

IRCCyN - Institut de Recherche en Communications et Cybernétique de Nantes

Master Recherche « Automatique et Systèmes de Productions »

Rapport de séminaire bibliographique

Raisonnement temporel en sémantique forte à partir de réseaux de Petri temporels exprimés en Logique Linéaire.

Louis-Marie Traonouez

Encadré par : David Delfieu

Table des matières

Introduction	5
1 La logique linéaire	9
1.1 Limites de la logique classique	9
1.2 Connecteurs de la logique linéaire	10
1.3 Le calcul des séquents	11
1.3.1 Introduction	11
1.3.2 Notion de séquent	12
1.3.3 Notion de règle	12
1.3.4 Preuve d'un séquent	13
1.3.5 Règle de coupure	14
1.4 Fragment MILL de la logique linéaire	14
1.5 Manipulation de ressources en logique linéaire	15
1.6 Conclusion	16
2 Formalisation des réseaux de Petri en logique linéaire	17
2.1 Intérêt de l'approche par la logique linéaire	17
2.2 Définitions d'un réseau de Petri	18
2.2.1 Structure et marquage	18
2.2.2 Représentations d'un réseau de Petri	18
2.2.3 Sensibilisations et franchissements de transitions	20
2.2.4 Conflits et parallélisme	21
2.3 Traduction des réseaux de Petri en logique linéaire	22
2.3.1 Traduction de la structure du réseau	22
2.3.2 Traduction du fonctionnement du réseau	23
2.3.3 Preuve d'un séquent	24
2.4 Conclusion	25
3 Analyses temporelles des réseaux de Petri T-temporels	27
3.1 Réseaux de Petri T-temporels	27
3.2 Algorithme de VALETTE avec estampilles temporelles	27
3.2.1 Estampilles temporelles	28
3.2.2 Calcul des estampilles temporelles	28
3.2.3 Algorithme	28
3.2.4 Application sur un exemple	29
3.3 Retour sur la notion de scénario	30

3.3.1	Conflits de transitions et conflits de jetons	31
3.3.2	Scénarios complètement spécifiés	31
3.3.3	Parallélisme et logique linéaire	32
3.4	Conclusion	32
4	Prise en compte de la sémantique forte	33
4.1	Sémantique faible et sémantique forte	33
4.1.1	Différences entre les sémantiques	33
4.1.2	Sémantique de tir et logique linéaire	34
4.2	Algorithme de preuve en sémantique forte	34
4.2.1	Repérage des conflits de transitions	34
4.2.2	Modification de l'ordre de la preuve	35
4.2.3	Nouveau calcul des dates	36
4.2.4	Algorithme	37
4.3	Application sur un exemple	37
4.4	Conclusion	39
	Conclusion	39
A	Règles du calcul des séquents en logique classique	43
B	Règles du calcul des séquents en logique linéaire	45

Table des figures

2.1	Représentation graphique d'un réseau de Petri marqué.	19
2.2	Franchissement d'une transition dans un réseau de Petri.	20
2.3	Arbre de preuve canonique d'un séquent	25
3.1	Réseau de Petri avec parallélisme	29
3.2	Conflits de jetons et de transitions	31
4.1	Situations de conflits dans un réseau T-temporel	34
4.2	Conflit indirect entre deux transitions	35
4.3	Exemple de réseau de Petri avec conflit	36
4.4	Exemple de réseau de Petri avec conflit	38

Introduction

La logique linéaire a été introduite en 1987 par J-Y. GIRARD. Contrairement à la logique classique, où toutes les vérités sont éternelles, c'est une logique non monotone ce qui la rend particulièrement adaptée pour raisonner sur des ressources. A partir de cette propriété, plusieurs travaux, et en particulier ceux menés au Laboratoire d'Analyses et d'Architecture des Systèmes du CNRS à Toulouse, ont montré que l'on pouvait représenter et manipuler les réseaux de Petri à l'aide de la logique linéaire.

Cette représentation des réseaux de Petri a alors été utilisée pour effectuer des raisonnements temporels sur des réseaux de Petri T-temporels. Cependant, dans les travaux du LAAS les analyses ne sont effectuées que dans le cadre de la sémantique faible qui ne traite pas l'urgence des événements. Cette urgence, prise en compte dans la sémantique forte, est un aspect primordiale dans les systèmes temps réels. Des travaux antérieurs, réalisés à l'IRCCyN dans le cadre de DEA, ont porté sur la réintroduction de la sémantique forte dans l'analyse des réseaux de Petri par la logique linéaire.

Dans ce rapport nous présenterons les principes de bases de la logique linéaire, puis nous étudierons les méthodes proposées pour traduire et analyser les réseaux de Petri à l'aide de la logique linéaire. Nous verrons comment utiliser la logique linéaire pour effectuer des analyses temporelles des réseaux de Petri T-temporels. Enfin, dans la dernière partie nous présenterons les méthodes d'analyses en sémantique forte.

Chapitre 1

La logique linéaire

Théorie récente [1] et très novatrice, elle a suscité un fort engouement dans le domaine de la logique et de l'informatique théorique. Nous allons dans ce chapitre présenter cette logique qui introduit de nouveaux connecteurs par rapport à la logique classique. Nous présenterons également le calcul des séquents qui est le formalisme syntaxique utilisé pour exprimer la logique linéaire.

1.1 Limites de la logique classique

Des vérités éternelles

Pour raisonner sur des systèmes dynamiques, la logique classique est inadéquate, principalement à cause de sa monotonie. En effet la validité d'une proposition ne peut pas varier, c'est une vérité éternelle. Ceci est illustré par le fonctionnement du connecteur *implique* : lorsque l'on écrit *A implique B*, cela signifie que si l'on a *A*, alors on a *B* mais *A* reste vraie.

Pour combler cette limitation, des logiques temporelles peuvent être utilisées. Elles sont basées sur la logique modale. En logique modale les vérités sont locales, limitées à un monde. Cependant pour passer d'un monde à l'autre il est nécessaire de s'appuyer sur un graphe d'état.

Impossibilité de raisonner sur des ressources

Cette monotonie est également préjudiciable pour représenter la notion de ressource, qui est essentielle dans les systèmes dynamiques et dans les réseaux de Petri. Si l'on essaye de raisonner sur des ressources avec la logique classique, on aboutit à des absurdités. Prenons l'exemple des cigarettes présenté par GIRARD [1] et repris par GIRAULT [2] :

On a la propriété : (P1) 3€
On a les règles : (R1) 3€ ⇒ un paquet de Gauloises
(R2) 3€ ⇒ un paquet de Gitanes

Si on applique la règle (R1) sur (P1) on obtient *un paquet de Gauloises*. Cependant à cause de la monotonie de la logique classique (P1) reste vraie, et on a donc :

un paquet de Gauloises et 3€.

Puisque $(P1)$ est vraie, on peut ensuite appliquer la règle $(R2)$ sur $(P1)$, ce qui nous donne :

un paquet de Gitanes et 3€,

et on se retrouve au final avec un paquet de Gauloises, un paquet de Gitanes et 3€. Ce raisonnement est évident absurde au sens des ressources.

Le problème de l'idempotence

Une des propriétés caractéristiques de la logique classique est l'idempotence des connecteurs ET et OU qui spécifie que :

$$A \wedge A \Leftrightarrow A \quad \text{et} \quad A \vee A \Leftrightarrow A$$

Cette propriété entraîne également en terme de ressources des absurdités :

Si j'ai 1€ alors j'ai 1€ et 1€.

Ceci montre que l'on n'est pas capable avec la logique classique de compter le nombre de ressources.

On s'aperçoit donc que la logique classique n'est pas adaptée pour représenter des systèmes dynamiques. Nous allons voir par contre que la logique linéaire manipule très naturellement la notion de ressource.

1.2 Connecteurs de la logique linéaire

Pour créer la logique linéaire GIRARD [1] a supprimé l'idempotence des connecteurs ET et OU. Cela l'a amené à créer un nouveau système de connecteurs. Parmi ces connecteurs linéaires on trouve une négation linéaire et huit autres connecteurs répartis en trois familles : les multiplicatifs, les additifs et les exponentiels. La négation linéaire et les exponentiels sont des connecteurs unaires.

Connecteurs multiplicatifs

\otimes	<i>fois</i>	conjonction multiplicative
\wp	<i>par</i>	disjonction multiplicative
\multimap	<i>entraîne</i>	implication linéaire

Connecteurs additifs

$\&$	<i>avec</i>	conjonction additive
\oplus	<i>plus</i>	disjonction additive
\rightsquigarrow	<i>(aucun nom défini)</i>	implication additive ¹

Connecteurs exponentiels

$!$	<i>bien sûr</i>
$?$	<i>pourquoi pas</i>

¹Ce connecteur n'est pas mentionné par GIRARD mais est théoriquement concevable. C'est GIRAULT [2] qui l'introduit de la sorte.

Voici quelques propriétés des connecteurs de la logique linéaire.

La négation linéaire $()^\perp$, appelée *nil*, établit une dualité entre les connecteurs au sein de chaque famille. Par exemple :

$$(A \otimes B)^\perp \text{ équivaut à } A^\perp \wp B^\perp$$

Elle constitue en ce sens la clef de voûte du système. Elle possède tout comme son homologue de la logique classique la propriété d'involution :

$$\text{nil}(\text{nil}(A)) \text{ équivaut à } A.$$

Entre les connecteurs il va exister plusieurs relations de distributivité ou de semi-distributivité.

Les connecteurs *fois* et *par* sont les équivalents linéaires des connecteurs ET et OU auxquels on a enlevé la propriété d'idempotence. Cette propriété d'idempotence est toutefois rétablie pour les connecteurs additifs et exponentiels.

Les deux implications peuvent être définies, tout comme leur homologue classique, à l'aide de la négation linéaire et d'une disjonction :

$$\begin{aligned} A \multimap B &\text{ équivaut à } A^\perp \wp B \\ A \multimap B &\text{ équivaut à } A^\perp \oplus B \end{aligned}$$

Les connecteurs de la logique linéaire possèdent des éléments neutres et des éléments absorbants qui sont présentés dans le tableau 1.1

Connecteur	Élément neutre	Élément absorbant
\otimes	1 (<i>un</i>)	0 (<i>zéro</i>)
\wp	\perp (<i>faux</i>)	\top (<i>vrai</i>)
$\&$	\top (<i>vrai</i>)	
\oplus	0 (<i>zéro</i>)	

TAB. 1.1 – Élément neutres et éléments absorbants des connecteurs de logique linéaire

Pour décrire plus précisément le fonctionnement de la logique linéaire nous devons tout d'abord présenté le calcul des séquents qui est le formalisme syntaxique utilisé par GIRARD pour définir cette logique.

1.3 Le calcul des séquents

1.3.1 Introduction

La logique propositionnelle classique est habituellement définie en utilisant la méthode axiomatique. Cette méthode définit la syntaxe de la logique par un ensemble de propositions posées sans condition, *les axiomes*.

Plusieurs ensembles d'axiomes sont envisageables, on peut par exemple utiliser :

1. $A \Rightarrow (B \Rightarrow A)$
2. $((A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C)))$
3. $((\neg B \Rightarrow \neg A) \Rightarrow ((\neg B \Rightarrow A) \Rightarrow B))$

Des règles de déduction telles que le *modus ponens* ou la *substitution* permettent de prouver des théorèmes à partir des axiomes ou à partir d'autres théorèmes. La règle du *modus ponens* énonce par exemple que *si A et $A \Rightarrow B$ sont des théorèmes alors B est un théorème.*

Cette méthode a l'inconvénient que les démonstrations reposent essentiellement sur l'intuition. Introduit en 1934 par le logicien Gentzen, le calcul des séquents est un autre formalisme syntaxique qui ne comporte pas d'axiomes mais uniquement des règles de réduction. C'est ce formalisme que GIRARD a utilisé pour définir la logique linéaire.

1.3.2 Notion de séquent

Un séquent est une *déduction* qui exprime que d'un ensemble de prémisses on peut déduire un ensemble de conclusion. Toutes les prémisses et toutes les conclusions sont explicitées. Sa forme générale est la suivante :

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

- A, B, E, F sont des variables de formules propositionnelles. Ce sont donc des *méta-variables*, elles peuvent se substituer avec toute formule propositionnelle.
- le symbole \vdash , appelé *tournequet*, est un *méta-connecteur* qui sépare le membre droit contenant les prémisses, du membre gauche contenant les conclusions. Prémisses et conclusions sont des suites de méta-variables séparées par une virgule, elles peuvent donc éventuellement contenir plusieurs occurrences d'une même formule.
- la virgule qui sépare ces méta-variables est également un *méta-connecteur*. Elle a un sens différent selon qu'elle est placée à droite ou à gauche du *tournequet*.
- le membre gauche est formé de la conjonction des prémisses du séquent. La virgule a donc dans ce membre le sens du ET en logique classique (ou du *fois* en logique linéaire).
- le membre droit est formé de la disjonction des conclusions du séquent. La virgule a donc dans ce membre le sens du OU en logique classique (ou du *par* en logique linéaire).

On utilisera les notations suivantes :

- \nvdash pour signifier qu'un séquent n'est pas prouvable,
- $A \dashv\vdash B$ pour signifier que les deux séquents $A \vdash B$ et $B \vdash A$ sont prouvables,
- $A \dashv\vdash B$ pour signifier que seul le séquent $A \vdash B$ est prouvable,
- $A \not\vdash B$ pour signifier $A \nvdash B$ et $B \nvdash A$.

1.3.3 Notion de règle

Pour prouver la correction de la déduction exprimée par un séquent on applique les *règles* du calculs des séquents. Une *règle* exprime que de un ou de deux séquents prémisses on peut déduire un séquent conclusion. Puisqu'elle s'applique à des déductions (les séquents), une règle est une *méta-déduction*. Une règle a la forme générale suivante, avec éventuellement un seul séquent prémisses :

$$\frac{\text{séquent prémisses 1} \quad \text{séquent prémisses 2}}{\text{séquent conclusion}} \text{ nom de la règle}$$

Règles du calcul des séquents

Les règles du calcul des séquents sont divisées en trois groupes : *le groupe identité*, *le groupe structurel* et *le groupe logique*. Pour les décrire² on considère dans ce qui suit que F , G et H sont des formules propositionnelles alors que Γ et Δ sont des suites de formules propositionnelles qui constituent le contexte.

Le **groupe identité** contient deux règles : la *règle identité* et la *règle de coupure*. Ces deux règles sont communes à la logique classique et à la logique linéaire, elles expriment des principes logiques très généraux. La règle identité est la seule règle axiome c'est à dire qu'elle ne nécessite aucun séquent prémisses. La règle de coupure consiste à utiliser un lemme F pour prouver un séquent.

Règle identité

$$\frac{}{F \vdash F} \textit{id}$$

Règle de coupure

$$\frac{\Gamma \vdash F \quad \Delta, F \vdash H}{\Gamma, \Delta \vdash H} \textit{cut}$$

Le **groupe structurel** contient des règles qui opèrent sur la structure du séquent. En logique linéaire ce groupe ne contient que les règles d'échange qui rendent la virgule commutative dans le membre droit et dans le membre gauche. En logique classique, il contient aussi les règles de contraction et d'affaiblissement. Ces règles sont à l'origine de l'idempotence des connecteurs ET et OU, et leur suppression est à l'origine de la logique linéaire. En pratique, on considère les règles d'échanges comme implicites, c'est à dire que la virgule est commutative, ce qui permet d'alléger les preuves [2].

Le **groupe logique** contient les règles d'introduction des connecteurs et des constantes. Lu de haut en bas ces règles introduisent un connecteur dans le membre droit ou dans le membre gauche. De bas en haut elles servent à supprimer un connecteur. Voici par exemple les règles d'introduction des connecteurs *fois* et *entraîne* :

$$\frac{\Gamma, F, G \vdash H}{\Gamma, F \otimes G \vdash H} \otimes_L$$

$$\frac{\Gamma \vdash F \quad \Delta \vdash G}{\Gamma, \Delta \vdash F \otimes G} \otimes_R$$

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

$$\frac{\Gamma, F \vdash G}{\Gamma \vdash F \multimap G} \multimap_R$$

L'ensemble des règles du calcul des séquents classiques (LK) et du calcul des séquents linéaires (LL) pourront être trouvées en annexe A et B.

Hormis pour les règles d'identité et de coupure, tous les noms des règles possèdent un label, soit L soit R, qui indique le membre du séquent sur lequel elles agissent, soit le gauche soit le droit.

1.3.4 Preuve d'un séquent

La preuve d'un séquent est réalisée en appliquant les règles du bas vers le haut. L'application d'une règle est appelée **inférence**, elle permet de réduire le séquent conclusion en un ou deux séquents plus

²Nous décrivons ici les règles du calcul des séquents linéaires intuitionnistes, c'est à dire que les membres droits des séquents ne comportent qu'une seule formule.

simples à prouver. Une preuve possède ainsi une forme arborescente dont le séquent conclusion que l'on cherche à prouver est la racine. Pour que l'arborescence des inférences soit une **preuve**, il faut que toutes les feuilles commencent par des règles axiomes (qui ne possèdent pas de séquent prémisses).

Pour prouver un séquent, la stratégie est donc d'éliminer au fur et à mesure toutes les occurrences des connecteurs, jusqu'à ce qu'il ne reste que des séquents axiomes prouvable par la règle identité. Cependant cette opération n'est pas purement mécanique, l'ordre des inférences n'est pas arbitraire [2].

Voici en exemple la preuve du séquent de LK suivant³ : $A, A \Rightarrow B \vdash B \wedge A$.

$$\frac{\frac{\frac{\overline{A \vdash A} \text{ id} \quad \overline{B \vdash B} \text{ id}}{A, A \Rightarrow B \vdash B} \Rightarrow_R \quad \overline{A \vdash A} \text{ id}}{A, A, A \Rightarrow B \vdash B \wedge A} \wedge_R}{A, A \Rightarrow B \vdash B \wedge A} C_L$$

On s'aperçoit que la preuve de ce séquent fait intervenir la règle de contraction C_L qui n'est pas présente dans LL. Ceci nous montre la différence entre la logique linéaire et la logique classique. La prouvabilité de ce séquent est l'expression de la monotonie de la logique classique. La suppression de la règle de contraction (et de celle d'affaiblissement) ne permet plus de prouver un séquent analogue en logique linéaire ce qui la rend non monotone. En effet, le séquent de LL suivant : $A, A \multimap B \vdash B \otimes A$ n'est pas prouvable.

1.3.5 Règle de coupure

Cette règle est très particulière, c'est la seule règle de LK et de LL à introduire dans ses séquents prémisses une formule exogène au séquent conclusion. Cela consiste à utiliser un lemme intermédiaire, ce qui est une méthode très utilisée en mathématiques. Cependant, il existe dans LL et dans LK un théorème très important qui peut s'énoncer ainsi :

La règle de coupure est redondante.

Ce théorème signifie que tout séquent prouvable peut l'être sans l'utilisation de la règle de coupure. Il existe un algorithme de *réduction des coupures* qui permet d'éliminer les inférences de la règle de coupure dans une preuve en les faisant remonter vers les séquents axiomes.

1.4 Fragment MILL de la logique linéaire

Un constat s'impose : la logique linéaire propositionnelle est indécidable [2]. Cela signifie qu'il n'existe pas d'algorithme permettant de déterminer la prouvabilité d'un séquent quelconque de LL. Pour pouvoir étudier des problèmes de logique linéaire on se limite donc à certains fragments qui excluent certains connecteurs. Ces fragments sont toutefois définis de telle sorte qu'un séquent du fragment considéré, prouvable dans LL, le soit aussi dans ce fragment.

Dans notre application de la logique linéaire aux réseaux de Petri on se limitera au fragment MILL (Multiplicative Intuitionistic Linear Logic) qui est réduit aux deux seuls connecteurs multiplicatifs *fois* et *entraîne*. On considère de plus dans ce fragment que le membre droit du séquent ne contient qu'une seule formule. Ce fragment nous sera suffisant pour exprimer les réseaux de Petri et la prouvabilité d'un séquent devient dans ce fragment un problème NP-complet [2].

³Cet exemple est tiré de la thèse de F.GIRAULT [2], cependant nous ne faisons pas apparaître dans la preuve les règles d'échanges.

1.5 Manipulation de ressources en logique linéaire

Maintenant que nous avons défini les connecteurs de la logique linéaire et le calcul des séquents nous allons voir qu'elles interprétations donner aux séquents et aux connecteurs (en ce limitant au fragment MILL) pour raisonner sur des ressources. Nous nous apercevrons alors que la logique linéaire permet d'outrepasser les limites de la logique classique que nous avons exhibées au début de ce chapitre. Cette interprétation est un court extrait de celle réalisée par F.GIRAULT dans sa thèse [2].

Interprétation d'un séquent au sens des ressources

Un séquent représente une *action* sur des ressources. Dans le membre gauche on trouve les *consommés* de l'action et dans le membre droit le *produit* de l'action. Dans les deux membres, chaque formule de logique linéaire représente une *ressource* qui peut être, *composée* si elle fait intervenir des connecteurs, ou atomique sinon.

L'implication linéaire \multimap

L'implication linéaire *entraîne* possède le sens du tourniquet. Du point de vue des ressources, $A \multimap B$ est une *ressource d'action*. Elle exprime la possibilité de produire un exemplaire de la ressource B à partir d'un exemplaire de la ressource A. Cependant cette production ne peut être effectuée qu'une seule fois car lors de la production l'exemplaire de $A \multimap B$ est consommé, d'où le terme *ressource d'action*. Le séquent suivant :

$$A, A \multimap B \vdash B$$

exprime la transformation d'une ressource A en une ressource B. La prouvabilité de ce séquent et la non prouvabilité du suivant : $A, A \multimap B \vdash B \otimes A$ rend la logique linéaire non monotone, c'est à dire que la valeur de vérité d'une formule n'est pas fixe.

Illustrons ceci sur un achat dans une boulangerie. L'implication $1\text{€} \multimap 1\text{pain}$ signifie du point de vue du client qu'un pain est en vente au prix de 1€. Le client possède au départ la ressource 1€ et la ressource $1\text{€} \multimap 1\text{pain}$ qui lui indique une possibilité d'achat. L'achat d'un pain par le client est exprimé par le séquent :

$$1\text{€}, 1\text{€} \multimap 1\text{pain} \vdash 1\text{pain}$$

Au final le client ne possède plus que la ressource *pain*. La consommation de la ressource d'action indique qu'un pain en moins est en vente dans la boulangerie. Si il n'y a plus de pains, la ressource d'action n'est plus consommable.

La conjonction multiplicative \otimes

La conjonction multiplicative *fois* a le sens de la virgule dans le membre gauche d'un séquent ce qui s'exprime par $A, B \vdash A \otimes B$. Ainsi la règle d'introduction à gauche de ce connecteur consiste simplement à remplacer dans le membre droit une occurrence de la virgule par \otimes . Le connecteur *fois* est l'équivalent linéaire du connecteur classique ET. Cependant il ne possède pas la propriété d'idempotence, c'est à dire $A \otimes A \not\vdash A$.

En terme de ressource, il traduit un cumul conjoint de ressources. Le rejet de l'idempotence nous permet de compter les ressources. $A \otimes A$ signifie que l'on considère 2 exemplaires de la ressource A ce qui est différent de 1 exemplaire ou de 3. On utilise la notation raccourcie $A^{\otimes n}$ pour compter les n exemplaires d'une ressource A.

1.6 Conclusion

Le logique linéaire peut donc être interprétée comme une logique des ressources. Elle n'est pas monotone et ne possède pas la propriété d'idempotence ce qui lui permet d'exprimer les notions de consommation, de production et de cumul de ressources. Ces propriétés vont permettre à la logique linéaire d'exprimer le fonctionnement des réseaux de Petri.

Chapitre 2

Formalisation des réseaux de Petri en logique linéaire

Les réseaux de Petri ont été introduit en 1962 par CARL ADAM PETRI. On leur connaît de nombreuses applications dans les protocoles de communication, la commande d'ateliers de fabrication, la conception de logiciels temps réel et/ou distribués, les systèmes d'information (organisation des entreprises) ou les interfaces homme-machine. Il sont notamment à l'origine de la norme GRAFCET utilisée dans les ateliers de production pour la programmation des Automates Programmables Industriels.

Nous allons voir dans ce chapitre leur utilisation pour l'évaluation des performances des systèmes à évènements discrets mais avec une approche par la logique linéaire, basée sur les « évènements » alors que les approches classiques sont orientées « états ».

2.1 Intérêt de l'approche par la logique linéaire

Dans la représentation des systèmes à évènements discrets, les réseaux de Petri sont au carrefour de différentes approches.

Ils sont classiquement étudiés par des méthodes orientées « états » qui construisent le graphe d'états du système. Le premier problème est que face à des systèmes complexes la construction de ce graphe d'états entraîne une explosion combinatoire. Le deuxième problème est que ces méthodes ne traitent pas le parallélisme « vrai » alors que les réseaux de Petri en sont capables grâce aux multiples jetons. Elles ne traitent qu'un parallélisme « logique » [7], qui est réalisé par entrelacement. Il ne peut en effet y avoir qu'un seul changement d'état à la fois. On représente donc deux évènements parallèles par le choix de celui qui arrivera en premier, suivi de l'autre. Enfin, le dernier problème que l'on peut soulever est que ces méthodes utilisent une représentation matricielle du réseau qui lorsqu'il devient complexe demande beaucoup d'espace inutile. En effet, une majeure partie de ces matrices sera constituée de zéros.

L'approche par la logique linéaire est quant à elle orientée « évènements », les évènements étant les franchissements de transitions. La structure du réseau est traduite en logique linéaire et l'on conserve ainsi un parallélisme « vrai ». On s'intéresse à des scénarios de tir, c'est à dire un ensemble non ordonné de franchissements de transitions. Ceci à l'avantage de ne pas nécessiter la construction du graphe d'états. De plus on ne va manipuler que les éléments structurels strictement nécessaires ce qui permet de limiter le volume de données à manipuler.

2.2 Définitions d'un réseau de Petri

Nous allons rappeler les principales définitions sur les réseaux de Petri.

2.2.1 Structure et marquage

Définition 1 (Réseau de Petri)

Un réseau de Petri \mathcal{R} est un quadruplet $\mathcal{R} = \langle \mathcal{P}, \mathcal{T}, Pre, Post \rangle$ où :

- $\mathcal{P} = (p_1, p_2, \dots, p_n)$ est un ensemble fini de places,
- $\mathcal{T} = (t_1, t_2, \dots, t_n)$ est un ensemble fini de transitions,
- $Pre : \mathcal{P} \times \mathcal{T} \rightarrow \mathbb{N}$ est l'application places précédentes,
- $Post : \mathcal{P} \times \mathcal{T} \rightarrow \mathbb{N}$ est l'application places suivantes,

Définition 2 (Marquage)

Le marquage M d'un réseau de Petri est une application $M : \mathcal{P} \rightarrow \mathbb{N}$ qui correspond à un état du réseau.

Définition 3 (Réseau Marqué)

Un réseau de Petri marqué est le couple $\mathcal{N} = \langle \mathcal{R}, M_0 \rangle$ où :

- \mathcal{R} est un réseau de Petri,
- M_0 est le marquage initial.

2.2.2 Représentations d'un réseau de Petri

On peut représenter un réseau de Petri de différente manière.

Représentation graphique

Un réseau de Petri peut être représenté sous la forme d'un graphe possédant deux types de noeuds : les places et les transitions. Un arc relie la place p à la transition t si et seulement si $Pre(p, t) \neq 0$ et la valuation de l'arc est alors égale à $Pre(p, t)$. De manière similaire, un arc relie la transition t à la place p si et seulement si $Post(p, t) \neq 0$ et la valuation de l'arc est alors égale à $Post(p, t)$.

Les places sont représentées par des cercles et les transitions par des rectangles. Le marquage d'une place est quant à lui représenté par des points placés dans les places, un point correspondant à un jeton. Un exemple de graphe associé à un réseau de Petri est présenté sur la figure 2.1.

Notation matricielle

Les applications *places précédentes* et *places suivantes* peuvent être représentées sous forme matricielle. Voici par exemple, la représentation matricielle du réseau de Petri de la figure 2.1 :

$$\mathcal{P} = \{P_1, P_2, P_3\} \quad \mathcal{T} = \{T_1, T_2, T_3, T_4\}$$

$$Pre = \begin{matrix} & T_1 & T_2 & T_3 & T_4 \\ \begin{matrix} P_1 \\ P_2 \\ P_3 \end{matrix} & \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix} \quad Post = \begin{matrix} & T_1 & T_2 & T_3 & T_4 \\ \begin{matrix} P_1 \\ P_2 \\ P_3 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{matrix}$$

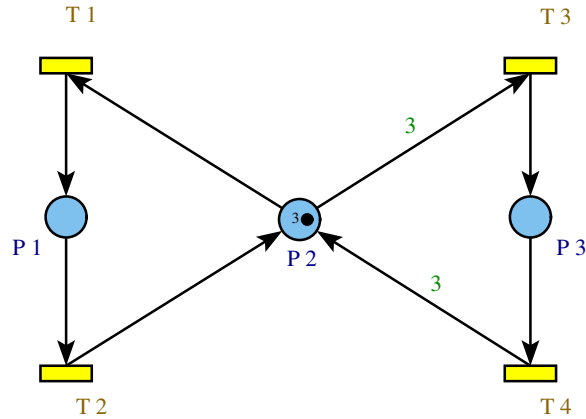


FIG. 2.1 – Représentation graphique d'un réseau de Petri marqué.

On utilise également la matrice $C = Post - Pre$ appelée *matrice d'incidence* du réseau de Petri.

Le marquage du réseau sera quant à lui représenté par un vecteur indexé par les places du réseau. Le marquage du réseau de Petri de la figure 2.1 est donc représenté par le vecteur :

$$M = \begin{matrix} P_1 \\ P_2 \\ P_3 \end{matrix} \begin{pmatrix} 0 \\ 3 \\ 0 \end{pmatrix}$$

On représente également sous la forme d'un vecteur les colonnes des matrices *Pre* et *Post* associées à la transition t et on les note $Pre(.,t)$ et $Post(.,t)$.

Représentation par un système de règle

Un réseau de Petri peut également être représenté par un système de règle. Cette représentation se rapproche de la représentation des réseaux de Petri par la logique linéaire qui sera détaillée par la suite. Plus particulièrement, un réseau de Petri est représenté à l'aide d'une classe particulière de système de règle appelée *grammaire*, qui est un système de réécriture de mots.

Pour cela, on se donne un alphabet et un ensemble de règles de réécriture. L'alphabet est l'ensemble des identificateurs des places. Le marquage du réseau est un mot écrit à l'aide de cet alphabet. Chaque transition du réseau est traduite par une règle de réécriture.

Dans l'exemple de la figure 2.1 :

- l'alphabet est $\{P_1, P_2, P_3\}$,
- le marquage initial est $P_2P_2P_2$, ce que l'on note P_2^3
- les règles de réécritures associées aux transitions sont :

$$\left\{ \begin{array}{l} T_1 : P_2 \rightarrow P_1 \\ T_2 : P_1 \rightarrow P_2 \\ T_3 : P_2^3 \rightarrow P_3 \\ T_4 : P_3 \rightarrow P_2^3 \end{array} \right.$$

2.2.3 Sensibilisations et franchissements de transitions

Transition franchissable

Une transition t est franchissable (ou sensibilisée ou *enabled*) à partir du marquage M si et seulement si :

$$\forall p \in \mathcal{P}, \quad M(p) \geq \text{Pre}(p,t),$$

ce que l'on note $M \geq \text{Pre}(\cdot,t)$ ou $M(>$ ou $M \xrightarrow{t}$.

Dans l'exemple de la figure 2.1 on a :

$$\text{Pre}(\cdot, T_1) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{Pre}(\cdot, T_2) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \text{Pre}(\cdot, T_3) = \begin{pmatrix} 0 \\ 3 \\ 0 \end{pmatrix} \text{Pre}(\cdot, T_4) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \text{ et } M_0 = \begin{pmatrix} 0 \\ 3 \\ 0 \end{pmatrix}$$

Il s'en suit que T_1 et T_3 sont sensibilisées pour le marquage initial M_0 alors que T_2 et T_4 ne le sont pas.

Franchissement d'une transition

Si la transition t est sensibilisée pour le marquage M , le franchissement (ou tir ou *firing*) de t donne un nouveau marquage M' tel que :

$$\forall p \in \mathcal{P}, \quad M'(p) = M(p) - \text{Pre}(p,t) + \text{Post}(p,t)$$

ce qui s'écrit en notation vectorielle $M' = M - \text{Pre}(\cdot,t) + \text{Post}(\cdot,t)$. On utilisera également pour signifier le franchissement de t les notations $M(t > M'$ ou $M \xrightarrow{t} M'$.

Dans l'exemple de la figure 2.1 le franchissement de la transition t_1 donne le nouveau marquage :

$$\begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

ce que l'on visualise sur la figure 2.2. Après ce franchissement, ce sont T_1 et T_2 qui sont sensibilisées.

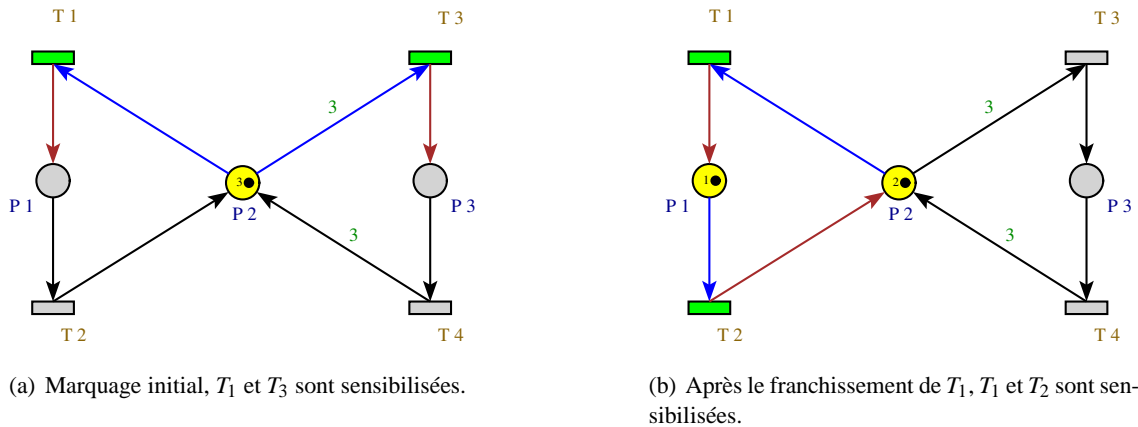


FIG. 2.2 – Franchissement d'une transition dans un réseau de Petri.

Séquence de franchissement

Si $M_0 \xrightarrow{t_a} M_1$ et $M_1 \xrightarrow{t_b} M_2$ on dit que la séquence $t_a; t_b$ est franchissable à partir de M_0 , ce que l'on note :

$$M_0 \xrightarrow{t_a; t_b} M_2$$

Le vecteur \bar{s} dont les composantes $\bar{s}(t)$ sont les nombres d'occurrences des transitions t dans une séquence de transition s est appelé *vecteur caractéristique* de s . Sa dimension est égale au nombre de transitions du réseau de Petri.

Dans l'exemple de la figure 2.1, on peut considérer la séquence de franchissement $s = T_1; T_1; T_2$ qui produit les marquages suivants :

$$\begin{pmatrix} 0 \\ 3 \\ 0 \end{pmatrix} \xrightarrow{T_1} \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \xrightarrow{T_1} \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} \xrightarrow{T_2} \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$$

Le vecteur caractéristique de s est :

$$\bar{s} = \begin{matrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{matrix} \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Équation fondamentale d'un réseau de Petri

Les évolutions du marquage d'un réseau de Petri sont données par l'équation :

$$M' = M - Pre.\bar{s} + Post.\bar{s}$$

ce que l'on peut également écrire :

$$M' = M + C.\bar{s} \quad \text{avec} \quad M \geq 0, \bar{s} \geq 0$$

Cette équation est appelée équation fondamentale d'un réseau de Petri.

Il faut toutefois noter que toute solution s de l'équation fondamentale ne constitue pas une séquence de transition effectivement franchissable du marquage M vers le marquage M' . Il faut également vérifier que toutes les transitions de la séquence sont franchissables à partir de leur marquage intermédiaire.

2.2.4 Conflits et parallélisme

Définition 4 (Conflit structurel) Deux transitions t_1 et t_2 sont en conflit structurel si et seulement si elles ont au moins une place d'entrée en commun :

$$\exists p \in \mathcal{P} \quad Pre(p, t_1).Pre(p, t_2) \neq 0$$

Définition 5 (Conflit effectif) t_1 et t_2 sont en conflit effectif pour un marquage M si et seulement si elles sont en conflit structurel et :

$$M \geq Pre(., t_1)$$

$$M \geq Pre(., t_2)$$

Définition 6 (Parallélisme structurel) Deux transitions t_1 et t_2 sont parallèles structurellement si le produit scalaire suivant est nul :

$$(Pre(.,t_1))^T \times Pre(.,t_2) = 0$$

C'est à dire qu'elles n'ont aucune place d'entrée commune.

Définition 7 (Parallélisme effectif) t_1 et t_2 sont parallèles pour un marquage M si et seulement si elles sont parallèles structurellement et :

$$M \geq Pre(.,t_1)$$

$$M \geq Pre(.,t_2)$$

2.3 Traduction des réseaux de Petri en logique linéaire

A la base du fonctionnement des réseaux de Petri on trouve les notions de consommation et de production de jetons. Ces notions nous l'avons vu sont aisément traductibles en logique linéaire. La traduction des réseaux de Petri en logique linéaire a donc été envisagée peu de temps après la publication de cette nouvelle théorie. Plusieurs approches ont été envisagées, GIRAULT fait dans sa thèse [2] une étude critique des différentes techniques de traduction proposées. La traduction que nous allons détailler est celle développée au LAAS [2, 3, 9] et reprise dans les travaux déjà réalisés à l'IRCCyN.

2.3.1 Traduction de la structure du réseau

Soit un réseau de Petri $\mathcal{R} = (\mathcal{P}, \mathcal{T}, Pre, Post)$. A chaque place $p \in \mathcal{P}$ on associe un atome propositionnel P . L'ensemble des formules propositionnelles P associées aux places p de \mathcal{R} constitue l'*alphabet logique* du réseau de Petri. A partir de cet alphabet on va écrire des formules de LL qui vont représenter la structure du réseau de Petri.

Marquage

Un marquage du réseau est traduit par une conjonction de ressources, donc en utilisant le connecteur \otimes , les ressources étant les jetons du marquage. Dans cette formule chaque ressource P est présente au temps de fois qu'il y a de jetons dans la place p . De manière générale, un marquage m de \mathcal{R} se traduit par la formule M de LL définie par :

$$M = \bigotimes_{p \in \mathcal{P}} A_p^{\otimes m(p)}$$

où $m(p)$ est la multiplicité de la place p dans le marquage m .

Dans l'exemple de la figure 2.1, le marquage initial du réseau est traduit par la formule de LL :

$$P_2 \otimes P_2 \otimes P_2 = P_2^{\otimes 3}$$

Transitions

Chaque transition $t \in \mathcal{T}$ de \mathcal{R} est traduite par une formule implicative utilisant le connecteur \multimap . Le membre droit du connecteur est le marquage de pré-condition de la transition, le membre gauche le marquage de post-condition, ils sont tous les deux exprimés à l'aide d'une formule en \otimes . Ceci s'écrit :

$$t : \quad \bigotimes_{i \in \text{Pre}(p_i, t)} P_i \multimap \bigotimes_{o \in \text{Post}(p_o, t)} P_o$$

Cette formule traduit bien le sens d'une transition dans un réseau de Petri : elle consomme le marquage d'entrée pour produire le marquage de sortie.

Dans l'exemple de la figure 2.1, les transitions sont traduites par les formules suivantes :

$$\begin{aligned} T_1 &: P_2 \multimap P_1 \\ T_2 &: P_1 \multimap P_2 \\ T_3 &: P_2 \otimes P_2 \otimes P_2 \multimap P_3 \\ T_4 &: P_3 \multimap P_2 \otimes P_2 \otimes P_2 \end{aligned}$$

2.3.2 Traduction du fonctionnement du réseau

Alors que la structure du réseau n'est traduite qu'à l'aide de formules de logique linéaire, pour traduire le fonctionnement du réseau, c'est à dire le franchissement des transitions, nous utilisons la notion de séquent.

Le franchissement d'une transition t à partir d'un marquage m pour donner le nouveau marquage m' est exprimé par le séquent suivant :

$$M, T \vdash M'$$

où M et M' sont les formules de LL associées aux marquages m et m' et T est la formule de LL associée à la transition t .

Le franchissement d'une séquence de transition $t_1; t_2; \dots; t_n$ est exprimée de manière similaire par le séquent :

$$M, T_1, T_2, \dots, T_n \vdash M'$$

Dans [2] il a été démontré que tout séquent de ce type est prouvable si et seulement si le marquage m' est accessible à partir du marquage m .

Remarques

1. Les formules de LL associées aux transitions sont considérées comme des ressources. Elles sont donc consommées par le séquent de la même manière que le marquage. Ainsi dans l'exemple de la figure 2.1, pour franchir la séquence $t_1; t_1; t_2$ il faut utiliser deux ressources d'action T_1 ce qui se traduit par le séquent :

$$P_2^{\otimes 3}, T_1, T_1, T_2 \vdash P_1 \otimes P_2^{\otimes 2}$$

2. Une séquence de transitions est totalement ordonnée. Par contre dans un séquent les transitions ne sont pas ordonnées, la virgule étant commutative. Un séquent de LL représente donc l'ensemble des séquences de transitions permettant d'accéder à un marquage donné avec le nombre de franchissements donnés.

2.3.3 Preuve d'un séquent

Pour réaliser la preuve d'un séquent exprimant l'accessibilité d'un marquage, nous utilisons principalement les deux règles du calcul des séquents suivantes :

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C} \otimes_L \qquad \frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C} \multimap_L$$

La règle \otimes_L permet de déconnecter le marquage afin de rendre les jetons utilisables indépendamment. Dans le réseau de la figure 2.1 en appliquant cette règle à partir du marquage initial on passe de $P_2^{\otimes 3}$ à P_2, P_2, P_2 . Ceci permet de déconnecter les trois jetons de la place P_2 pour pouvoir par exemple en utiliser un seul pour franchir T_1 .

La règle \multimap_L permet d'effectuer le tir d'une transition. Pour cela on retire des ressources les jetons consommés et la ressource de la transition tirée. Ceci produit à droite un séquent prémisses qui est, si la règle est correctement appliquée, trivial à prouver, et à gauche un séquent qui représente le réseau de Petri après le tir de la transition et qui contient les transitions restantes.

Les deux règles *id* et \otimes_R pourront être utilisées pour simplement terminer de manière rigoureuse les preuves. Sans ces règles on peut considérer que la preuve est correcte lorsque toutes les feuilles sont du type $A \vdash A$ ou $A_1, A_2, \dots, A_n \vdash A_1 \otimes A_2 \otimes \dots \otimes A_n$.

Remarque : au cours de la preuve, lorsque l'on considère les atomes déconnectés du membre gauche d'un séquent, on ne parlera pas de marquage mais d'*étape courante*. Nous verrons en quoi diffèrent ces deux termes lorsque l'on étudiera les réseaux de Petri temporels dans la partie 3.2.

Algorithme de preuve canonique

L'algorithme suivant proposé par R. VALETTE du LAAS permet de prouver un séquent exprimant l'accessibilité d'un marquage. La règle \multimap_L est appliquée en choisissant une transition candidate dans l'ordre lexicographique. Cet ordre arbitraire permet de réaliser des preuves canoniques, c'est à dire que pour chaque séquent un arbre de preuve unique est calculé. En cas de parallélisme ou de conflits plusieurs ordres de tir existent. Pour étudier l'ensemble des comportements il faudrait donc réaliser plusieurs arbres de preuves. Cependant nous verrons qu'en utilisant des annotations temporelles nous pourrions nous contenter de l'arbre de preuve canonique, du moins pour étudier le parallélisme.

Appliquer la règle \otimes_L au temps de fois que nécessaire pour transformer le marquage initial en une liste d'atomes séparés par la virgule

Tant que la règle \multimap_L est applicable (c'est à dire si le marquage d'entrée d'au moins une des transitions est inclus dans la liste d'atomes de l'étape courante) **Faire**

- Appliquer la règle \multimap_L à la transition candidate figurant en premier dans l'ordre lexicographique
- Terminer la preuve du séquent gauche généré en utilisant, si nécessaire, la règle \otimes_R
- Appliquer, si nécessaire, la règle \otimes_L au séquent droit

Fin Tant que

Algorithme 1 : Algorithme de preuve canonique

Considérons le séquent suivant : $P_2^{\otimes 3}, T_3, T_4 \vdash P_2^{\otimes 3}$ qui correspond à la séquence de tir $t_3; t_4$ pour le réseau de Petri de la figure 2.1. L'arbre de preuve canonique de ce séquent fournit par cet algorithme est présenté sur figure 2.3.

$$\begin{array}{c}
\frac{\frac{\frac{\overline{P_2 \vdash P_2} \text{ Id}}{P_2, P_2, P_2 \vdash P_2^{\otimes 3}} \otimes R \quad \frac{\overline{P_2 \vdash P_2} \text{ Id} \quad \overline{P_2 \vdash P_2} \text{ Id}}{P_2, P_2 \vdash P_2^{\otimes 2}} \otimes R}{P_2, P_2, P_2 \vdash P_2^{\otimes 3}} \otimes R \quad \frac{\overline{P_3 \vdash P_3} \text{ Id}}{P_3, P_3 \multimap P_2^{\otimes 3} \vdash P_2^{\otimes 3}} \multimap L}{\frac{P_2, P_2, P_2, P_2^{\otimes 3} \multimap P_3, P_3 \multimap P_2^{\otimes 3} \vdash P_2^{\otimes 3}}{P_2^{\otimes 3}, P_2^{\otimes 3} \multimap P_3, P_3 \multimap P_2^{\otimes 3} \vdash P_2^{\otimes 3}} \otimes L} \otimes R}
\end{array}$$

FIG. 2.3 – Arbre de preuve canonique d'un séquent

2.4 Conclusion

Nous avons vu dans ce chapitre que les réseaux de Petri et la logique linéaire sont deux théories très proches. Il est ainsi facile de traduire un réseau de Petri en logique linéaire. Il existe un algorithme permettant de réaliser des preuves canoniques de séquents exprimant des scénarios sur des réseaux de Petri et ainsi de démontrer l'accessibilité d'un marquage. Nous allons maintenant voir que cette approche des réseaux de Petri par la logique linéaire peut être très intéressante avec les réseaux de Petri T-temporels.

Chapitre 3

Analyses temporelles des réseaux de Petri T-temporels

Les réseaux de Petri ordinaires définissent des relations de causalité entre évènements. Le temps est donc pris en compte de manière qualitative. Les réseaux de Petri T-temporels sont un modèle de réseau de Petri qui prend en compte le temps de manière quantitative. Nous allons voir en quoi la logique linéaire peut être utile dans l'analyse de ces réseaux.

3.1 Réseaux de Petri T-temporels

Définition 8 *Un réseau de Petri T-temporel est une paire $\langle \mathcal{N}, I \rangle$ où :*

- \mathcal{N} est un réseau de Petri $\langle \mathcal{P}, \mathcal{T}, Pre, Post \rangle$ muni d'un marquage initial M_0 ,
- I est une fonction qui à chaque transition t fait correspondre un intervalle fermé rationnel $i(t) = [\theta_{s_{min}}(t), \theta_{s_{max}}(t)]$ qui décrit une **durée de sensibilisation**.

Sémantique faible des réseaux de Petri T-temporels

Une transition sensibilisée ne peut être franchie que si sa durée de sensibilisation est comprise dans l'intervalle $i(t)$. On associe donc à chaque transition sensibilisée une horloge $v(t)$ qui comptabilise le temps écoulé depuis la sensibilisation de la transition t . L'état du réseau est donc défini par le marquage courant et par la valeur des horloges associées aux transitions sensibilisées.

Pour faire évoluer le réseau il existe deux types de changement d'état. Le premier consiste à franchir une transition sensibilisée dont l'horloge est comprise dans l'intervalle de sensibilisation. Ce franchissement est immédiat. Les transitions nouvellement sensibilisées par ce franchissement ont leur horloge initialisée à zéro. Les horloges des transitions qui étaient déjà sensibilisées ne sont pas modifiées. Le deuxième type de changement d'état est l'écoulement du temps. On augmente alors toutes les horloges de la durée écoulée.

En sémantique faible il n'y a pas d'urgence pour le tir d'une transition. En cas de conflit, on peut décider de ne pas tirer une transition pour réserver un jeton pour une autre transition en conflit.

3.2 Algorithme de VALETTE avec estampilles temporelles

Cette algorithme est présenté dans [4, 5]. Il reprend l'algorithme de preuve canonique mais au cours de la preuve des estampilles temporelles associées aux atomes sont calculées. Ces estampilles

vont permettre de déterminer la durée d'un scénario.

3.2.1 Estampilles temporelles

On associe à chaque atome une estampille temporelle. Cette estampille est la date d'arrivée du jeton dans la place, c'est à dire la date de production du jeton. Ces estampilles sont calculées de manière symbolique à l'aide des durées de sensibilisation des transitions. Elles permettent de déterminer les relations de précédence entre les franchissement de transitions et de calculer la durée totale d'un scénario.

On peut également associer à un atome une date de consommation. On notera $A(D_p, \cdot)$ un atome dont la date de production est D_p et qui n'a pas de date de consommation et $A(D_p, D_c)$ si il a une date de consommation D_c .

En calculant la différence $D_c - D_p$ on détermine la *durée de séjour* d'un atome, c'est à dire la durée de séjour du jeton dans la place correspondante.

Étape courante

Lors d'une preuve, dans le membre gauche d'un séquent, la liste d'atomes séparés par le méta-connecteur " ," est appelée *étape courante*.

Une étape courante ne correspond pas nécessairement à un marquage effectivement atteint lors d'un scénario. En effet, selon leur date d'arrivée et leur durée de séjour, les jetons d'une étape courante peuvent coexister ou ne pas coexister simultanément. Ainsi, la présence dans une preuve d'une étape courante du type $A(D_{P_A}, D_{C_A}), B(D_{P_B}, D_{C_B})$ n'implique pas que le marquage $A \otimes B$ soit atteint. Il est par exemple possible que le jeton A soit consommé avant que le jeton B ne soit produit.

3.2.2 Calcul des estampilles temporelles

Lors de l'application de la règle \multimap_L qui permet le tir d'une transition on effectue le calcul des estampilles temporelles.

- La date de production du marquage produit par la transition est égale au maximum des dates de production des atomes consommés augmenté de la durée de sensibilisation de la transition tirée.
- Les autres atomes conservent leur estampille temporelle.

La date de production calculée est la date de tir de la transition. On peut associer cette date aux atomes consommés en tant que date de consommation.

3.2.3 Algorithme

L'algorithme de VALETTE avec annotation temporelle (Algorithme 2) n'est applicable que sur des scénarios complètement spécifiés. Cela signifie qu'il ne doit pas comporter de conflits non résolus. La classe des réseaux de Petri qui sont des *graphes d'évènements*¹ permet de s'assurer que les scénarios sont complètement spécifiés. Ce point sera réabordé dans la partie 3.3.

On calcule avec cette algorithme les dates de production de tous les jetons impliqués dans le scénario. On peut notamment en déduire la *durée du scénario*. Elle est égale à la différence entre le maximum des dates de production du marquage final et le maximum des dates de production du marquage initial.

¹Deux transitions n'ont pas de places d'entrée ou de sortie commune.

Vérifier que le fragment du réseau correspondant au scénario est un graphe d'évènements
 Appliquer la règle $\otimes L$: les estampilles temporelles des atomes de l'étape courante sont initialisées à 0

Tant que la règle $\multimap L$ est applicable **Faire**

- Appliquer la règle $\multimap L$ à la transition candidate figurant en premier dans l'ordre lexicographique : l'estampille temporelle du marquage produit est égale au maximum des estampilles des atomes consommés augmenté de la durée de sensibilisation de cette transition ; les autres estampilles sont inchangées
- Terminer la preuve du séquent gauche généré en utilisant, si nécessaire, la règle $\otimes R$
- Appliquer, si nécessaire, la règle $\otimes L$ au séquent droit : l'estampille temporelle des atomes déconnectés est égale à celle du marquage

Fin Tant que

Algorithme 2 : Algorithme de preuve avec estampilles temporelles

3.2.4 Application sur un exemple

Considérons le réseau de Petri de la figure 3.1. Il ne présente pas de conflit mais deux transitions en parallèle : T_2 et T_3 . Nous allons étudier le scénario défini dans le séquent suivant : $P_1, T_1, T_2, T_3, T_4 \vdash P_6$.

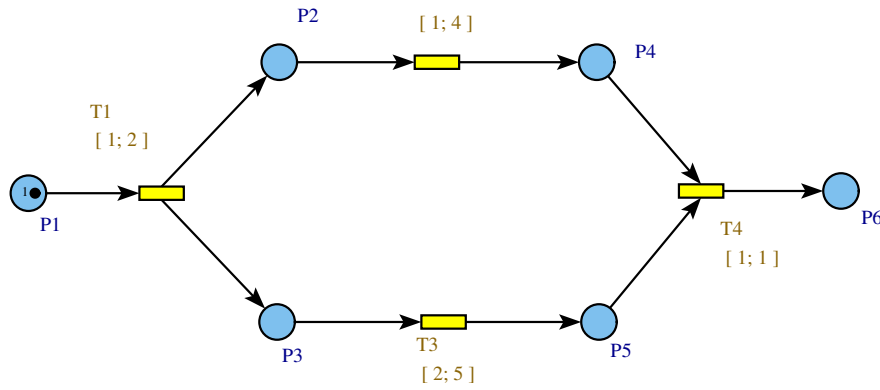


FIG. 3.1 – Réseau de Petri avec parallélisme

Voici l'arbre de preuve produit par l'algorithme 2, décomposé en différentes parties.

Franchissement de T_1 :

$$\frac{\frac{P1(0, d_1) \vdash P1 \quad Id \quad \frac{P2(d_1, \cdot), P3(d_1, \cdot), T_2, T_3, T_4 \vdash P6}{P2 \otimes P3(d_1, \cdot), T_2, T_3, T_4 \vdash P6} \otimes L}{P1(0, \cdot), P1 \multimap P2 \otimes P3, T_2, T_3, T_4 \vdash P6} \multimap L}{P1(0, \cdot), P1 \multimap P2 \otimes P3, T_2, T_3, T_4 \vdash P6} \multimap L$$

Franchissement de T_2 puis de T_3 :

$$\frac{\frac{P2(d_1, d_1 + d_2) \vdash P2 \quad Id \quad \frac{P3(d_1, d_1 + d_3) \vdash P3 \quad Id \quad P4(d_1 + d_2, \cdot), P5(d_1 + d_3, \cdot), T_4 \vdash P6}{P3(d_1, \cdot), P4(d_1 + d_2, \cdot), P3 \multimap P5, T_4 \vdash P6} \multimap L}{P2(d_1, \cdot), P3(d_1, \cdot), P2 \multimap P4, P3 \multimap P5, T_4 \vdash P6} \multimap L}{P2(d_1, \cdot), P3(d_1, \cdot), P2 \multimap P4, P3 \multimap P5, T_4 \vdash P6} \multimap L$$

Bien que parallèles, il faut dans la preuve donner un ordre de franchissement entre T_2 et T_3 . Cet ordre est arbitraire, nous choisissons l'ordre lexicographique. Cependant les estampilles temporelles calcu-

lées prennent correctement en compte le parallélisme entre T_2 et T_3 .

Franchissement de T_4 :

$$\frac{\frac{\overline{P4(d_1 + d_2, D_{T_4})} \vdash P4 \quad Id \quad \overline{P5(d_1 + d_3, D_{T_4})} \vdash P5 \quad Id}{\overline{P4(d_1 + d_2, D_{T_4}), P5(d_1 + d_3, D_{T_4})} \vdash P4 \otimes P5} \otimes R \quad \frac{\overline{P6(D_{T_4}, \cdot)} \vdash P6 \quad Id}{\overline{P4(d_1 + d_2, \cdot), P5(d_1 + d_3, \cdot), P4 \otimes P5} \dashv\vdash P6} \dashv\vdash L}{\overline{P4(d_1 + d_2, \cdot), P5(d_1 + d_3, \cdot), P4 \otimes P5} \dashv\vdash P6} \dashv\vdash L$$

D_{T_4} est la date de tir de la transition T_4 , $D_{T_4} = \max(d_1 + d_2, d_1 + d_3) + d_4$.

Résultats

Le tableau 3.1 récapitule les dates de production et de consommation calculées pour chaque jeton utilisé dans le scénario.

Atomes	Date de production	Date de consommation
P_1	0	d_1
P_2	d_1	$d_1 + d_2$
P_3	d_1	$d_1 + d_3$
P_4	$d_1 + d_2$	D_{T_4}
P_5	$d_1 + d_3$	D_{T_4}
P_6	D_{T_4}	.

TAB. 3.1 – Estampilles temporelles calculées par l’algorithme de VALETTE

La durée du scénario est égale à la date de production de l’atome P_6 . Elle donc égale à :

$$\max(d_1 + d_2, d_1 + d_3) + d_4 = d_1 + \max(d_2, d_3) + d_4$$

En regardant les dates de production de P_4 et P_5 on s’aperçoit que le parallélisme de T_2 et T_3 a bien été pris en compte. En effet, pour P_4 sa date est $d_1 + d_2$ ce qui indique uniquement une précédence entre T_1 et T_2 . De même pour P_5 , T_3 ne précède que T_1 alors que dans l’arbre de preuve T_3 a été tirée après T_2 .

L’avantage de cette algorithme est que l’on obtient des dates formelles qui sont plus riches en informations que des valeurs numériques. On peut de plus effectuer plusieurs applications numériques en faisant varier les longueurs des intervalles de sensibilisation.

3.3 Retour sur la notion de scénario

Un scénario est un ensemble non-ordonné de transitions. Cela correspond à un ensemble de comportements du réseau de Petri. A un scénario peut correspondre plusieurs arbres de preuves selon l’entrelacement des transitions. Cependant, en nous intéressant à un seul de ces arbres, construit de manière canonique, donc unique pour chaque séquent, nous pouvons en déduire des informations temporelles valables pour tous les entrelacements. Ainsi dans l’exemple précédent la durée calculée pour le scénario étudiée est valable pour l’entrelacement choisit dans l’arbre de preuve (T_2 avant T_3) mais aussi pour l’autre entrelacement (T_3 avant T_2).

Cependant ceci n'est possible que si le scénario est *complètement spécifié*. En effet en présence de conflits, il sera nécessaire de calculer plusieurs arbres de preuves pour étudier l'ensemble des comportements d'un scénario.

3.3.1 Conflits de transitions et conflits de jetons

Nous allons définir ces deux notions de conflit qui sont relatives à l'arbre de preuve d'un scénario. Considérons un scénario exprimé par le séquent suivant :

$$\text{Étape}, M_{e1} \multimap M_{s1}, \dots, M_{en} \multimap M_{sn} \vdash \text{Marquage}$$

Définition 9 (Conflit de transitions) Deux transitions t_i et t_j du scénario sont en conflit si l'un des atomes de l'étape courante *Étape* est inclus dans les préconditions de t_i et de t_j . Les conflits considérés ici ne sont pas nécessairement des conflits effectifs.

Cette définition étend la définition de conflit donnée dans la partie 2.2 afin de l'adapter à la logique linéaire.

Considérons comme exemple le réseau de la figure 3.2(a) et le scénario $P_1 \otimes P_2, T_1, T_2, T_3 \vdash P_3 \otimes P_4$. Les deux transitions T_2 et T_3 sont en conflit. Si l'on tire T_2 avant T_3 , la durée du scénario est $\max(d_2, d_1 + d_2)$, alors que si l'on tir T_2 après T_3 , la durée est $\max(d_3, d_1 + d_2)$.

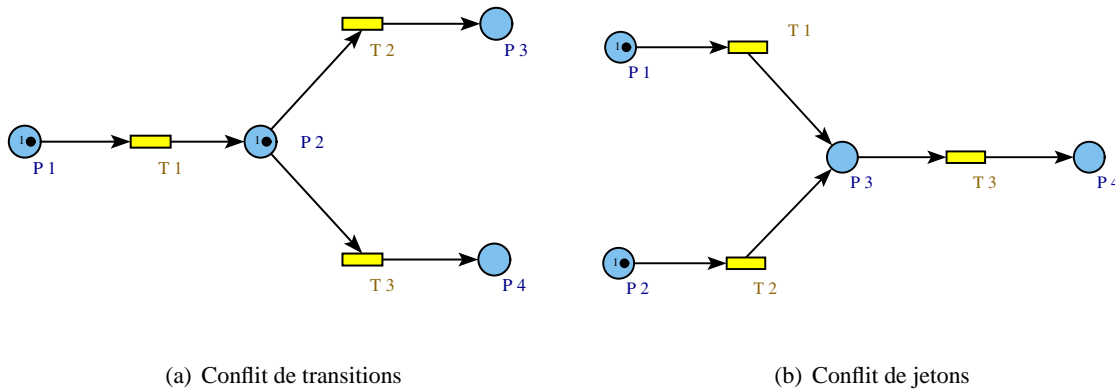


FIG. 3.2 – Conflits de jetons et de transitions

Définition 10 (Conflit de jeton) Il y a un conflit de jeton si l'étape courante *Étape* contient un nombre de jetons dans la place P supérieur au nombre de jetons nécessaires au tir d'une transition t du scénario et si les labels temporels de ces jetons sont non totalement ordonnés.

Le réseau de la figure 3.2(b) présente un conflit de jetons. En effet dans le scénario suivant : $P_1 \otimes P_2, T_1, T_2, T_3 \vdash P_3 \otimes P_4$ le tir de T_3 pourra se faire soit avec le jeton provenant du tir de T_1 soit par celui provenant de T_2 . Dans le premier cas la durée du scénario est $\max(d_2, d_1 + d_3)$ et dans le deuxième $\max(d_1, d_2 + d_3)$.

3.3.2 Scénarios complètement spécifiés

Définition 11 Si au cours de la preuve d'un séquent $M, \sigma \vdash M'$ il apparaît un conflit entre des jetons ou entres des transitions, alors on dit que le scénario défini par ce séquent est incomplètement spécifié.

Dans ces scénarios il est nécessaire d'effectuer des choix externes pour complètement spécifier le scénario. Plusieurs arbres de preuves seront alors nécessaires pour étudier de manière exhaustive le scénario.

L'algorithme de VALETTE se restreint donc à des scénarios complètement spécifiés car il ne peut fournir qu'un seul arbre de preuve. Cependant, la définition d'un scénario complètement spécifié est dynamique. Il est en effet nécessaire pour la vérifier d'effectuer la preuve du scénario puisque les définitions des conflits de transitions et de jetons sont relatives à une étape courante de la preuve.

Toutefois, il existe une condition suffisante. En effet, les réseaux de Petri qui sont des graphes d'évènements, c'est à dire tels que deux transitions du réseau n'ont aucune place d'entrée ou de sortie commune, assure que les scénarios sont complètement spécifiés. Cette contrainte peut ne pas être appliquée à tout le réseau mais seulement au sous-réseau impliqué dans le scénario. Cette vérification est donc effectuée au début de l'algorithme.

3.3.3 Parallélisme et logique linéaire

Le parallélisme est une des caractéristiques principales des réseaux de Petri. La possibilité de créer plusieurs jetons permet en effet à plusieurs activités de se dérouler en parallèle. Dans les approches orientées « états », ce parallélisme des réseaux de Petri n'est pris en compte que par l'entrelacement des franchissements. Ceci peut entraîner des problèmes pour l'étude de certaines propriétés. En particulier, dans les réseaux de Petri T-temporels, l'exploration de l'espace d'états peut être effectuée par le graphe des classes. Si l'on étudie certaines propriétés temporelles avec ce graphe, telles que la durée d'un scénario, on aboutit à des erreurs [4] qui sont dues à l'absence de parallélisme vrai.

L'approche par la logique linéaire permet elle de prendre en compte le parallélisme vrai. Tout d'abord un séquent de logique linéaire ne précise pas d'ordre entre les transitions. Ainsi il représente tous les entrelacements possibles entre ses transitions. Pour le prouver, un seul arbre de preuve est nécessaire. Ensuite, pour effectuer des analyses temporelles, nous associons à chaque jeton un label temporel. Ces labels sont indépendants entre eux, ainsi, lors du tir d'une transition, seuls les labels concernés par ce tir sont modifiés. Au final, pour calculer la durée d'un scénario un seul arbre de preuve est nécessaire, et il prend en compte le parallélisme vrai. Le résultat que l'obtient est alors correct, il contient exactement la durée possible d'un scénario.

3.4 Conclusion

En assignant des estampilles temporelles aux jetons, on est capable à l'aide de l'algorithme de VALETTE de calculer de manière formelle des durée de scénarios. Cependant cette algorithme est limité à la sémantique faible des réseaux de Petri T-temporels. Cette sémantique est incapable de représenter l'urgence ce qui est pourtant nécessaire, notamment pour les applications temps réel. Nous allons voir comment cet algorithme est adapté afin de prendre en compte la sémantique forte.

Chapitre 4

Prise en compte de la sémantique forte

Cette prise en compte de la sémantique forte a été réalisée par S.REVOL puis M.SOGBHOSSOU lors de leur DEA à l'IRCCyN [6, 8]. Nous allons dans ce chapitre présenter leurs travaux.

4.1 Sémantique faible et sémantique forte

4.1.1 Différences entre les sémantiques

Nous avons présenté dans le chapitre précédent la sémantique de tir faible (STFA) des réseaux de Petri T-temporels. En sémantique faible, si une transition sensibilisée atteint sa durée maximale de sensibilisation on n'est pas obligé de tirer la transition. Elle ne peut cependant plus être tirée par la suite ce qui peut entraîner des jetons morts. Ne pas franchir une transition n'a d'intérêt qu'en cas de conflit de transitions. Dans ce cas, on peut en effet souhaiter réserver les jetons en conflits pour le tir d'une autre transition qui n'est pas encore sensibilisée.

Dans la sémantique de tir forte (STFO) une transition qui atteint sa borne maximale doit obligatoirement être tirée. Formellement, on ne peut pas faire écouler une durée de temps, si en ajoutant cette durée aux horloges, une des horloges dépasse la borne maximale de son intervalle de sensibilisation.

La sémantique forte a beaucoup d'intérêt dans le cas des applications temps réel car elle permet de modéliser le mécanisme de *chien de garde* qui est présenté sur la figure 4.1(a). Ce mécanisme fonctionne ainsi :

- Lorsqu'un jeton arrive en P_1 le chien est armé, l'horloge de la transition T_2 est démarrée.
- Si un jeton arrive en P_2 avant que l'horloge atteigne sa borne maximale, alors la sémantique forte impose que la transition T_1 dont la durée de sensibilisation est $[0, 0]$ soit franchie immédiatement. Ceci correspond en général à l'exécution normale du système.
- Par contre si l'horloge atteint la borne maximale fixée ici à 10, alors la sémantique forte impose le tir de T_2 , ce qui peut par exemple correspondre à une routine de secours.

Ce mécanisme n'est par contre pas modélisable en STFA.

Dans le calcul des dates de tir des transitions la sémantique forte va modifier les résultats. Si l'on considère par exemple la figure 4.1(b). Si le marquage initial est produit à la date 0, en STFA la date de tir de T_2 est comprise dans l'intervalle $[2, 5]$. En STFO par contre, au bout de 3 unités de temps T_1 devient urgente et l'on est donc obligé de la franchir. T_2 ne peut donc pas être franchie au delà de la date 3. En STFO il va donc falloir modifier le calcul des dates de tir afin d'obtenir des valeurs correctes. Ces valeurs ne sont cependant modifiées qu'en cas de conflit.

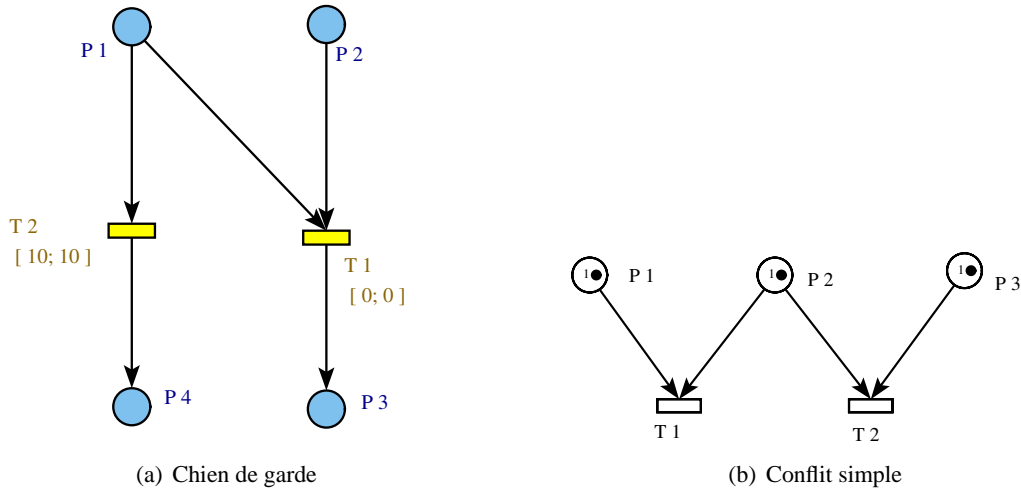


FIG. 4.1 – Situations de conflits dans un réseau T-temporel

4.1.2 Sémantique de tir et logique linéaire

Dans le cas de la sémantique faible, les réseaux de Petri T-temporels sont monotones, c'est à dire que le rajout de jetons dans des places quelconques du réseau ne modifie pas le comportement du réseau et les marquages accessibles. En effet, une séquence de transition franchissable à partir d'un marquage donné, le sera également pour tous marquages supérieurs puisque qu'il sera possible de réserver tous les jetons nécessaires à cette séquence. La logique linéaire obéit également à cette règle. En effet, si un séquent est prouvable il le sera également en rajoutant des ressources.

Au contraire, en sémantique forte, l'ajout d'un jeton dans le réseau ou d'une ressource dans un séquent peut modifier la validité du scénario étudié. En effet, ce jeton peut activer un conflit qui aura pour conséquence d'invalider le scénario. De plus, alors qu'en STFA on pouvait se limiter à une partie du réseau de Petri étudié, en STFO on est obligé de considérer l'ensemble du réseau.

C'est pour cette raison de monotonie que l'algorithme de VALETTE n'est utilisable qu'en sémantique faible. Cependant, REVOL a proposé dans [6] une version modifiée de l'algorithme de VALETTE qui prend en compte la STFA.

4.2 Algorithme de preuve en sémantique forte

4.2.1 Repérage des conflits de transitions

Les transitions dont la date de tir peut être modifiée par la sémantique forte sont celles qui sont en conflits. Il est donc nécessaire de repérer les transitions en conflits dans le réseau de Petri, c'est à dire celles qui partagent une même place d'entrée. On recherche donc toutes les transitions en conflits structurels.

Mais considérons l'exemple de la figure 4.2. Il présente deux conflits, entre T_1 et T_2 d'une part et entre T_2 et T_3 d'autre part. En STFO, T_1 ne peut pas être tirée au delà de $d2max$ (si $d2max < d1min$ T_1 n'est même normalement jamais tirable). Par contre, si T_3 est tirée alors T_2 n'est plus tirable et T_1 n'est donc plus en conflit effectif. Elle peut alors être franchie jusqu'à sa borne maximale. On s'aperçoit donc que T_1 et T_3 ont une action l'une sur l'autre parce qu'elles sont toutes les deux en conflits avec

T_2 . On dira que T_1 et T_3 sont en conflit indirect même si leur interaction ne peut être que bénéfique.

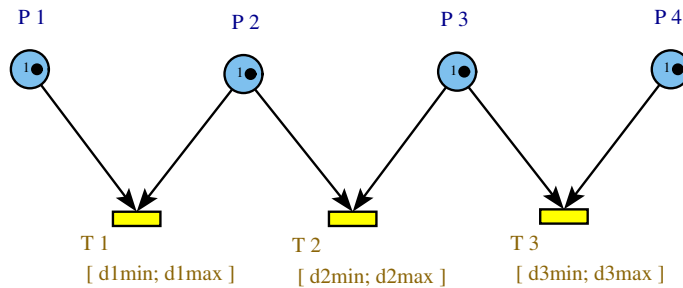


FIG. 4.2 – Conflit indirect entre deux transitions

Pour identifier les conflits dans le réseau, on va donc construire des groupes de conflits qui regrouperont les transitions en conflit entre elles, que ce soient des conflits directs ou indirects. Ces groupes de conflits sont construits par transitivité suivant la règle suivante : *deux transitions en conflit sont dans un même groupe*.

4.2.2 Modification de l'ordre de la preuve

Nous avons vu dans la partie 3.3, que lorsqu'il y a dans un scénario des conflits de transitions, l'ordre de tir des transitions avait de l'importance vis à vis des dates calculées. On va donc entre les transitions en conflits fixer un ordre de tir qui sera l'ordre dans lequel les transitions sont placées dans le séquent. Pour examiner tout les comportements possibles, il sera nécessaire de modifier cet ordre afin d'étudier tous les entrelacements.

Remarques

- Nous avons considéré les règles d'échanges étaient implicites ce qui entraîne que la virgule est commutative. Ce n'est plus tout à fait le cas dans cet algorithme.
- Dans le cas de deux transitions appartenant à des groupes de conflits différents, l'ordre d'examen sera fixé par le séquent, mais il ne sera pas nécessaire d'étudier l'ordre inverse car le parallélisme sera correctement pris en compte.
- Dans le cas des conflits indirects de la figure 4.2 les deux transitions peuvent être franchies en parallèles mais l'ordre à de l'importance. Il sera donc nécessaire d'effectuer l'entrelacement.
- Ainsi, par rapport à ce qu'on a dit dans la partie 3.3 sur le parallélisme et la logique linéaire, nous réintroduisons ici de l'entrelacement, afin de prendre en compte les conflits.

Par ailleurs, en cas de conflit non-effectif il est nécessaire de savoir si ce conflit pourra devenir effectif et dans quel intervalle de temps. Considérons par exemple le réseau de la figure 4.3. La date de tir de la transition T_2 dépend du conflit entre T_1 et T_2 . Il est donc nécessaire de savoir à partir de quand ce conflit devient effectif. Il faut donc calculer au préalable la date de tir de T_3 et pour cela il faut que T_3 soit tirée avant T_2 .

De manière générale, on va tirer en priorité les transitions qui ne sont pas en conflits. Le tir de ces transitions en priorité va permettre de rendre effectifs le maximum de conflits. Étant donné que l'ordre de tir est normalement arbitraire dans une preuve ceci est acceptable.

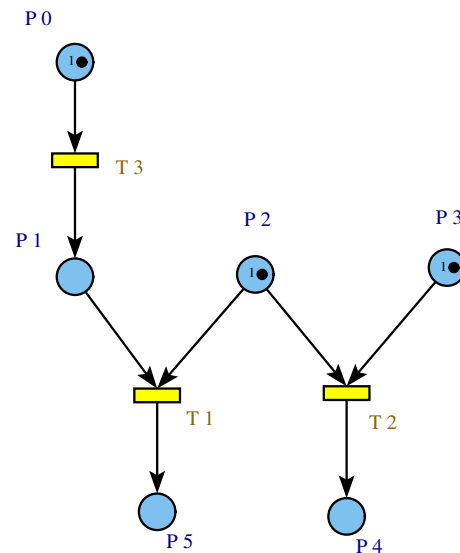


FIG. 4.3 – Exemple de réseau de Petri avec conflit

Ainsi à une étape d'une preuve, l'ordre suivant d'examen des transitions sera appliqué :

- En priorité les transitions qui n'appartiennent pas à un groupe de conflits. Entre elles un ordre lexicographique est appliqué.
- Ensuite les transitions en conflit, dans l'ordre de lecture du séquent.

4.2.3 Nouveau calcul des dates

Comme nous l'avons vu la sémantique forte modifie les fenêtres de tir des transitions en conflits. Il faut prendre en compte ceci en modifiant le calcul des dates dans l'algorithme de VALETTE.

Alors que l'algorithme de VALETTE parlait d'estampilles temporelles associées aux atomes, nous parlerons dans cette algorithme de date de tir des transitions que nous noterons D_i (les intervalles de sensibilisation sont noté d_i). La date de tir d'une transition est la durée écoulée par rapport à une origine de temps absolue. Elle est égale aux dates de production des atomes produits par la transition et aux dates de consommation des atomes consommés ce qui permet de retrouver les estampilles temporelles de l'algorithme de VALETTE. Par ailleurs dans cet algorithme le calcul de la date au plus tôt doit être différencié de celui de la date au plus tard.

Deux cas se présentent. Pour les transitions qui ne sont pas en conflit, le calcul de leur date de tir est le même que celui effectué dans l'algorithme de VALETTE. La date de tir de ces transitions n'est en effet pas influencée par l'hypothèse de sémantique forte. Le calcul est donc le suivant : la date au plus tôt (respectivement au plus tard) du tir de la transition est égale au maximum des dates de production au plus tôt (respectivement au plus tard) des jetons consommés, augmentée de la durée minimale (respectivement maximale) de sensibilisation.

Pour les transitions appartenant à un groupe de conflit, nous calculons au préalable les dates de tir en l'absence de conflit des transitions du groupe, c'est à dire à l'aide de la méthode précédente. Afin de prendre en compte le conflit, nous modifions ensuite la date de tir de la transition que l'ont souhaite tirer, de la manière suivante :

- la date de tir au plus tôt devient le maximum entre les dates de tir au plus tôt des transitions du groupe de conflits déjà tirées et sa date de tir au plus tôt en l'absence de conflit.

- la date de tir au plus tard devient le minimum des dates de tir au plus tard (calculées en l'absence de conflit) des transitions du groupe de conflits franchissables.

Remarque

Ce calcul prend en compte l'ordre total qu'il existe entre les transitions d'un groupe de conflits. Ainsi dans le cas du conflit indirect de la figure 4.2, il peut sembler incorrect de contraindre la date de tir au plus tard de T_1 par celle de T_3 . Mais dans le cas d'un ordre total, si l'on spécifie que T_1 est avant T_3 alors nécessairement T_1 ne peut pas être tirée après la date au plus tard de T_3 . De même pour la date minimale, si T_3 est après T_1 alors nécessairement T_3 ne peut être tirée au plus tôt qu'après la date au plus tôt de T_1 .

4.2.4 Algorithme

L'algorithme 3 est l'algorithme de preuve en sémantique forte qu'a proposé S.REVOL [6].

```

Créer les groupes de conflits
Appliquer la règle  $\otimes L$ 
Tant que la règle  $\multimap L$  est applicable Faire
  Appliquer la règle  $\multimap L$  en priorité aux transitions candidates qui ne sont pas en conflit
  Si la transition n'est pas en conflit Alors
    | Appliquer le calcul classique des dates
  Sinon
    | Appliquer le nouveau calcul des dates
  Fin Si
  Terminer la preuve du séquent gauche généré en utilisant, si nécessaire, la règle  $\otimes R$ 
  Appliquer, si nécessaire, la règle  $\otimes L$  au séquent droit
Fin Tant que

```

Algorithme 3 : Algorithme de preuve sémantique forte

Le scénario étudié par cet algorithme est valide, si d'une part, on prouve le séquent à la fin de l'algorithme, et d'autre part, si les transitions franchies dans ce scénario sont effectivement franchissable. En effet on pourrait tomber sur une transition qui ne possède aucune fenêtre de tir c'est à dire telle que $D_{max} < D_{min}$.

4.3 Application sur un exemple

Nous allons étudier l'exemple de la figure 4.4, dans lequel nous nous intéressons au scénario T_2, T_3 . Cet exemple présente un conflit entre T_1 et T_2 donc bien que ne figurant pas dans le scénario la transition T_1 devra être prise en compte.

Le séquent exprimant le scénario est le suivant :

$$P_0 \otimes P_2 \otimes P_3, P_2 \otimes P_3 \multimap P_4, P_0 \multimap P_1 \vdash P_1 \otimes P_4$$

La première étape est la construction des groupes de conflits. T_1 et T_2 sont en conflit, et constituent donc le seul groupe de conflits.

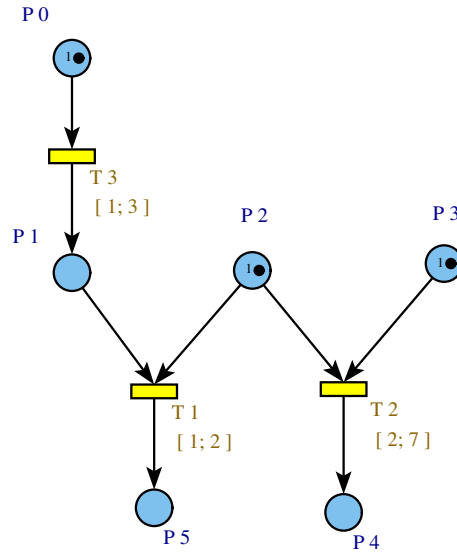


FIG. 4.4 – Exemple de réseau de Petri avec conflit

Ensuite on applique la règle \otimes_L pour déconnecter les jetons du marquage initial. Nous considérons que ces jetons sont tous présents à la date 0.

$$\frac{P_0, P_2, P_3, P_2 \otimes P_3 \multimap P_4, P_0 \multimap P_1 \vdash P_1 \otimes P_4}{P_0 \otimes P_2 \otimes P_3, P_2 \otimes P_3 \multimap P_4, P_0 \multimap P_1 \vdash P_1 \otimes P_4} \otimes_L$$

Pour appliquer la règle de tir \multimap_L deux transitions sont candidates : T_3 et T_2 . T_3 n'est pas en conflit et elle est donc prioritaire. Nous effectuons le tir de la transition :

$$\frac{\overline{P_0 \vdash P_0} \text{ id} \quad P_1, P_2, P_3, P_2 \otimes P_3 \multimap P_4 \vdash P_1 \otimes P_4}{P_0, P_2, P_3, P_2 \otimes P_3 \multimap P_4, P_0 \multimap P_1 \vdash P_1 \otimes P_4} \multimap_L$$

Nous calculons la date de tir de la transition : n'étant pas en conflit, sa date de tir est calculée par la méthode utilisée dans l'algorithme de VALETTE. Le tableau suivant permet de stocker le résultat :

Nom de la date :	D_1
Transition tirée :	T_3
Ressources consommées :	P_0
Ressources produites :	P_1
Valeur au plus tôt :	$d_{3_{min}}$
Valeur au plus tard :	$d_{3_{max}}$

Dans l'itération suivante il ne reste que T_2 comme transition tirable.

$$\frac{\overline{P_2 \vdash P_2} \text{ id} \quad \overline{P_3 \vdash P_3} \text{ id} \quad \overline{P_1 \vdash P_1} \text{ id} \quad \overline{P_4 \vdash P_4} \text{ id}}{\frac{P_2, P_3 \vdash P_2 \otimes P_3}{P_1, P_4 \vdash P_1 \otimes P_4} \otimes_R} \multimap_L$$

T_2 est en conflit avec T_1 sa date de tir est donc calculée par le nouveau calcul des dates. Tout d'abord, on précalcule sa date de tir en l'absence de conflit qui est $[d_{2_{min}}, d_{2_{max}}]$ ainsi que la date de tir au plus tard en l'absence de conflit de T_1 qui est $d_{3_{max}} + d_{1_{max}}$.

On modifie alors la date de tir de T_2 . Pour la valeur au plus tôt, aucune transition du groupe de conflits n'a déjà été franchie, la valeur n'est donc pas modifiée. Pour la valeur au plus tard, T_1 est également franchissable dans le groupe de conflits. On prend donc comme valeur au plus tard le minimum des dates aux plus tard c'est à dire :

$$\min(d_{3_{max}} + d_{1_{max}}, d_{2_{max}})$$

On obtient finalement la date suivante pour le tir de T_2 :

Nom de la date :	D_2
Transition tirée :	T_2
Ressources consommées :	P_2, P_3
Ressources produites :	P_4
Valeur au plus tôt :	$d_{2_{min}}$
Valeur au plus tard :	$\min(d_{3_{max}} + d_{1_{max}}, d_{2_{max}})$

L'arbre de preuve est correct, cependant pour que le scénario soit valide il faut également que T_2 soit franchissable dans un intervalle de temps, ce qui se vérifie par la condition :

$$d_{2_{min}} \leq \min(d_{3_{max}} + d_{1_{max}}, d_{2_{max}})$$

L'application numérique donne la fenêtre de tir suivante pour T_2 : $[2, 5]$. L'algorithme de VALETTE en sémantique faible aurait lui trouvé la valeur $[2, 7]$.

4.4 Conclusion

Ce nouvel algorithme permet de prendre en compte la sémantique forte pour calculer des dates de tir. Il prend en compte les conflits de transitions en fixant un ordre total entre ces transitions. Pour étudier l'ensemble des comportements, il est donc nécessaire d'effectuer plusieurs arbres de preuves en permutant l'ordre de tir des transitions en conflit.

Pour les conflits de jetons, il est également nécessaire d'effectuer plusieurs arbres de preuves. Cependant il faut pouvoir différencier les jetons impliqués. Un mécanisme d'étiquetage des jetons tel que proposé par N.RIVIÈRE dans [7] peut alors être utile.

Enfin, dans le cas de la sémantique forte, il est nécessaire de considérer l'ensemble du réseau de Petri et de tirer toutes les transitions qui ne sont pas en conflits. Cela peut être fastidieux et entraîner des franchissements inutiles. On pourrait ce restreindre à une sous partie du réseau de Petri. Les travaux de DEA de M.SOGBOHOSSOU [8] proposent de limiter le réseau de Petri à étudier en recherchant les dépendances entre transitions.

Cette nécessité de prendre en compte l'ensemble du réseau de Petri pose problème pour établir un scénario à étudier. Si l'on souhaite en effet étudier les comportements d'un petit ensemble de transitions, il est nécessaire de connaître tous les franchissements de transitions n'appartenant pas à cet ensemble mais nécessaires pour activer tous les conflits possibles.

Conclusion

Nous avons présenté dans ce rapport une méthode originale d'étude des réseaux de Petri T-temporels. Cette approche est basée sur la logique linéaire et elle est orientée « événements ». Elle peut être utile pour étudier les comportements possibles d'un ensemble donné de transitions. Un algorithme permet ainsi de calculer des dates de tir et des durées de scénarios de manière formelle et en prenant correctement en compte le parallélisme du réseau de Petri. On peut ainsi étudier les contraintes temporelles de l'ensemble de comportements considéré.

Cependant cette algorithme est limité à la sémantique de tir faible des réseaux de Petri T-temporels. Un nouvel algorithme permet lui de prendre en compte la sémantique forte en modifiant l'ordre de tir des transitions et le calcul des dates. Cet algorithme prend en compte les conflits de transitions mais réintroduit pour cela de l'entrelacement entre les ordres de tir.

Les perspectives de recherche sur ce sujet sont multiples. Tout d'abord, les conflits de jetons ne sont pas encore traités. Il sera de nouveau nécessaire de réintroduire des entrelacements, mais il faut tout d'abord pouvoir identifier les conflits et les jetons. Ensuite, dans les cas de conflits, il faudrait aussi pouvoir traiter les entrelacements, c'est à dire les multiples arbres de preuves, de manière plus automatique. L'analyse des résultats, afin par exemple de dégager des conditions temporelles, peut également constituer un sujet d'étude. Enfin, des recherches plus générales pourraient s'intéresser à l'utilisation d'autres connecteurs de la logique linéaire.

Annexe A

Règles du calcul des séquents en logique classique (LK)

Groupe Identité

$$\frac{}{F \vdash F} \textit{id} \qquad \frac{\Gamma \vdash F, \Delta \quad \Gamma', F \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \textit{Cut}$$

Groupe Structurel

$$\frac{\Gamma, F, G, \Gamma' \vdash \Delta}{\Gamma, G, F, \Gamma' \vdash \Delta} X_L \qquad \frac{\Gamma \vdash \Delta, F, G, \Delta'}{\Gamma \vdash \Delta, G, F, \Delta'} X_R$$

$$\frac{\Gamma, F, F \vdash \Delta}{\Gamma, F \vdash \Delta} C_L \qquad \frac{\Gamma \vdash F, F, \Delta}{\Gamma \vdash F, \Delta} C_R$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, F \vdash \Delta} W_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash F, \Delta} W_R$$

Groupe Logique

$$\frac{\Gamma \vdash F, \Delta}{\Gamma, \neg F \vdash \Delta} \neg_L \qquad \frac{\Gamma, F \vdash \Delta}{\Gamma \vdash \neg F, \Delta} \neg_R$$

$$\frac{\Gamma, F, G \vdash \Delta}{\Gamma, F \wedge G \vdash \Delta} \wedge_L \qquad \frac{\Gamma \vdash F, \Delta \quad \Gamma' \vdash G, \Delta'}{\Gamma, \Gamma' \vdash F \wedge G, \Delta, \Delta'} \wedge_R$$

$$\frac{\Gamma, F \vdash \Delta \quad \Gamma', G \vdash \Delta'}{\Gamma, \Gamma', F \vee G \vdash \Delta, \Delta'} \vee_L \qquad \frac{\Gamma \vdash F, G, \Delta}{\Gamma \vdash F \vee G, \Delta} \vee_R$$

$$\frac{\Gamma \vdash F, \Delta \quad \Gamma', G \vdash \Delta'}{\Gamma, \Gamma', F \Rightarrow G \vdash \Delta, \Delta'} \Rightarrow_L \qquad \frac{\Gamma, F \vdash G, \Delta}{\Gamma \vdash F \Rightarrow G, \Delta} \Rightarrow_R$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, \top \vdash \Delta} \top_L \qquad \frac{}{\vdash \top} \top_R$$

$$\frac{}{\perp \vdash} \perp_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \perp, \Delta} \perp_R$$

Annexe B

Règles du calcul des séquents en logique linéaire (LL)

Groupe Identité

$$\overline{\Gamma \vdash \Gamma} \textit{id} \qquad \frac{\Gamma \vdash F, \Delta \quad \Gamma', F \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \textit{cut}$$

Groupe Structurel

$$\frac{\Gamma, F, G, \Gamma' \vdash \Delta}{\Gamma, G, F, \Gamma' \vdash \Delta} X_L \qquad \frac{\Gamma \vdash \Delta, F, G, \Delta'}{\Gamma \vdash \Delta, G, F, \Delta'} X_R$$

Négation Linéaire

$$\frac{\Gamma \vdash F, \Delta}{\Gamma, F^\perp \vdash \Delta} \perp_L \qquad \frac{\Gamma, F \vdash \Delta}{\Gamma \vdash F^\perp, \Delta} \perp_R$$

Règles logiques pour les connecteurs multiplicatifs

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

$$\frac{\Gamma, F \vdash \Delta \quad \Gamma', G \vdash \Delta'}{\Gamma, \Gamma', F \wp G \vdash \Delta, \Delta'} \wp_L \qquad \frac{\Gamma \vdash F, G, \Delta}{\Gamma \vdash F \wp G, \Delta} \wp_R$$

$$\frac{\Gamma \vdash F, \Delta \quad \Gamma', G \vdash \Delta'}{\Gamma, \Gamma', F \multimap G \vdash \Delta, \Delta'} \multimap_L \qquad \frac{\Gamma, F \vdash G, \Delta}{\Gamma \vdash F \multimap G, \Delta} \multimap_R$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, \mathbf{1} \vdash \Delta} \mathbf{1}_L$$

$$\overline{\vdash \mathbf{1}} \mathbf{1}_R$$

$$\overline{\perp} \perp_L$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \perp, \Delta} \perp_R$$

Règles logiques pour les connecteurs additifs

$$\frac{\Gamma, F \vdash \Delta}{\Gamma, F \& G \vdash \Delta} \&_{L1} \quad \frac{\Gamma, G \vdash \Delta}{\Gamma, F \& G \vdash \Delta} \&_{L2} \qquad \frac{\Gamma \vdash F, \Delta \quad \Gamma \vdash G, \Delta}{\Gamma \vdash F \& G, \Delta} \&_R$$

$$\frac{\Gamma, F \vdash \Delta \quad \Gamma, G \vdash \Delta}{\Gamma, F \oplus G \vdash \Delta} \oplus_L$$

$$\frac{\Gamma \vdash F, \Delta}{\Gamma \vdash F \oplus G, \Delta} \oplus_{R1} \quad \frac{\Gamma \vdash G, \Delta}{\Gamma \vdash F \oplus G, \Delta} \oplus_{R2}$$

$$\overline{\Gamma, \mathbf{0} \vdash \Delta} \mathbf{0}_L$$

$$\overline{\Gamma \vdash \top, \Delta} \top_R$$

Règles logiques pour les connecteurs exponentiels

$$\frac{\Gamma, F \vdash \Delta}{\Gamma, !F \vdash \Delta} !_L$$

$$\frac{! \Gamma \vdash F, ? \Delta}{! \Gamma \vdash !F, ? \Delta} !_R$$

$$\frac{! \Gamma, F \vdash ? \Delta}{! \Gamma, ?F \vdash ? \Delta} ?_L$$

$$\frac{! \Gamma \vdash F, \Delta}{! \Gamma \vdash ?F, \Delta} ?_R$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, !F \vdash \Delta} W!_L$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash ?F, \Delta} W?_R$$

$$\frac{\Gamma, !F, !F \vdash \Delta}{\Gamma, !F \vdash \Delta} C!_L$$

$$\frac{\Gamma \vdash ?F, ?F, \Delta}{\Gamma \vdash ?F, \Delta} C?_R$$

Bibliographie

- [1] Jean-Yves Girard. *Linear Logic. Theoretical Computer Science*, vol. 50, 1987.
- [2] François Girault. *Formalisation en logique linéaire du fonctionnement des réseaux de Petri*. Thèse de doctorat, Université Paul Sabatier, Toulouse, Décembre 1997.
- [3] Luis Allan Künzle. *Raisonnement Temporel Basé sur les Réseaux de Petri pour des Systèmes Manipulant des Ressources*. Thèse de doctorat, Université Paul Sabatier, Toulouse, Septembre 1997.
- [4] Brigitte Pradin-Chézalviel and Robert Valette. *Accessibilité de marquage et logique linéaire dans un réseau de Petri t-temporel*. pages 123–134, Toulouse, 18-19 mai 2000. FAC'2000.
- [5] Brigitte Pradin-Chézalviel, Robert Valette, and Luis Allan Künzle. *Formalisation de scénarios, réseaux de Petri et logique linéaire*. pages 84–95, Toulouse, 25-26 février 1999. FAC'99.
- [6] Sébastien Revol. *Modélisation des réseaux de Petri temporels à l'aide de la logique linéaire*. Rapport de dea, Institut de Recherche Cybernétique de Nantes, Ecole Centrale de Nantes, 2004.
- [7] Nicolas Rivière. *Modélisation et analyse temporelle par réseaux de Petri et logique linéaire*. Thèse de doctorat, Université Paul Sabatier, Toulouse, Novembre 2003.
- [8] Médésu Sogbohossou. *Raisonnement temporel sur un réseau de Petri t-temporel à l'aide de la logique linéaire*. Rapport de dea, Institut de Recherche Cybernétique de Nantes, Ecole Centrale de Nantes, 2005.
- [9] Robert Valette, François Girault, Luis Allan Künzle, Brigitte Pradin-Chézalviel, and Janette Cardoso. *Vérification de contraintes temporelles pour des systèmes de contrôle-commande à l'aide des réseaux de Petri*. pages 255–267, Montréal, 1-4 octobre 1996. 9 Entretiens du Centre Jacques Cartier.