} \quad \dfrac{}{D \vdash D}\,\mathrm{id}}{E, D \vdash E \otimes D}\otimes_R \quad \dfrac{}{F \vdash F}\,\mathrm{id}}{E, D, t_4 \vdash F}\multimap_L(t_4)$$

$$\dfrac{\dfrac{}{A \vdash A}\,\mathrm{id} \quad \dfrac{\dfrac{}{B \vdash B}\,\mathrm{id} \quad \dfrac{\dfrac{}{C \vdash C}\,\mathrm{id} \quad E, D, t_4 \vdash F}{D, C, t_3, t_4 \vdash F}\multimap_L(t_3)}{B, C, t_2, t_3, t_4 \vdash F}\multimap_L(t_2)}{B \otimes C, t_2, t_3, t_4 \vdash F}\otimes_L}{A, t_1, t_2, t_3, t_4 \vdash F}\multimap_L(t_1)$$

C.2 Arbre de preuve canonique avec dates symboliques

Nous décomposons l'arbre de preuve en plusieurs fragments : un fragment pour chaque franchissement de transition de la liste l_0. Le séquent conclusion de départ est $A, t_1, t_2, t_3, t_4 \vdash F$.

Fragment1

$$\dfrac{\dfrac{}{A(D_{pA}, .) \vdash A(., D_{cA})}\,\mathrm{id} \quad \dfrac{B(D_{pB}, .), C(D_{pC}, .), t_2, t_3, t_4 \vdash F}{B(D_{pB}, .) \otimes C(D_{pC}, .), t_2, t_3, t_4 \vdash F}\otimes_L}{A(D_{pA}, .), t_1, t_2, t_3, t_4 \vdash F}\multimap_L(t_1)$$

Fragment2

$$\dfrac{\dfrac{}{B(D_{pB},.) \vdash B(.,D_{cB})}\,\mathrm{id} \quad D(D_{pD},.), C(D_{pC},.), t_3, t_4 \vdash F}{B(D_{pB},.), C(D_{pC},.), t_2, t_3, t_4 \vdash F} \multimap_L(t_2)$$

Fragment3

$$\dfrac{\dfrac{}{C(D_{pC},.) \vdash C(.,D_{cC})}\,\mathrm{id} \quad E(D_{pE},.), D(D_{pD},.), t_4 \vdash F}{D(D_{pD},.), C(D_{pC},.), t_3, t_4 \vdash F} \multimap_L(t_3)$$

Fragment4

$$\dfrac{\dfrac{\dfrac{}{E(D_{pE},.) \vdash E(.,D_{cE})}\,\mathrm{id} \quad \dfrac{}{D(D_{pD},.) \vdash D(.,D_{cD})}\,\mathrm{id}}{E(D_{pE},.), D(D_{pD},.) \vdash E \otimes D} \otimes_R \quad \dfrac{}{F(D_{pF},.) \vdash F(.,D_{cF})}\,\mathrm{id}}{E(D_{pE},.), D(D_{pD},.), t_4 \vdash F} \multimap_L(t_4)$$

C.3 Dates et durées

Atomes	Dates de production
A	$D_{pA} = D_A$
B	$D_{pB} = D_A + d_1$
C	$D_{pC} = D_A + d_1$
D	$D_{pD} = D_A + d_1 + d_2$
E	$D_{pE} = D_A + d_1 + d_3$
F	$D_{pF} = D_A + d_1 + max(d_2, d_3) + d_4$

Atomes	Dates de consommation
A	$D_{cA} = D_A + d_1$
B	$D_{cB} = D_A + d_1 + d_2$
C	$D_{cC} = D_A + d_1 + d_3$
D	$D_{cD} = D_A + d_1 + max(d_2, d_3) + d_4$
E	$D_{cE} = D_A + d_1 + max(d_2, d_3) + d_4$
F	$D_{cF} = \ inconnue$

Atomes	Durées de séjour
A	d_1
B	d_2
C	d_3
D	$max(d_2, d_3) + d_4 - d_2$
E	$max(d_2, d_3) + d_4 - d_3$
F	$inconnue$

Bibliographie

[1] R. Alur and D. L. Dill. Automata for modeling real-time systems. In *ICALP'90*, Lecture Notes in Computer Science, pages 323–335, Warwick, England, 1990. Springer-Verlag.

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

[3] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using Time Petri nets. *IEEE Transactions on Software Engineering*, 17 :259–273, 1992.

[4] E. Best and R. Devillers. Sequential and concurrent behaviour in Petri nets theory. *Theoretical Computer Science*, 50 :87–136, 1987.

[5] P. Bonhomme. *Réseaux de Petri p-temporels : contributions à la commande robuste.* PhD thesis, LAMII/CESALP/ESIA, Université de Savoie, Annecy, France, Juillet 2001.

[6] M. Boyer. *Contribution à la modélisation des systèmes à temps contraint et application au multimédia.* PhD thesis, Université Paul Sabatier, Toulouse, France, Juillet 2001.

[7] C. Brown. Relating Petri nets to formulae of linear logic. LFCS series, Report ECS-LFCS-89-87, Laboratory for Foundations of Computer Science, University of Edinburgh, UK, 1989.

[8] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49 :61–95, 1991.

[9] J. Delatour. *Contribution à la spécification des systèmes temps réels : L'approche UML/PNO.* PhD thesis, Université Paul Sabatier, Toulouse, France, Septembre 2003.

[10] H. Demmou, S. Khalfaoui, N. Rivière, E. Guilhem, and R. Valette. Extracting critical scenarios from a petri net model using linear logic. In *Journal Européen des Systèmes Automatisés, JESA'02*, 2002.

[11] D. Dubois and H. Prade. Processing fuzzy temporal knowledge. *IEEE transactions on Systems, Man and Cybernetics*, 19(4) :729–744, 1989.

[12] S. E. Elmaghraby. *Activity networks : project planning and control by network models.* John Wiley & sons, New York, 1977.

[13] E. A. Emerson, E. Clarke, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2) :244–263, 1986.

[14] E. A. Emerson and C. L. Lei. Modalities for model checking : branching time strikes back. *Science of Computer Programming*, 8 :275–306, 1987.

[15] U. Engberg and G. Winskel. Petri nets as models of linear logic. In *Proceedings of the 15th Colloquium on Trees in Algebra and Programming*, volume 431 of *Lecture Notes in Computer Science*, pages 147–161, Copenhagen, Denmark, 1990. Berlin, Germany : Springer-Verlag, 1990.

[16] J. Engelfriet. Branching processes of Petri nets. *Acta Informatica*, 28 :575–591, 1991.

[17] J. Esparza, S. Romer, and W. Vogler. An improvement of McMillan's unfolding algorithm. In *TACAS'96*, volume 1055 of *Lecture Notes in Computer Science*, pages 87–106, 1996.

[18] J. Fanchon, N. Rivière, B. Pradin-Chezalviel, and R. Valette. Preuves de logique linéaire et process de réseaux de Petri. In *Modélisation des systèmes réactifs, MSR'03*, Metz, France, 6-8 octobre 2003.

[19] V. Gehlot. *A proof theoretic approach to semantics of concurrency.* PhD thesis, University of Pennsylvania, 1992.

[20] J. Y. Girard. Linear logic. *Theoretical Computer Science*, 50 :1–102, 1987.

[21] F. Girault. *Formalisation en logique linéaire du fonctionnement des réseaux de Petri.* PhD thesis, Université Paul Sabatier, Toulouse, France, Décembre 1997.

[22] P. Godefroid. *Partial-order methods for the verification of concurrent systems, an approach to the state-explosion problem.* PhD thesis, Université de Liège, 1994.

[23] U. Goltz and W. Reisig. The non-sequential behaviour of Petri nets. *Information and Computation*, 57 :125–147, 1983.

[24] C. A. Gunter and V. Gehlot. Nets as tensor theories. In *10^{th} International Conference on Application and Theory of Petri nets*, Lecture Notes in Computer Science, pages 174–191, Bonn, Germany, 1989. Springer-Verlag.

[25] P. W. Hoodgers, H. C. M. Kleijn, and P. S. Thiagarajan. Event structure semantics for general Petri nets. *Theoretical Computer Science*, 153 :129–170, 1996.

[26] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[27] M.-J. Huguet, P. Lopez, and T. Vidal. Dynamic task sequencing in temporal problems with uncertainty. In *AIPS'02*, pages 41–48, 2002.

[28] M. Jantzen and R. Valk. Formal properties of place/transition nets. In Brauer, W., editor, *Lecture Notes in Computer Science : Net Theory and Applications, Proc. of the Advanced Course on General Net Theory of Processes and Systems, Hamburg, 1979*, volume 84, pages 165–212, Berlin, Heidelberg, New York, 1980. Springer-Verlag.

[29] W. Khansa. *Réseaux de Petri p-temporels : contributions à l'étude des systèmes à événements discrets.* PhD thesis, Université de Savoie, Annecy, France, 1997.

[30] W. Khansa, P. Aygalinc, and J. P. Denat. Structural analysis of p-time Petri nets. In *CESA'96 IMACS*, pages 127–136, Lille, France, 1996.

[31] W. Khansa, J. P. Denat, and S. Collart-Dutilleul. P-time Petri nets for manufacturing systems. In *WODES'96*, pages 94–102, 1996.

[32] L. A. Künzle. *Raisonnement temporel basé sur les réseaux de Petri pour des systèmes manipulant des ressources.* PhD thesis, Université Paul Sabatier, Toulouse, France, Septembre 1997.

[33] C. Mancel, P. Lopez, N. Rivière, and R. Valette. Relationships between Petri nets and constraint graphs : application to manufacturing. In *15th IFAC world congress*, page n. 634, Barcelona, Spain, 21-26 July 2002.

[34] A. Mazurkiewicz. Trace theory. In *Petri Nets, Applications and Relationship to other Models of Concurrency*, number 255 in Lecture Notes in Computer Science, pages 279–324. Springer-Verlag, 1987.

[35] K. L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In *4th International conference on Computer Aided Verification*, volume 663 of *Lecture Notes in Computer Science*, pages 164–177, 1992.

[36] M. Menasche and B. Berthomieu. Time Petri nets for analysing and verifying time dependent protocols. *Third international workshop on protocol specification, testing and verification*, June 1983.

[37] P. Merlin and D. J. Farber. Recoverability of communication protocols : implementation of a theoretical study. *IEEE Trans. on communications*, COM-24(9 September) :1036–1043, 1976.

[38] J. Meseguer, U. Montanari, and V. Sassone. Process versus unfolding semantics for place/transition Petri nets. *Theoretical Computer Science*, 153 :171–210, 1996.

[39] U. Montanari. Network of constraints : fundamental properties and applications to picture processing. *Inf. Sci.*, 7 :95–132, 1974.

[40] M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains. *Theoretical Computer Science*, 13 :85–108, 1981.

[41] J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Englewood Cliffs, New Jersey : Prentice Hall, Inc., 1981.

[42] C. A. Petri. *Kommunikation mit Automaten.* Bonn : Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

[43] C. A. Petri. Nets, time and space. *Theoretical Computer Science*, 153(1–2) :3–48, 1996.

[44] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13 :45–60, 1981.

[45] B. Pradin-Chézalviel and R. Valette. Accessibilité de marquage et logique linéaire dans un réseau de Petri t-temporel. In *Journées Formalisation des Activités Concurrentes, FAC'2000*, pages 123–134, Toulouse, France, 18-19 mai 2000.

[46] B. Pradin-Chézalviel, R. Valette, and L. A. Künzle. Scenario duration characterization of t-timed Petri nets using linear logic. In *IEEE PNPM'99*, pages 208–217, Zaragoza, Spain, September 6-10 1999.

[47] C. Ramchandani. *Analysis of asynchronous concurrent systems by timed Petri nets.* PhD thesis, MIT, 1974.

[48] W. Reisig and G. Rozenberg. *Lectures on Petri Nets I : Basic Models*, volume 1491 of *Lecture Notes in Computer Science.* Springer-Verlag, 1998.

[49] P. O. Ribet, F. Vernadat, and B. Berthomieu. On combining the persistent sets method with the covering steps graph method. In *22nd IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE'2002)*, Lecture Notes in Computer Science, pages 344–359, Houston, USA, 2002. Springer-Verlag.

[50] N. Rivière, B. Pradin-Chézalviel, and R. Valette. Reachability and temporal conflicts in t-time Petri nets. In *IEEE PNPM'01*, pages 229–238, Aachen, Germany, September 11-14 2001.

[51] P. N. M. Sampaio. *Conception formelle de documents multimedia interactifs : une approche s'appuyant sur RT-Lotos.* PhD thesis, Université Paul Sabatier, Toulouse, France, 2003.

[52] P. N. M. Sampaio and J.-P. Courtiat. Scheduling and presenting interactive multimedia documents. In *2001 IEEE International Conference on Multimedia (ICME'2001)*, pages 1224–1227, Tokyo, Japan, August 22-25 2001.

[53] P. N. M. Sampaio and J.-P. Courtiat. A formal approach for the presentation of interactive multimedia documents. In *8th ACM International Conference on Multimedia*, pages 435–438, Los Angeles, USA, October 30 - November 4 2000.

[54] E. Schwalb and R. Dechter. Processing disjunctions in temporal constraint networks. *Artificial Intelligence*, 93 :29–61, 1997.

[55] E. Schwalb and L. Vila. Temporal constraints : a survey. *Constraint : an International journal (Kluwer Academic)*, 2 :129–149, 1998.

[56] P. Senac, M. Diaz, A. Léger, and P. de Saqui-Sannes. Modeling logical and temporal synchronization in hypermedia systems. *IEEE Journal on selected areas in Communications*, 14(1) :84–103, January 1996.

[57] J. Sifakis. Use of petri nets for performance evaluation. *Measuring, modelling and evaluating computer systems*, pages 75–93, 1977.

[58] F. Vernadat, P. Azéma, and F. Michel. Covering step graph. In *Application and theory of Petri nets 1996*, Lecture Notes in Computer Science, pages 516–535. Springer-Verlag, 1996.

[59] T. Vidal. Problèmes temporels avec incertitudes : approches polynômiales par décomposition et par filtrage. In *JNPC'2001*, pages 297–308, 2001.

[60] B. Walter. Timed net for modeling and analysing protocols with time. In North Holland, editor, *IFIP Conference on Protocol specification testing and verification*, 1983.

[61] J. Winkowski. Processes of timed Petri nets. *Theoretical Computer Science*, 243 :1–34, 2000.

[62] P. Wolper and P. Godefroid. Partial-order methods for temporal verification. In *Concur'93*, Lecture Notes in Computer Science, pages 344–359. Springer-Verlag, 1993.

Modélisation et analyse temporelle par réseaux de Petri et logique linéaire

L'objectif de cette thèse est de contribuer à l'élaboration de méthodes d'aide à la conception de systèmes coopératifs en prenant en compte les contraintes temporelles de manière quantitative. L'approche développée est fondée sur les réseaux de Petri, la logique linéaire et les graphes de contraintes temporelles. C'est une approche orientée « événements » et non orientée « états » comme c'est souvent le cas dans les approches fondées sur les réseaux de Petri. Elle est décomposée en deux étapes : une étape d'analyse « qualitative » et une étape d'analyse « quantitative ».

La première consiste à obtenir les relations de causalité entre les événements appartenant à un scénario donné. L'équivalence entre un arbre de preuve en logique linéaire et le processus fini obtenu par dépliage d'un réseau de Petri à partir du même marquage initial montre que ces relations sont des relations de précédence. L'introduction de la notion de séquent caractéristique permet de mettre en œuvre une approche compositionnelle des processus à partir des règles du calcul des séquents.

La deuxième étape consiste à passer du graphe décrivant les relations de précédence à un graphe de contraintes temporelles exprimant de façon linéaire l'ensemble des contraintes temporelles quantitatives que doivent vérifier les dates des franchissements des transitions dans un scénario. Il devient ainsi possible d'exploiter tous les résultats des techniques classiques d'analyse et de propagation de contraintes. Cette démarche est complètement cohérente avec les réseaux de Petri p-temporels mais difficilement compatible avec les t-temporels car ils engendrent des ensembles de contraintes qui sont plus complexes. Nous avons illustré cette démarche par un problème simple d'ordonnancement de documents multimédias.

Nous avons par la suite montré comment, pour les réseaux de Petri t-temporels, nous pouvions calculer les dates de franchissements et les durées de séjour des jetons dans les places en restant sous une forme symbolique dans le cadre de la sémantique faible.

Mots clefs : Réseaux de Petri, logique linéaire, ordres partiels, processus de réseaux de Petri, graphes de contraintes temporelles.

Temporal modelling and analysis by Petri nets and linear logic

The aim of this thesis is to contribute to the elaboration of design assistance methods of cooperative systems while taking into account temporal constraints in a quantitative way. The developed approach is based on Petri nets, linear logic and temporal constraints networks. This is an "event" oriented approach and not a "state" oriented one as it is often the case in the approaches based on Petri nets. It is split in two steps : a step of "qualitative" analysis and a step of "quantitative" one.

The first consists in obtaining the causality relations between the events belonging to a given scenario. The equivalence between a proof tree in linear logic and the finite process obtained by the unfolding of a Petri net from the same initial marking shows that these relations are precedence relations. The introduction of the concept of characteristic sequent makes it possible to implement a compositional approach of the processes from the rules of the linear logic sequent calculus.

The second step consists in changing the graph describing the precedence relations into a temporal constraints graph expressing in a linear way the set of the quantitative temporal constraints which have to be verified by the dates of the firing transitions in a scenario. Thus, it become possible to exploit all the results of traditional techniques of analysis and constraints propagation. This step is completely consistent with p-time Petri nets but not easily compatible with the t-timed ones because they generate sets of constraints which are more complex. This approach is illustrated by a simple scheduling problem of multimedia documents.

We showed thereafter how, for the t-timed Petri nets, we could process the firing dates and the sojourn durations of the tokens in the places of a net while remaining in a symbolic form within the framework of the weak semantics.

Keywords : Petri nets, linear logic, partial orders, Petri nets processes, temporal constraints graphs.