Année Universitaire 2005-2006

IRCCyN - Institut de Recherche en Communications et Cybernétique de Nantes

Master Recherche « Automatique et Systèmes de Productions »

Thèse de Master ASP

Stratégies d'analyse d'une fraction d'un réseau de Petri T-temporel

Louis-Marie Traonouez

Encadré par : David Delfieu

Table des matières

Table des figures

Introduction

J'ai réalisé mon master recherche dans l'équipe Systèmes Temps Réels du laboratoire IRCCyN (Institut de Recherche en Communication et Cybernétique de Nantes). Dans cette équipe le formalisme des réseaux de Petri est étudié pour modéliser et vérifier les systèmes temps réels. Ce formalisme est très adapté pour représenter des systèmes avec un fort degré de parallélisme, et sa version temporelle, les réseaux de Petri T-temporels, est capable de représenter l'aspect critique des systèmes temps réel.

Plusieurs problématiques existent autour des réseaux de Petri. L'une d'elle est d'essayer de contrer l'explosion combinatoire qui a lieu dans les réseaux complexes. Plutôt que d'étudier l'ensemble du réseau, nous proposons nous d'en étudier seulement certaines parties. Des méthodes ont déjà été développées dans cette idée. Elles utilisent un concept novateur : la logique linéaire, qui est une extension non monotone de la logique classique.

Ces méthodes se limitaient au départ à la sémantique faible des réseaux de Petri. Dans le but d'être appliquée aux systèmes temps réels il est nécessaire d'utiliser la sémantique forte. Des travaux antérieurs [12] se sont donc intéressés à la transposition de la méthode en sémantique forte. Dans l'idée d'étudier seulement une partie du réseau de Petri, d'autres travaux ont proposé une méthode d'étude des dépendances entre transitions [14].

Notre contribution de recherche est dans la poursuite de cette thématique. Nous avons travaillé particulièrement sur l'analyse arrière et l'enrichissement de scénario, ainsi que sur l'analyse symbolique des résultats fournis par la méthode de preuve en sémantique forte.

Chapitre 1

Les réseaux de Petri T-temporels

Les réseaux de Petri T-temporels sont une extension des réseaux de Petri, introduite par Merlin en 1974 [9], qui permet de prendre en compte le temps de manière quantitative. Cette prise en compte du temps est nécessaire pour modéliser des systèmes critiques tels que les systèmes temps réels. Dans ce premier chapitre, nous présenterons ce formalisme et la problématique sur laquelle nous avons travaillé.

1.1 Définitions

Il existe plusieurs manières d'associer le temps à un réseau de Petri. On peut par exemple l'associer soit aux places, soit aux transitions, soit aux arcs du réseau. Dans les réseaux de Petri T-temporels, on associe à chaque transition un intervalle de temps appelé *durée de sensibilisation*. Cet intervalle indique le temps minimum et maximum nécessaire après sensibilisation, pour que la transition soit franchissable. Cet intervalle peut éventuellement être ouvert, avec une borne maximum infinie. Formellement, un réseau de Petri T-temporel est défini ainsi :

Définition 1 (Réseau de Petri T-temporel)
Un réseau de Petri T-temporel est un n-uplet $(P,\ T,{}^\bullet(.),\ (.)^\bullet, M_0,\ I)$ *où :*

- P *est un ensemble fini de* places,
- T *est un ensemble fini de* transitions, *avec* $P \cap T = \emptyset$,
- ${}^\bullet(.) \in (\mathbb{N}^P)^T$ *est la* fonction d'incidence amont *(également notée* $Pre(.,t)$*)*,
- $(.)^\bullet \in (\mathbb{N}^P)^T$ *est la* fonction d'incidence aval *(également notée* $Post(.,t)$*)*,
- $M_0 \in \mathbb{N}^P$ *est le* marquage initial,
- $I\ :\ T \to \mathfrak{I}(\mathbb{Q}_{\geq 0})$ *est une fonction qui associe à chaque transition un intervalle de tir appelé* durée de sensibilisation.

Un marquage M du réseau est un élément de $\mathbb{N}^P$. On associe ainsi à chaque place p du réseau le nombre de jetons $M(p)$ contenus dans cette place.

Une transition t est dite *sensibilisée* par un marquage M si et seulement si $M \geq {}^\bullet t$. Le franchissement de cette transition produit le nouveau marquage $M' = M - {}^\bullet t + t^\bullet$.

Sémantique des réseaux de Petri T-temporels

Il existe plusieurs sémantiques qui décrivent le fonctionnement des réseaux de Petri T-temporels. Elles peuvent être classées dans deux catégories : les sémantiques fortes et les sémantiques faibles.

Les différences entre ces deux types de sémantiques et les problématiques qu'elles entraînent sont étudiées dans la section suivante. Dans ce rapport, nous nous plaçons dans le cadre de la sémantique forte, qui est celle qui présente le plus d'intérêt vis à vis des systèmes temps réel, car elle est la seule à pouvoir exprimer l'urgence d'un évènement.

Toutefois, même au sein de la sémantique forte, plusieurs sémantiques sont possibles. Le lecteur intéressé trouvera dans [2] une comparaison des différentes sémantiques. Nous discuterons dans la partie 2.5 de la compatibilité de notre approche vis à vis de ces sémantiques. Ci-dessous, nous décrivons la sémantique la plus répandue dans les réseaux de Petri T-temporels, telle que décrite dans [8].

Dans cette sémantique, une horloge $\nu(t)$ est associée aux transitions t du réseau. Cette horloge sera comparée avec la durée de sensibilisation de la transition pour savoir si le tir de cette transition est autorisé. Nous noterons $\nu(t) \in I(t)^{\downarrow}$ pour signifier que $\nu(t)$ est inférieur à la borne maximale de de $I(t)$.

L'ensemble des transitions sensibilisées par un marquage M est noté $enabled(M)$. Si une transition t' est tirable à partir du marquage M, une transition t est dite *nouvellement sensibilisée* par le tir de t', ce que l'on note $\downarrow enabled(t,M,t')$, si t est sensibilisée par le nouveau marquage $M - {}^\bullet t' + t'^\bullet$, mais ne l'était pas par le marquage $M - {}^\bullet t'$. Formellement,

$$\downarrow enabled(t,M,t') = (t \in enabled(M - {}^\bullet t' + t'^\bullet)) \wedge (t \notin enabled(M - {}^\bullet t') \vee (t = t')).$$

Définition 2 (Sémantique d'un réseau de Petri T-temporel)
La sémantique d'un réseau de Petri T-temporel $\mathfrak{T}$ est définie sous la forme d'un système de transition temporisé $S_{\mathfrak{T}} = (Q, q_0, \rightarrow)$ tel que :

- $Q = \mathbb{N}^P \times (\mathbb{R}^+)^T$,
- $q_0 = (M_0, 0)$,
- $\rightarrow \in Q \times (T \cup \mathbb{R}^+) \times Q$ *est la relation de transition qui inclus des transitions continues et des transitions discrètes :*
  - *la relation de transition continue est définie* $\forall d \in \mathbb{R}^+$ *par :*

$$(M,\nu) \xrightarrow{d} (M',\nu') \quad ssi \quad \begin{cases} \nu' = \nu + d, \\ \forall t \in enabled(M),\ \nu'(t) \in I(t)^{\downarrow}. \end{cases}$$

  - *la relation de transition discrète est définie* $\forall t \in T$ *par :*

$$(M,\nu) \xrightarrow{t} (M',\nu') \quad ssi \quad \begin{cases} t \in enabled(M), \\ M' = M - {}^\bullet t + t^\bullet, \\ \nu(t) \in I(t), \\ \forall t',\ \nu'(t') = \begin{cases} 0 \text{ si } \downarrow enabled(t',M,t), \\ \nu'(t') \text{ sinon.} \end{cases} \end{cases}$$

1.2 Sémantique forte et sémantique faible

1.2.1 Différences entre les deux types de sémantique

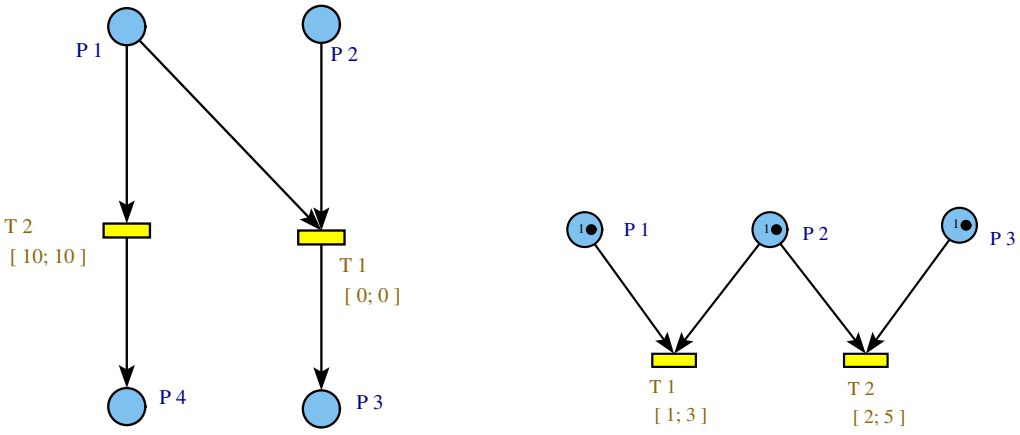
La sémantique des réseaux de Petri T-temporels présentée précédemment est une sémantique de tir forte (STFO). En effet, dans cette sémantique, on force le tir d'une transition si son horloge a atteint la valeur maximale de la durée de sensibilisation. Ceci se traduit, en terme de système de transitions, par l'impossibilité de faire évoluer le temps à l'aide d'une transition continue, si une des horloges atteint sa valeur maximale.

Au contraire, dans une sémantique de tir faible (STFA), on n'est pas obligé de tirer une transition dont l'horloge a atteint sa valeur maximale. Toutefois, une fois cette valeur dépassée, si la transition n'a pas été tirée, elle ne pourra plus l'être avant d'être resensibilisée. Refuser de tirer une transition n'a d'intérêt qu'en cas de conflits de transitions. Cela permet de réserver des jetons en conflit pour le tir d'une autre transition.

Ainsi en sémantique faible, les valeurs temporelles ne modifient pas les résultats d'accessibilité du réseau de Petri non temporisé. En effet, pour franchir une séquence de transitions, on peut réserver tous les jetons nécessaires au franchissement, même si ces jetons sont en conflits avec d'autres transitions. De plus, en sémantique faible, si l'on considère une séquence franchissable, et si l'on rajoute des jetons dans le réseau, cette séquence restera franchissable, même si ces jetons sensibilisent des transitions en conflit. Il y a dans ce sens monotonie du réseau.

Dans le cadre des systèmes temps réels, la sémantique forte a beaucoup d'intérêt. Elle permet en effet de modéliser l'urgence de certains évènements. Par exemple, le mécanisme de *chien de garde* présenté sur la figure 1.1(a), n'est modélisable qu'en sémantique forte. Ce mécanisme permet de créer une alarme de la manière suivante :

- le chien est armé lorsqu'un jeton arrive en P1, cela déclenche l'horloge de la transition T2 qui est l'alarme ;
- si un jeton arrive en P2 avant la fin de l'alarme, le tir de T1 est forcé, le chien est désarmé, cela peut correspondre au scénario normal ;
- par contre, si l'horloge de T2 atteint sa valeur limite de 10 avant l'arrivée du jeton en P2, c'est le tir de T2 qui sera forcé, cela peut correspondre à un scénario dégradé.



(a) Chien de garde (b) Conflit simple

FIG. 1.1 – Situations de conflits dans un réseau de Petri T-temporel

En sémantique faible, même si un jeton arrive en P1 avant la fin de l'alarme, on peut toujours choisir de tirer T2 au lieu de T1. Le rajout d'un jeton en P2 n'a donc pas d'influence sur l'accessibilité de P4, c'est ce qu'on appelle monotonie.

1.2.2 Logique linéaire et sémantique forte

Nous ne rappellerons pas dans ce rapport les concepts de la logique linéaire. Ils pourront être trouvés dans le rapport bibliographique du master, et dans ces publications [5, 6, 7, 13, 11, 10, 15, 3, 12, 14]. Quelques rappels sur la traduction des réseaux de Petri en logique linéaire seront donnés au chapitre suivant.

Les travaux d'applications de la logique linéaire aux réseaux de Petri T-temporels [6, 7, 13, 11, 10, 15] qui ont été menés au Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) de Toulouse par R. VALETTE et son équipe, ce sont jusqu'à aujourd'hui cantonné à la sémantique faible.

Ceci, car la logique linéaire n'est pas conforme à la sémantique forte. En effet, en logique linéaire si l'on rajoute des ressources de contexte dans un séquent prouvable, c'est à dire des ressources non utilisées que l'on rajoute dans les prémisses et dans la conclusion, ce séquent sera toujours prouvable. Ceci correspond parfaitement à la sémantique faible : rajouter des ressources dans un séquent est équivalent à rajouter des jetons dans le réseau ; il y a donc en logique linéaire la même monotonie que dans les réseaux de Petri T-temporels en sémantique faible.

Si l'on considère l'exemple du chien de garde sur la figure 1.1(a), le scénario d'alarme est exprimé par le séquent prouvable suivant : $P_1,\ T_2 \vdash P_4$. Mais le séquent $P_1,\ P_2\ , T_2 \vdash P_4 \otimes P_2$ est également prouvable, ce qui n'est pas toujours vrai en sémantique forte.

Cette propriété de monotonie est préjudiciable dans le cadre de la sémantique forte. Le calcul des dates symboliques effectué par l'algorithme de Valette, n'est plus valable en sémantique forte. Dans l'exemple de la figure 1.1(b), la date de tir de T2 calculée par l'algorithme de Valette est $[d_{2min}, d_{2max}] = [2,5]$. Or en sémantique forte, le tir de T2 n'est plus possible au delà de 3, car à cet instant le tir de T1 devient obligatoire. La valeur correcte pour la date de tir de T2 est donc $[2,3]$.

Les travaux menés à l'IRCCyN sous la direction de D. DELFIEU [12, 14], ont toutefois permis, en modifiant l'algorithme de Valette, de prendre en compte la sémantique forte dans l'approche des réseaux de Petri par la logique linéaire. Nous verrons au chapitre suivant comment cela est effectué.

1.3 Problématique et approche proposée

Pour effectuer des analyses sur un réseau de Petri, il est nécessaire au préalable de calculer le graphe d'état du réseau. Dans le cas de réseaux importants, on doit faire face dans ce calcul à une explosion combinatoire. Ce graphe d'état devient même infini si l'on prend en compte le temps. Dans le cadre des réseaux de Petri T-temporels, on regroupe les états afin d'obtenir un graphe fini. Une méthode classique de regroupement est le graphe des classes d'états [1]. Pour des réseaux importants, on y retrouve la même explosion combinatoire. Il existe des méthodes pour gérer cette explosion combinatoire. Dans [4] les auteurs propose la méthode des zones, qui se distingue du graphe des classes par sa manière de regrouper les états. L'explosion combinatoire ne peut cependant être contenue que jusqu'à un certain point.

Notre approche est différente. Elle consiste à effectuer des analyses du réseau sans calculer l'ensemble du graphe d'état. Pour cela, nous nous limitons à l'analyse d'une partie du réseau. Nous travaillons en effet sur un ensemble de transitions, qui délimitent une partie du réseau. Cette approche est donc basée sur les évènements, plutôt que sur les états. Elle utilise la logique linéaire pour représenter le réseau de Petri. Cela permet de travailler avec un parallélisme vrai, ce qui réduit d'autant plus l'explosion combinatoire, car on évite ainsi d'effectuer des entrelacements pour représenter le parallélisme.

Nous allons présenter dans ce rapport les différentes stratégies d'analyses que nous avons développées.

Chapitre 2

Analyse temporelle, en sémantique forte, d'un scénario exprimé en logique linéaire

Dans ce chapitre nous présentons une méthode qui permet d'analyser temporellement, avec des valeurs symboliques, un scénario de tir d'un réseau de Petri T-temporel. Cette méthode est l'adaptation à la sémantique de forte de la méthode développée au LAAS [11]. Elle utilise le mécanisme de preuves des séquents de logique linéaire.

2.1 Notion de scénario

Un scénario de tir dans un réseau de Petri est un ensemble non-ordonné de transitions, où chaque transition peut éventuellement apparaître plusieurs fois.

Définition 3 *On appelle* ***multi-ensemble*** ξ *sur un ensemble fini* X *la donnée d'une fonction* $1_\xi : X \to \mathbb{N}$ *appelée* fonction caractéristique *de* ξ*. On dit que* $x \in X$ *est* élément *de* ξ *si et seulement si* $1_\xi(x) \geq 1$*, l'entier* $1_\xi(x)$ *est la* multiplicité *de* x *dans* ξ*. On appelle* constituant *de* ξ *toute occurrence d'un de ses éléments.*

Un scénario est donc un multi-ensemble de l'ensemble T des transitions du réseau.

Ainsi, en travaillant sur des scénarios nous considérons l'ensemble des séquences de transitions qui peuvent être constituées à l'aide des éléments de ce scénario. Sont donc notamment comprises dans un scénario, toutes les séquences qui ne diffèrent que par l'entrelacement de leurs transitions parallèles. Dans un scénario, nous prenons donc en compte le parallélisme des réseaux de Petri sans devoir effectuer des entrelacements pour l'exprimer. La méthode développée permet de traiter directement un scénario, et donc toutes les séquences de transitions qu'il représente, aussi rapidement qu'une seule séquence.

Lorsque nous nous intéressons à un scénario, nous délimitons une partie du réseau de Petri. Cette partie est constituée des transitions du scénario et des places utilisées dans ces transitions. Donc lorsque nous analysons temporellement un scénario, nous étudions la partie du réseau qu'il délimite. Ainsi, la méthode d'analyse temporelle de scénario qui va être présentée dans ce chapitre, constitue une stratégie d'analyse d'une fraction d'un réseau de Petri.

2.2 Modélisation des réseaux de Petri en logique linéaire

Dans ce chapitre, nous allons brièvement rappeler comment sont modélisés les réseaux de Petri en logique linéaire (LL). Nous présenterons également l'algorithme de preuve en sémantique faible (algorithme de Valette). Pour plus de détails, le lecteur intéressé pourra se reporter de nouveau sur le rapport bibliographique ou sur ces publications [5, 6, 7, 13, 11, 10, 15, 3, 12, 14].

Pour illustrer cette modélisation, nous considérons l'exemple de la figure 2.1.

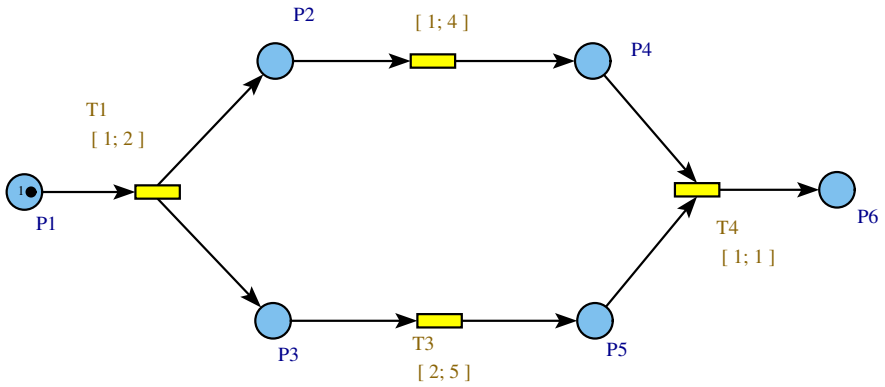


FIG. 2.1 – Réseau de Petri T-temporel

2.2.1 Structure du réseau

A chaque place p du réseau, on associe une ressource de logique linéaire P. Un marquage m du réseau de Petri est traduit par une conjonction multiplicative de ressources, c'est à dire une formule M de logique linéaire du type :

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

où $m(p)$ est la multiplicité de la place p dans le marquage m, ce qui correspond au nombre de jetons contenus dans cette place.

Les transitions du réseau de Petri sont traduites par une formule implicative, avec comme prémisse le marquage de pré-condition de la transition, et comme conclusion le marquage de post-condition. C'est à dire, pour une transition t, une formule T du type :

$$T = \bigotimes_{i \in Pre(p_i,t)} P_i \multimap \bigotimes_{o \in Post(p_o,t)} P_o$$

Ainsi, l'exemple de la figure 2.1 est traduit par les formules de logique linéaire suivantes :

$$\begin{aligned} M_0 &= P_1 \\ T_1 &= P_1 \multimap P_2 \otimes P_3 \\ T_2 &= P_2 \multimap P_4 \\ T_3 &= P_3 \multimap P_5 \\ T_4 &= P_4 \otimes P_5 \multimap P_6 \end{aligned}$$

2.2.2 Fonctionnement du réseau

Le franchissement d'une transition t à partir d'un marquage m pour donner le nouveau marquage m' est exprimé par le séquent de logique linéaire 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 $t_1;t_2;\ldots;t_n$ est exprimé de manière similaire par le séquent :

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

Dans [6], 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. Ce séquent ne représente pas la seule séquence de transitions t_1 ; t_2 ; … ; t_n mais le scénario constitué par ces transitions.

2.2.3 Algorithme de preuve avec estampilles temporelles

L'Algorithme 1 a été proposé par R. VALETTE et son équipe au LAAS. Il permet de réaliser une preuve canonique d'un séquent exprimant un scénario de tir dans un réseau de Petri. La réalisation de cette preuve permet de prouver l'accessibilité du marquage final à partir du marquage initial.

Cet algorithme intègre également des informations temporelles permettant de calculer la durée d'un scénario dans un réseau de Petri T-temporels. Pour cela, chaque atome est doté d'une estampille temporelle indiquant sa date de production. Au cours de la preuve, on ajoute les estampilles des atomes consommés et les durées de sensibilisations des transitions, afin de calculer les estampilles des atomes produits. La durée du scénario est l'estampille temporelle du dernier atome produit. Ces estampilles temporelles peuvent être calculées de manière symbolique ce qui rajoute de l'intérêt à l'approche.

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 1 : Algorithme de preuve avec estampilles temporelles

Cependant, cet algorithme ne fonctionne qu'en sémantique faible. Nous avons en effet évoqué dans la partie 1.2 les difficultés d'utilisation de la logique linéaire dans le cadre de la sémantique forte des réseaux de Petri T-temporels. Cet algorithme se limite donc à la sémantique faible en vérifiant au préalable que le scénario est complètement spécifié, ce qui correspond à *Vérifier que le fragment du réseau correspondant au scénario est un graphe d'évènements*. On élimine avec cette vérification

les situations de conflits (entre jetons ou entre transitions) qui modifient les résultats dans le cas de la sémantique forte.

2.3 Algorithme de preuve en sémantique forte

Afin de prendre en compte la sémantique forte, S. REVOL et D. DELFIEU ont proposé une version modifiée de l'algorithme de Valette.

2.3.1 Repérage des transitions en conflit

Lorsque l'on passe de la sémantique faible à la sémantique forte, seules les transitions qui sont en conflits peuvent avoir leur date de tir modifiée. En effet, le tir forcé de la transition en conflit peut empêcher le tir d'une transition au-delà d'une certaine date. Il est donc nécessaire d'effectuer un repérage de ces transitions.

Cependant, deux transitions, même si elles ne sont pas en conflit entre elles, mais par exemple toutes les deux en conflit avec une troisième transition, peuvent avoir de l'influence l'une sur l'autre. On dira qu'elles sont en **conflit indirect**.

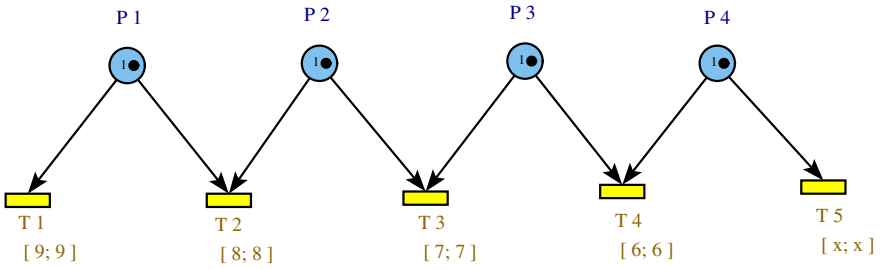


FIG. 2.2 – Conflit indirect entre transitions

Considérons par exemple le réseau de la figure 2.2. Si l'on s'intéresse à T1 : elle est en conflit avec T2, mais si T3 est tirée avant T2 ce conflit n'est plus effectif, et si T4 est tirée avant T3, T3 ne sera plus tirable donc T2 le sera à nouveau ... Ainsi, dans cet exemple même la transition T5 peut avoir de l'influence sur le tir de T1. En effet, pour $x < 6$, T1 sera tirable. Pour $x > 6$, T1 ne le sera pas.

Pour prendre en compte ses influences particulières entre les transitions, nous allons les regrouper dans des groupes de conflit. Un groupe de conflit regroupe toutes les transitions qui sont en conflit entre elles (conflit direct ou indirect). Ils sont construits par transitivité suivant la règle suivante : *deux transitions en conflit sont dans un même groupe.*

2.3.2 Modification de l'ordre de la preuve

Nous avons dit que dans un scénario les transitions n'étaient pas ordonnées. De même dans un séquent, les ressources d'actions correspondant aux transitions ne sont pas ordonnées (la virgule est commutative). Pour prendre en compte la sémantique forte, nous allons cependant considérer que les transitions qui sont en conflit (direct ou indirect) sont ordonnées. En effet, selon l'ordre de tir de ces transitions les résultats temporels obtenus pourront être différents. Pour prendre en compte l'ensemble des séquences de transitions, une seule preuve ne suffira pas. Il sera nécessaire de permuter

les transitions au sein de chaque groupe de conflit (voir le chapitre 5). On est donc dans ce cas obligé de réintroduire des entrelacements.

Par ailleurs, pour que les résultats soient les plus complets possibles, il faut faire en sorte que les conflits soient effectifs, sans quoi ils ne seraient pas pris en compte. Pour cela, il faut tirer en priorité les transitions qui ne sont pas en conflit. Dans la construction du scénario, il faudra également s'assurer que les transitions nécessaires pour rendre les conflits effectifs figurent dans le scénario et en nombre suffisant.

Ainsi, le nouvel ordre de tir est le suivant :

- 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.

2.3.3 Nouveau calcul des estampilles temporelles

Pour prendre en compte la sémantique forte, il faut modifier le calcul des estampilles temporelles (ou date de tir des transitions) de l'algorithme de Valette. En effet, en cas de conflit direct, la date de tir maximale d'une transition est limitée par la date de tir maximale de la transition en conflit. On s'aperçoit donc qu'il va falloir distinguer le calcul la date de tir maximale, de celui de la date de tir minimale.

De plus, il faut prendre en compte les interactions des conflits indirects. Pour cela, on fixe un ordre total entre les transitions d'un même groupe de conflits. Cet ordre total va nécessiter la modification de la date de tir minimale.

Le calcul des dates de tir devient donc le suivant :

- 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. Cela permet de prendre en compte l'ordre total qu'il existe entre ces transitions : une transition ne peut pas être tirée plus tôt que celles qui ont été tirées avant elle.
- 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. Ceci permet de prendre en compte, à la fois les conflits directs : la date maximum de tir est limité par celle de la transition en conflit, et l'ordre total en cas de conflit indirect : un transition ne peut pas être tirée plus tard que celles qui doivent être tirées après elle.

2.3.4 Algorithme de preuve

L'algorithme 2 est l'algorithme de preuve en sémantique forte qu'a proposé S. REVOL [12].

2.4 Application sur un cas pratique

Nous allons appliquer cette méthode sur un cas pratique. Le réseau de Petri à étudier est présenté sur la figure 2.3, il est extrait de la thèse de D. LIME [8]. Il modélise l'exécution de tâches qui s'exécutent en parallèles et qui communiquent entre elles.

Il y a en tout cinq tâches, qui sont réinitialisées par une horloge de manière périodique. La période est ici fixée à 20 unités de temps. On remarque également dans le système une ressource critique qui est le bus de communication CAN que deux des tâches utilisent. Cette ressource critique est modélisée par la place initialement marqué `bus CAN state`, qui est au centre d'un conflit entre deux transitions :

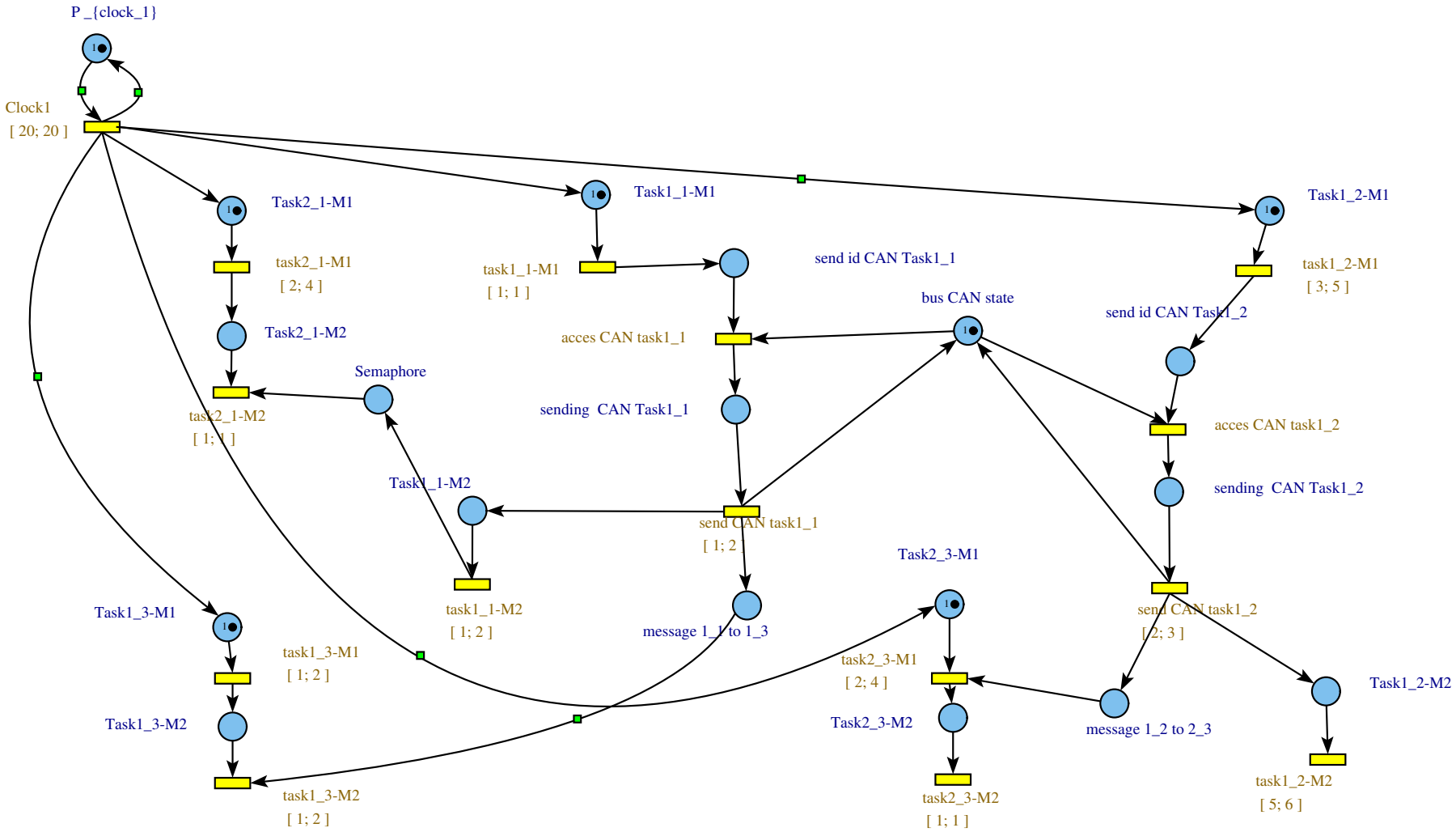


FIG. 2.3 – Étude de cas

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 2 : Algorithme de preuve en sémantique forte

`acces CAN task 1_1` et `acces CAN task 1_2`.

Nous nous proposons d'étudier la durée d'exécution de l'ensemble des tâches. Cela nous permettra de vérifier que la période de l'horloge n'est pas inférieure à la durée totale d'exécution. Pour cela nous ne considérons pas le réseau en entier. Nous allons considérons uniquement les tâches, c'est à dire supprimer l'horloge. Pour faire les analyses nous utilisons le logiciel LLBOX, implémenté par F. FRIZON DE LAMOTTE [3], conçu pour réaliser des preuves de scénario, et qui possède des plugins adaptés à la sémantique forte.
Tout d'abord, nous chargeons le réseau qui a été édité à l'aide du logiciel Romeo, un logiciel qui permet de faire des vérifications de réseau de Petri en utilisant le graphe des classes ou la méthode des zones.

```
LLBOX> romeo Lexemple.xml
```

Nous déterminons ensuite les groupes de conflit du réseau, et nous les affichons :

```
LLBOX> make_groups
LLBOX> print GroupsPrinter
Group 1
id      nom     formule
--------------------------------
9       accesCANtask1_1 busCANstate * sendidCANTask1_1 --o sendingCANTask1_1
16      accesCANtask1_2 busCANstate * sendidCANTask1_2 --o sendingCANTask1_2
```

Comme prévu, nous trouvons le conflit entre les deux transitions qui utilisent la ressource critique `busCANstate`.
Nous pouvons alors lancer la preuve du séquent et nous affichons les résultats, c'est à dire les dates de tir des transitions :

```
LLBOX> prove StrongProver
LLBOX> print StrongDatesPrinter
*******************************************************
Date: D1        Tir de: task2_1-M1
Date min: D0+d(task2_1-M1)
```

```
Date max: D0+d(task2_1-M1)
**********************************************************
Date: D2        Tir de: task1_1-M1
Date min: D0+d(task1_1-M1)
Date max: D0+d(task1_1-M1)
**********************************************************
Date: D3        Tir de: task1_2-M1
Date min: D0+d(task1_2-M1)
Date max: D0+d(task1_2-M1)
**********************************************************
Date: D4        Tir de: task1_3-M1
Date min: D0+d(task1_3-M1)
Date max: D0+d(task1_3-M1)
**********************************************************
Date: D5        Tir de: accesCANtask1_1
Date min: max(D0,D2)+d(accesCANtask1_1)
Date max: min(max(D0,D2)+d(accesCANtask1_1),max(D0,D3)+d(accesCANtask1_2))
**********************************************************
Date: D6        Tir de: sendCANtask1_1
Date min: D5+d(sendCANtask1_1)
Date max: D5+d(sendCANtask1_1)
**********************************************************
Date: D7        Tir de: task1_1-M2
Date min: D6+d(task1_1-M2)
Date max: D6+d(task1_1-M2)
**********************************************************
Date: D8        Tir de: task2_1-M2
Date min: max(D1,D7)+d(task2_1-M2)
Date max: max(D1,D7)+d(task2_1-M2)
**********************************************************
Date: D9        Tir de: task1_3-M2
Date min: max(D4,D6)+d(task1_3-M2)
Date max: max(D4,D6)+d(task1_3-M2)
**********************************************************
Date: D10       Tir de: accesCANtask1_2
Date min: max(max(D0,D2)+d(accesCANtask1_1),max(D6,D3)+d(accesCANtask1_2))
Date max: max(D6,D3)+d(accesCANtask1_2)
**********************************************************
Date: D11       Tir de: sendCANtask1_2
Date min: D10+d(sendCANtask1_2)
Date max: D10+d(sendCANtask1_2)
**********************************************************
Date: D12       Tir de: task1_2-M2
Date min: D11+d(task1_2-M2)
Date max: D11+d(task1_2-M2)
**********************************************************
Date: D13       Tir de: task2_3-M1
```

```
Date min: max(D0,D11)+d(task2_3-M1)
Date max: max(D0,D11)+d(task2_3-M1)
*****************************************************
Date: D14       Tir de: task2_3-M2
Date min: D13+d(task2_3-M2)
Date max: D13+d(task2_3-M2)
```

Les dates sont ici en symbolique et exprimées de manière récursive en fonction des dates précédentes. Nous nous apercevons ici d'un défaut de LLBOX : il ne permet pas de calculer les valeurs numériques des dates. Nous devons donc faire le calcul à la main.

Voici les valeurs numériques des intervalles de tir :

$$\begin{array}{ll} D1 \in [2,4] & D8 \in [4,6] \\ D2 \in [1,1] & D9 \in [3,5] \\ D3 \in [3,5] & D10 \in [3,5] \\ D4 \in [1,2] & D11 \in [5,8] \\ D5 \in [1,1] & D12 \in [6,14] \\ D6 \in [2,3] & D13 \in [7,12] \\ D7 \in [3,5] & D14 \in [8,13] \end{array}$$

On en déduit alors la durée totale du scénario qui est le maximum entre toutes ces dates, ce qui nous donne $[8,14]$. La durée maximum est donc de 14, ce qui est bien inférieur à la période de l'horloge qui est de 20. Cependant, ce scénario ne comporte pas nécessairement l'ensemble des exécutions possibles. En effet, nous fixons dans ce scénario un ordre total entre les deux transitions en conflit. Pour des résultats complets, il faut étudier l'ordre inverse, ce que nous verrons au chapitre 5

2.5 Compatibilité de la méthode avec les sémantiques des réseaux de Petri T-temporels

Dans la partie 1.1, nous avons présenté une sémantique des réseaux de Petri T-temporels. Cette sémantique est appelée *sémantique intermédiaire* (I) et est celle qui est couramment utilisée. Dans [2], les auteurs présentent deux autres sémantiques des réseaux de Petri T-temporels : la *sémantique atomique* (A) et la *sémantique atomique persistante* (PA). Ces trois sémantiques se distinguent uniquement par le calcul du prédicat $\downarrow enabled(t,M,t')$ qui détermine les transitions nouvellement sensibilisées après un franchissement.

Dans I :

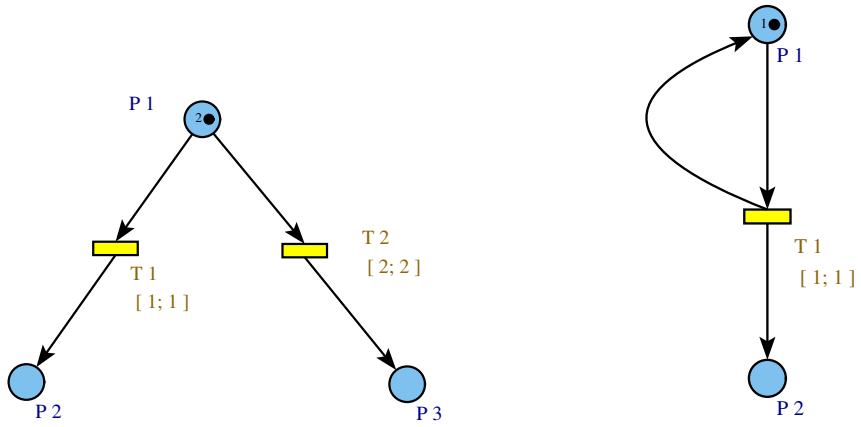
$$\downarrow enabled(t,M,t') = (t \in enabled(M - {}^\bullet t' + t'^\bullet)) \wedge (t \notin enabled(M - {}^\bullet t') \vee (t = t')).$$

Dans A :

$$\downarrow enabled(t,M,t') = (t \in enabled(M - {}^\bullet t' + t'^\bullet)) \wedge (t \notin enabled(M) \vee (t = t')).$$

Dans PA :

$$\downarrow enabled(t,M,t') = (t \in enabled(M - {}^\bullet t' + t'^\bullet)) \wedge (t \notin enabled(M)).$$



(a) (b)

FIG. 2.4 – Situations de posant des problèmes de sémantique

Dans notre approche, si l'on considère l'exemple de la figure 2.4(a), la transition T2 n'est pas tirable. En effet, si l'on considère le scénario $P_1 \otimes P_1,\ T_2 \vdash P_1 \otimes P_3$, la date de tir calculée pour T2 sera :

$$D2_{min} = d2_{min} = 2,$$

$$D2_{max} = min(d2_{max}, d1_{max}) = min(2,1) = 1.$$

La valeur maximum étant inférieure à la valeur minimum, on en déduit que T2 n'est pas tirable. Si l'on considère ensuite le scénario $P_1 \otimes P_1,\ T_1,\ T_2 \vdash P_2 \otimes P_3$, nous obtenons un résultat identique car la date de production du jeton P1 utilisé pour le tir de T2 n'est pas modifiée par le tir de T1.

Dans les sémantiques *I* et *A* au contraire, la transition T2 peut être franchie après T1. En effet, dans ces deux sémantiques T1 est considérée comme nouvellement sensibilisée par son propre tir, à cause de l'expression $(t = t')$. En conséquence, l'horloge de T1 est réinitialisée après son franchissement. Celle de T2 par contre garde sa valeur qui est à cet instant égale à 1. Ainsi, au bout d'une unité de temps supplémentaire, l'horloge de T1 sera à 1 et celle de T2 à 2. T2 pourra donc être franchie.

La sémantique *PA* par contre, ne permet pas le tir de T2, car on a supprimé l'expression $(t = t')$. En effet, après son franchissement, l'horloge de T1 n'est pas réinitialisée. Elle garde donc sa valeur égale à 1, et ainsi le deuxième jeton en P1 doit également être utilisé pour franchir T1.

On pourrait alors penser que notre approche satisfait la sémantique *PA*. Cependant, l'exemple de la figure 2.4(b) nous prouve le contraire. En effet, dans cet exemple, si l'on considère le scénario $P_1,\ T_1,\ T_1 \vdash P_1 \otimes P_2 \otimes P_2$, la date du deuxième tir de T_1 est égale à 2, car le jeton P_1 consommé est produit à la date 1.

Dans la sémantique *PA*, le deuxième tir de T_1 intervient immédiatement après le premier, c'est à dire en 1, car l'horloge de la transition n'est pas réinitialisée. Cette fois c'est l'expression $t \notin enabled(M)$ qui est en cause. L'expression $t \notin enabled(M - {}^\bullet t')$ serait plus approprié à notre approche. Elle signifie en effet que, pour qu'une transition soit nouvellement sensibilisée, il faut qu'elle le soit par des nouveaux jetons.

Cette incompatibilité avec ces sémantiques des réseaux de Petri T-temporels est en fait due à la façon de prendre en compte le temps dans notre approche. Nous ne considérons pas des horloges associées aux transitions, mais uniquement les dates de production des jetons. Notre approche correspondrait donc plus à l'association d'horloges aux jetons. Des sémantiques permettant une multi-sensibilisation des transitions pourraient également être utilisable.

Une nouvelle sémantique ?

A l'aide des reflexions faîtes ci-dessus, nous pouvons également construire une sémantique :

$$\downarrow enabled(t,M,t') = (t \in enabled(M - {}^\bullet t' + t'^\bullet)) \wedge (t \notin enabled(M - {}^\bullet t')).$$

Cette sémantique pourrait s'appeler *PI* car la même modification a lieu entre elle et *I*, et entre *PA* et *A*. Nous gardons de *I*, l'expression $t \notin enabled(M - {}^\bullet t')$, dont on a dit qu'elle nous arrangeait, et nous prenons de *PA*, l'idée de supprimer l'expression $(t = t')$.

Cette sémantique permet à nos résultats d'être valables pour les deux réseaux de la figure 2.4 (en effet, sur la figure 2.4(a) l'horloge de T_1 n'est pas réinitialisée donc T_2 n'est pas tirable, et sur la figure 2.4(b) l'horloge de T_1 est réinitialisée donc il nécessaire d'attendre pour tirer T_1 à nouveau).

Chapitre 3

Étude des dépendances entre transitions

Dans ce chapitre nous présentons une stratégie d'analyse proposée par M. SOGBOHOSSOU et D. DELFIEU [14]. Elle s'intéresse à l'étude des dépendances entre deux transitions non consécutives du réseau.

3.1 Définitions

3.1.1 Notion de dépendance entre transitions

Notre but est d'étudier les dépendances logiques entre transitions, c'est à dire les asservissements possibles entre le franchissement d'une transition amont, et le franchissement d'une transition aval.

Nous pouvons définir trois types de dépendances :

- *la dépendance forte* : le franchissement de la transition amont entraîne nécessairement le franchissement ultérieur de la transition aval, et réciproquement, le franchissement de la transition aval nécessite obligatoirement le franchissement au préalable de la transition amont.
  Sur la figure 3.1, ce type de dépendance est illustré sur les réseaux -a- et -b- entre les transitions t_i et t_f, et sur le réseau -c- entre les transitions t_i et t_f ou t_i et t_g,
- *la dépendance faible* : il existe au moins un chemin menant de la transition amont t_i, à la transition aval t_f. Cependant, cette dépendance n'est pas forte car il existe, soit des chemins en provenance de t_i qui n'aboutissent pas à t_f (réseau -d- sur la figure 3.1), soit des chemins aboutissant à t_f mais qui ne proviennent pas de t_i (réseau -e- sur la figure 3.1).
- *la non dépendance* : il n'existe pas de chemin entre la transition amont et la transition aval. C'est la cas par exemple sur le réseau -f- de la figure 3.1.

Globalement, la dépendance forte indique que deux transitions sont en série, alors que la non dépendance indique au contraire qu'elles sont en parallèle. La dépendance faible signifie qu'il existe entre ces deux transitions, soit des conflits de transitions, soit des conflits de jetons.

La dépendance forte et la non dépendance sont les plus intéressantes. Elles expriment des idées intéressantes, celle d'un ordre total entre les deux transitions, ou celle d'absence d'ordre entre elles. La dépendance faible est moins utile car elle rassemble des exemples variés.

Dans le cadre des réseaux de Petri T-temporels, ces dépendances structurelles peuvent être modifiées. En effet, avec la sémantique forte, l'indéterminisme de certains conflits de transitions peut être supprimé. Ainsi, en cas de dépendance faible, des scénarios qui n'aboutissaient pas à la transition aval peuvent devenir invalides. La dépendance peut donc devenir forte. Au contraire, toujours en cas de

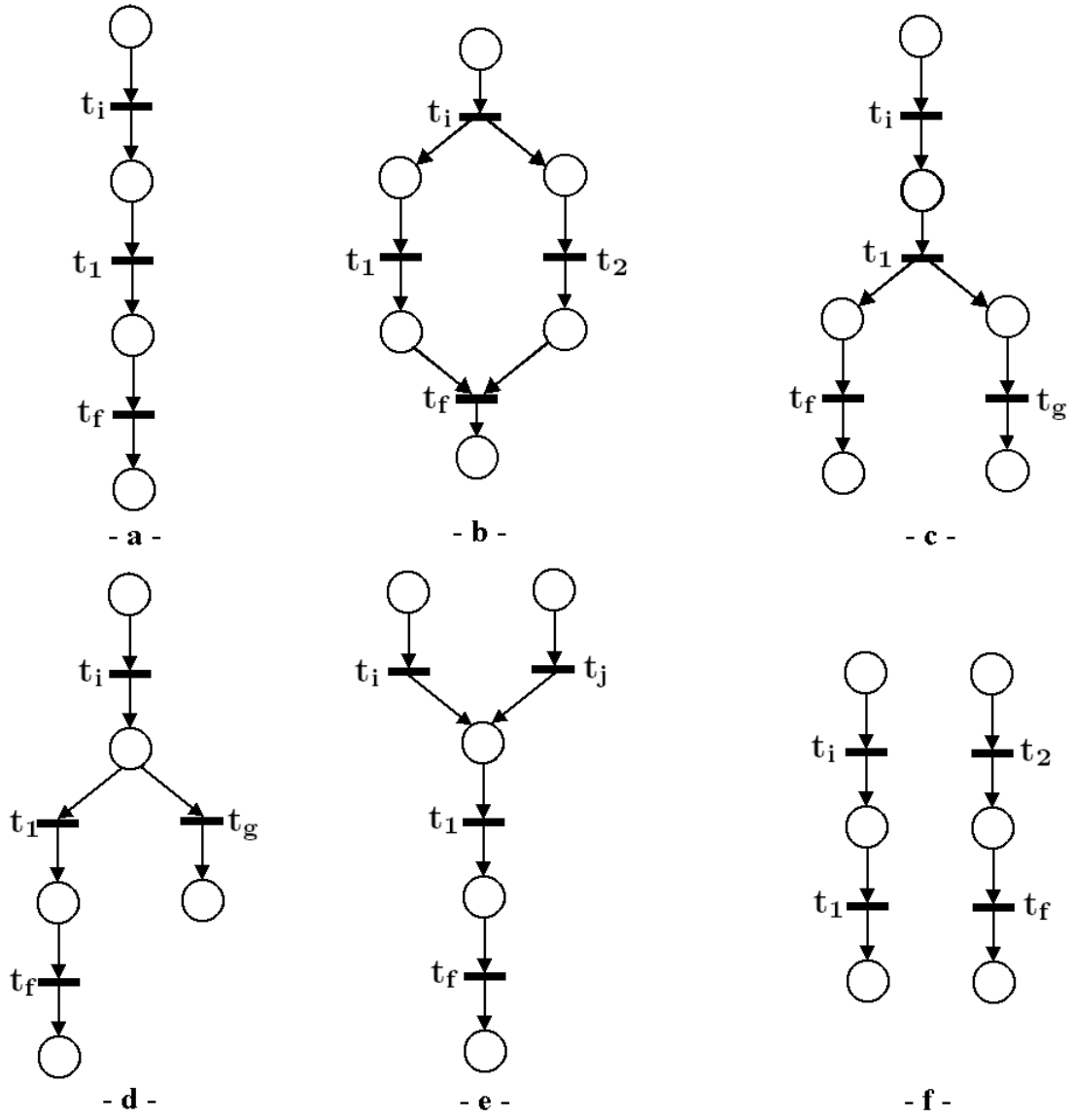


FIG. 3.1 – Exemple de dépendances entre transitions

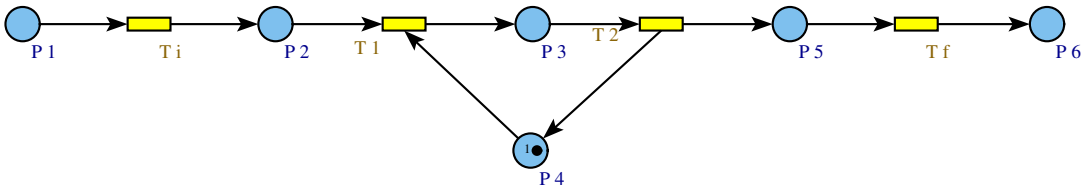


FIG. 3.2 – Marquage de contexte en P4

dépendance faible, les scénarios amenant à la transition aval peuvent également devenir invalides. La dépendance est alors supprimée, il y a non dépendance.

Une dépendance structurelle forte ou une non dépendance ne peuvent cependant pas devenir une dépendance faible.

3.1.2 Marquage de contexte

La notion de dépendance est structurelle et ne dépend donc pas du marquage initial du réseau de Petri. Nous ne prendrons donc pas en compte le marquage initial. Cependant, dans certains cas il sera nécessaire de considérer une partie du marquage initial que nous appellerons *contexte*.

Définition 4 *Un **marquage de contexte** est une partie du marquage initial du réseau de Petri, qui peut être utilisée, mais qui doit au final être réinitialisée, et qui ne peut pas être située sur une place d'entrée de la transition amont ou sur une place de sortie de la transition aval.*

La figure 3.2 présente un réseau de Petri avec un marquage de contexte. La figure 3.3(b) présente elle un cas de marquage qui ne peut pas être considéré comme un marquage de contexte.

3.1.3 Sous-réseau de Petri

En cas de dépendance, il est possible d'isoler une partie du réseau de Petri. Cette partie est un sous-réseau de Petri, qui est formé par un sous-ensemble de l'ensemble T des transitions du réseau, et par l'ensemble des places du réseau impliquées dans ces transitions, c'est à dire soit les places précédentes, soit les places suivantes d'une des transitions du sous-réseau.

Ce sous-réseau est un réseau de Petri $\mathcal{R}' = (P', T', {}^\bullet(.),\ (.)^\bullet)$ où :

– $T' \subset T$,

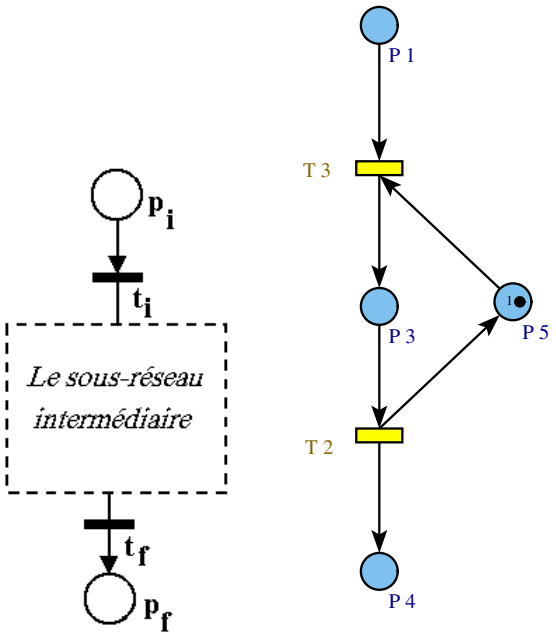
– $P' = \{\ p \in P \quad | \quad \exists t' \in T', \quad (p \in {}^\bullet t') \vee (p \in t'^\bullet)\ \}$,

– les fonctions d'incidence amont et aval ne sont pas modifiées.

Une *transition amont* au sous-réseau de Petri est une transition du sous-réseau dont les places d'entrée ne sont pas les places de sortie d'une transition du sous-réseau.

Une *transition aval* au sous-réseau de Petri est une transition du sous-réseau dont les places de sortie ne sont pas les places d'entrées d'une transition du sous-réseau.

Le marquage initial du réseau n'est donc pas utilisé. Pour étudier les dépendances et les sous-réseaux délimités, nous considérons uniquement, selon l'utilisation, le marquage de pré-condition de la transition amont ou le marquage de post-condition de la transition aval, et éventuellement un marquage de contexte.



(a) Cas général (b) Cas où le sous-réseau n'est pas déterminable

FIG. 3.3 – Détermination d'un sous-réseau en cas de dépendance forte

Il ne doit y avoir qu'une seule transition amont et qu'une transition aval. Les transitions du sous-réseau sont alors celles qui apparaissent dans les scénarios menant de la transition aval, à la transition amont. Pour pouvoir extraire un sous-réseau, la structure générale du réseau de Petri doit donc être telle que présentée sur la figure 3.3(a).

Sur la figure 3.3(b) par contre, il serait nécessaire de considérer un marquage de contexte en P3, mais il serait situé sur une place d'entré de la transition amont (et aussi sur une place de sortie de la transition aval). Cela n'est donc pas conforme à la définition d'un marquage de contexte. Ce cas est alors exclu car les définitions précédentes ne permettent pas de définir les transitions amont et aval. Ainsi, malgrès la dépendance forte entre T_i et T_f, on ne peut pas délimiter de sous-réseau.

3.1.4 Réseau de Petri inversé

Pour déterminer les dépendances entre transitions, le concept de réseau de Petri inversé est utilisé.

Soit un réseau de Petri $\mathcal{R} = (P, T, Pre, Post)$, son *réseau de Petri inversé* est un réseau de Petri $\mathcal{R}^{-1} = (P', T', Pre', Post')$ où :

- $P' = P$,
- $T' = T$,
- les applications Pre' et $Post'$ sont définies par :

$$\forall t \in T, \begin{cases} Pre'(.,t) = Post(.,t) \\ Post'(.,t) = Pre(.,t) \end{cases}$$

On présente sur la figure 3.4 un réseau de Petri et son inverse. On s'aperçoit qu'il suffit d'inverser le sens de tous les arcs dans le réseau.

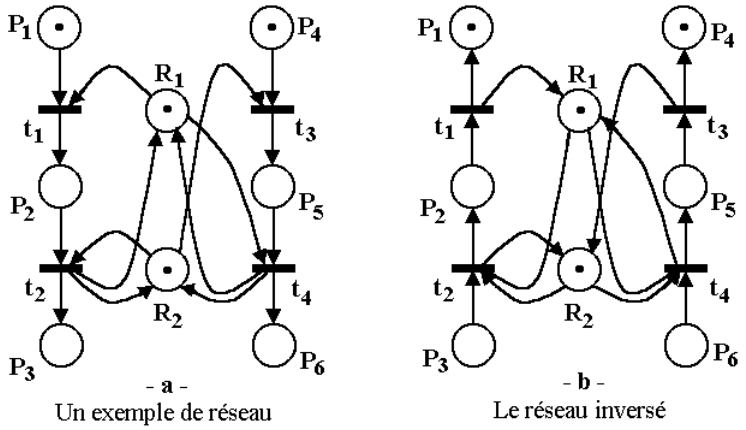


FIG. 3.4 – Un réseau de Petri et son inverse

3.2 Objectifs

Notre but est de déterminer le type de dépendance entre deux transitions d'un réseau de Petri, et d'extraire, en cas de dépendance, les scénarios de tir amenant de la transition amont à la transition aval, en considérant le marquage initial de la transition amont et le contexte.

3.2.1 Intérêts de l'approche

Cette approche doit être couplée avec l'analyse temporelle de scénarios que nous avons présentée au chapitre précédent. La méthode d'analyse temporelle nécessite en effet de posséder un scénario à étudier. La détermination des dépendances permet de construire des scénarios pertinents, qu'il est utile d'étudier temporellement.

Réduction de réseau de Petri T-temporels

Une des applications possibles est la réduction des réseaux de Petri T-temporels. En cas de dépendance forte, on peut, dans certaines conditions, remplacer le sous-réseau délimité par une seule transition, en calculant sa durée de sensibilisation.

Il faut pour cela analyser temporellement tous les scénarios amenant de la transition amont à la transition aval. On obtient ainsi la durée de chaque scénario. La date de tir possible de la transition aval est l'union de toutes ces durées. Le sous-réseau peut être remplacé par une seule transition, de la manière suivante :

- le marquage de pré-condition de cette transition est le marquage de pré-condition de la transition amont,
- le marquage de post-condition de cette transition est le marquage de post-condition de la transition aval,
- la durée de sensibilisation de cette transition est la date de tir de la transitions aval.

Un exemple de réduction d'un réseau de Petri T-temporel est présenté sur la figure 3.5. La durée de sensibilisation de la transition de substitution est $[a,b] = [a1+a2+a3\ ,\ b1+b2+b3]$.

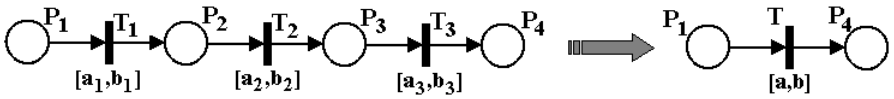


FIG. 3.5 – Exemple de réduction d'un réseau de Petri T-temporels

Cette substitution n'est possible que dans les conditions suivantes :

- hormis la transition amont, aucune transition du sous-réseau ne possède une place d'entrée qui n'est pas la place de sortie d'une transition du sous-réseau (contre-exemple de la figure 3.6(a)),
- de même, hormis la transition aval, aucune transition du sous-réseau ne possède une place de sortie qui n'est pas la place d'entrée d'une transition du sous-réseau(contre-exemple de la figure 3.6(b)),
- la date de tir calculée pour la transition aval (c'est à dire la durée de sensibilisation pour la transition de substitution) est un intervalle (contre-exemple de la figure 3.6(c) : les dates de tir possibles de T_f sont $[1,2] \cup [3,5]$ ce qui ne forme pas un intervalle).

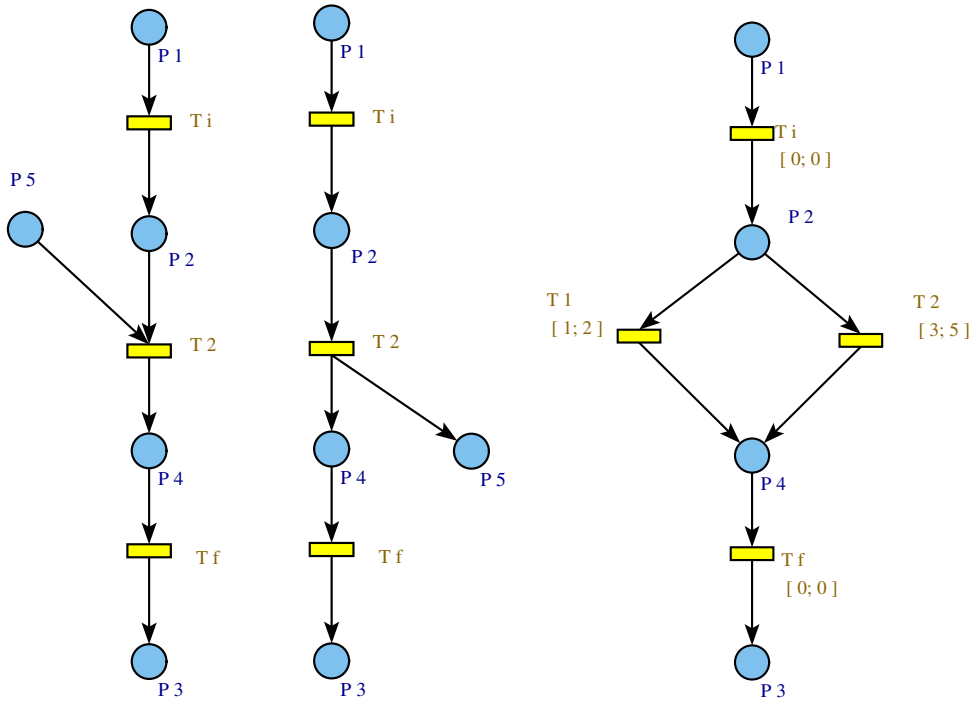


FIG. 3.6 – Sous-réseaux ne pouvant pas être réduits

Dans les deux premiers cas, la réduction est impossible car les places incriminées permettent des communications avec le reste du réseau, par d'autres biais que les transitions amont et aval. Dans le dernier cas, les intervalles de tir de la transition aval étant disjoints, il n'est pas possible d'attribuer une durée de sensibilisation à la transition de substitution. La substitution ne peut donc pas avoir lieu.

Détermination de conditions de sûreté ou de vivacité

A partir d'une transition amont, on veut pouvoir vérifier qu'une transition ne sera pas franchie. On vérifie pour cela qu'il y a non dépendance entre la transition amont et cette transition. En cas de dépendance faible, une analyse temporelle symbolique permettra de trouver les conditions qui supprimeront cette dépendance.

On peut également vérifier qu'une transition sera nécessairement franchie. On vérifie pour cela qu'il y a dépendance forte entre les transitions. A nouveau, en cas de dépendance faible, on pourra déterminer les conditions sur les durées de sensibilisation qui la transformeront en dépendance forte.

3.2.2 Méthode employée

La méthode de détermination des dépendances entre transitions a été crée par M. SOGBOHOSSOU et D. DELFIEU [14]. Elle consiste à effectuer deux types d'analyses. A partir d'un marquage initial égal au marquage de pré-condition de la transition amont t_i, on étudie l'accessibilité avant. On détermine ainsi tous les scénarios partant de t_i. Un scénario est un succès si l'on atteint t_f. Autrement, il faut fixer des conditions d'arrêt pour les scénarios qui échouent.

Pour chacun des scénarios ayant réussi, on effectue une analyse arrière. Pour cela on inverse le réseau, et on tire les transitions inversées, en partant du marquage final du scénario avant. On remonte ainsi, de la même manière qu'en accessibilité avant, jusqu'à la transition amont, ce qui signifie alors un succès.

Si tous les scénarios, avant et arrière, ont réussi, on en déduit que la dépendance entre t_i et t_f est forte. Si au moins un scénario a échoué, la dépendance est faible. Si aucun n'a réussi il y a non dépendance.

Pour statuer correctement sur les dépendances, il est nécessaire d'effectuer ces deux types d'analyse. Elles correspondent aux deux causes de la dépendance faible. L'accessibilité avant permet de découvrir des scénarios, partant de t_i, mais n'aboutissant pas à t_f. L'accessibilité arrière permet de découvrir (une fois remis dans l'ordre) des scénarios aboutissant à t_f, mais qui ne sont pas passés par t_i.

3.3 Algorithme de détermination des dépendances

3.3.1 Données de départ

Au départ, la transition amont t_i et la transition aval t_f sont connues. On en déduit alors le marquage initial qui sera utilisé pour la recherche des dépendances : il est égal au marquage de pré-condition de t_i.

On peut ensuite considérer comme marquage supplémentaire un marquage de contexte tel que définit au paragraphe 3.1.2.

Nous remarquons cependant que dans le réseau de la figure 3.6(a), nous ne pouvons pas considérer un marquage de contexte en P5 car il ne serait pas réinitialisé. Or, selon les définitions données dans la partie 3.1, il y a bien dépendance forte entre Ti et Tf. Cet algorithme est donc plus restrictif et ne trouvera pas cette dépendance.

3.3.2 Accessibilité avant et accessibilité arrière

Comme nous l'avons signalé, deux analyses sont nécessaires pour statuer sur le type de dépendance.

Nous effectuons tout d'abord une étude de l'accessibilité avant. A partir du marquage initial et du marquage de contexte, nous tirons les transitions sensibilisées jusqu'à ce qu'une des conditions d'arrêt soit vérifiée. En cas de conflit entre deux transitions il est nécessaire de suivre les deux possibilités. Il faut donc mémoriser l'état de l'analyse pour y revenir ultérieurement.

Nous devons ensuite étudier l'accessibilité arrière. Pour cela nous commençons par considérer le réseau de Petri inversé.

Nous utilisons ensuite l'équivalence entre accessibilité avant et accessibilité arrière. En effet, l'accessibilité arrière (c'est à dire en utilisant le réseau de Petri inversé) entre un marquage initial M_0 et un marquage final M_f, est équivalente à l'accessibilité avant (c'est à dire en utilisant le réseau de Petri non inversé) entre M_f et M_0 [14]. Ce résultat nous permet d'utiliser le même algorithme pour l'accessibilité arrière et pour l'accessibilité avant.

Le principe de l'analyse arrière est donc le même qu'en avant. Nous partons d'un des marquages finaux obtenus par l'analyse avant, nous appliquons le même algorithme, jusqu'à ce que les mêmes conditions d'arrêt soient vérifiées. Nous avons cependant dans l'analyse arrière l'avantage de connaître le marquage final produit par l'analyse avant. Ceci nous permet d'étudier un cas tel que celui de la figure 3.6(b). Ce réseau, en inversé, est pourtant similaire à celui de la figure 3.6(a) qui n'est pas traitable, mais le fait d'être en analyse arrière nous permet de posséder un jeton en P5 et ainsi de réussir l'analyse.

3.3.3 Conditions d'arrêt

Il est nécessaire de fixer des conditions d'arrêt de l'exploration des scénarios.

La première condition d'arrêt de l'exploration est synonyme de réussite. Pour cela il faut que la transition aval ait été franchie et que le marquage de contexte ait été réinitialisé au moins une fois. Ainsi, si la transition aval est franchie mais que le contexte n'est pas retourné à son état initial, il faut continuer à tirer des transitions jusqu'à sa réinitialisation. Dans un tel cas, on interdira cependant le tir de transitions situées au-delà de la transition aval et pour la caractérisation temporelle des scénarios obtenus, les transitions tirées après la transition aval seront exclues.

La deuxième condition d'arrêt est l'état de blocage. Le scénario en construction n'est pas encore un succès mais il n'y a plus de transition franchissable. Le scénario est donc un échec.

Enfin, une troisième condition d'arrêt est fixée. Au cours de l'exploration on peut rencontrer des boucles infinies. Pour ne pas bloquer l'algorithme sur ces boucles, on fixe un nombre de tirs limite. Si le nombre de franchissement du scénario atteint cette limite, il est considéré comme un échec.

Cette condition est arbitraire et mériterait d'être améliorée. L'idée serait de détecter les boucles, en comparant les marquages précédents. Ce processus est utilisé dans les calculs d'espace d'état pour déterminer si le réseau est non borné. Il faut pour cela comparer le nouveau marquage produit avec ceux déjà atteints. S'il est déjà rencontré ou strictement plus grand qu'un marquage rencontré, il faut alors arrêter l'analyse.

3.3.4 Algorithme

Voici à présent l'algorithme général (Algorithme 3) qui permet d'explorer les scénarios partant d'une transition amont jusqu'à une transition aval. Cet algorithme est donc valable aussi bien pour l'accessibilité avant que pour l'accessibilité arrière.

marquage courant ← marquage initial
Répéter
  Répéter
    Tant que *existe transition franchissable sans conflit ET aucune condition d'arrêt remplie* **Faire**
      Tirer la transition et l'enregistrer dans la liste du scénario
      Calculer le nouveau marquage
    Fin Tant que
    Si *existe transition franchissable en conflit ET aucune condition d'arrêt remplie* **Alors**
      Tirer la transition et l'enregistrer dans la liste du scénario
      Calculer le nouveau marquage
      Si *existe une autre transition en conflit à tirer* **Alors**
        Mémoriser sur la pile : l'état du processus pour ce prochain scénario à traiter, et la transition à tirer prochainement
      Fin Si
    Fin Si
  Jusqu'à *une condition d'arrêt est remplie*
  Enregistrer le scénario trouvé
  Si *pile de scénario non vide* **Alors**
    Désempiler
    Restaurer le contexte pour poursuivre la construction du scénario non terminé
  Fin Si
Jusqu'à *pile de sauvergarde est vide*

Algorithme 3 : Algorithme d'exploration des scénarios

Les transitions sont tirées avec une priorité pour les transitions qui ne sont pas en conflit. On retrouve ici la même priorité qui est appliquée dans l'algorithme de preuve. Cela permet d'activer le maximum de conflits. Pour les transitions en conflit, si plusieurs transitions sont effectivement franchissable, on mémorise l'état de l'exploration, et on explore chaque possibilité.

Lorsque l'algorithme se termine on connaît le nombre de scénario qui ont réussis, le nombre de ceux qui ont échoués, les transitions franchies pour chacun des ces scénarios, et le marquage final atteint.

L'algorithme de détermination des dépendances entre transitions consiste à :

- lancer cet algorithme pour l'accessibilité avant, avec le marquage initial de la transition amont et le marquage de contexte,
- inverser le réseau de Petri,
- relancer l'algorithme en accessibilité arrière, pour chaque marquage obtenu à l'aide d'un scénario valide par l'accessibilité avant ,
- statuer sur le type de dépendance en fonction des résultats obtenus.

A partir des résultats obtenus par les deux parcours avant et arrière, on détermine le type de dépendance de la manière suivante :

- si tous les scénarios avant et tous les scénarios arrière ont réussi, on en déduit une dépendance forte,
- si aucun scénario avant n'a réussi (aucun parcours arrière n'est dans ce cas effectué), il y a non dépendance,
- sinon, si un ou plusieurs scénarios, avant ou arrière, ont échoué, la dépendance est faible.

3.4 Application sur l'exemple

Nous étudions de nouveau le cas pratique de la figure 2.3. Un plugin de LLBOX permet d'étudier les dépendances dans les réseau de Petri.

Il nous faut pour cela déterminer une transition amont et une transitions aval. Nous choisissons d'étudier plus particulièrement la tâche `1_1`. Pour cela nous fixons comme transition amont : `task1_1-M1`, et comme transition aval : `task1_1-M2`. Nous fixons également comme marquage de contexte un jeton en `bus CAN state`.

L'étude des dépendances nous apprend alors :

- qu'il y bien dépendance forte entre la transition amont et la transition aval,
- qu'il n'existe qu'un seul scénario entre ces deux transitions : `task1_1-M1`, `acces CAN task1_1`, `send CAN task1_1`, `task1_1-M2`.
- En faisant une étude de ce scénario nous en déduisons que la durée de la tâche est comprise entre $[3,5]$.

De cette étude nous déduisons :

- que cette tâche est indépendante des autres, puisqu'il y dépendance forte. C'est à dire qu'elle n'attends pas de message provenant d'autres tâches.
- cependant le sous-réseau délimité ne pourra pas être réduit car elle communique des messages vers les autres tâches.

3.5 Conclusion

L'analyse des dépendances entre transitions est un bon complément de l'algorithme de preuve en sémantique forte. Cela permet de construire des scénarios pertinents à analyser. Elle peut être utile pour réduire des réseaux de Petri, mais plus généralement pour l'étude des relations de causalité entre transitions.

L'algorithme développé permet de déterminer correctement la plupart des cas de dépendance. Certaines limitations existent toutefois, notamment lorsqu'un marquage extérieur doit être utilisé (figure 3.6(a)). Citons également l'exemple de la figure 3.7, dans lequel les transitions T1 et T4 sont en dépendance forte. Cette dépendance forte ne sera pas détecté par l'algorithme. En effet, il y a deux scénarios avant : $T_1,\ T_2,\ T_4$ et $T_1,\ T_3,\ T_4$ qui produisent respectivement les marquages finaux $P_5 \otimes P_3$ et P_5. Si pour l'analyse arrière on essaye de passer par T2 en partant de P5 uniquement, on tombe sur un blocage.

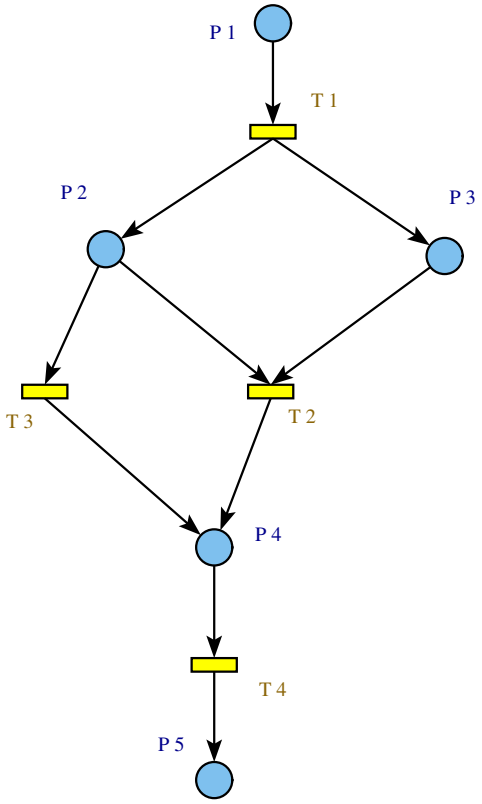


FIG. 3.7 – Dépendance forte non détectée

Chapitre 4

Analyse des résultats temporels

Les deux chapitres qui suivent constituent notre contribution à la thématique de recherche.

L'algorithme de preuve et d'analyse temporelle fournit les valeurs symboliques des dates de tir de chaque transition du scénario. Une des utilisations de ces résultats est de calculer les valeurs numériques, ce qui nous fournit les intervalles de tir possibles. Dans ce chapitre, nous allons présenter des méthodes pour utiliser les valeurs symboliques fournies.

4.1 Format d'écriture des dates de tir

Clarifions tout d'abord la façon de calculer et d'écrire les dates de tir des transitions. Dans un scénario, la date de tir Di d'une transition t_i est une valeur numérique. Cette valeur dépendra de l'exécution réelle du scénario. A l'aide de l'algorithme de preuve, nous calculons la borne minimale et la borne maximale de cette valeur, que nous pouvons noter Di_{min} et Di_{max}. Ainsi, on peut écrire $Di \in [Di_{min}\ ,\ Di_{max}]$.

Les bornes Di_{min} et Di_{max} sont calculées de manière symbolique, à l'aide des durées de sensibilisation des transitions, que nous noterons $[di_{min}\ ,\ di_{max}]$ pour la transition t_i, et à l'aide des dates de production des jetons, c'est à dire à l'aide des dates de tir précédemment calculées, donc des symboles $D1,\ D2, \ldots Dj$. La méthode d'écriture des dates de tir est donc récursive : elle s'écrit à l'aide des dates précédentes.

Considérons l'exemple de la figure 4.1, présenté dans la thèse de N. Rivière [13]. La date d'arrivée du jeton en P1 est $D0$ (elle peut être ou non égale à zéro).

– Nous écrivons alors date de tir $D1$ de T1 de la manière suivante :

$$D1 \in [D0 + d1_{min}\ ,\ D0 + d1_{max}]$$

– La date de tir de T3 sera :

$$D3 \in [D1 + d3_{min}\ ,\ D0 + d3_{max}]$$

– Et celle de T2 qui est en conflit avec T4 :

$$D2 \in [D1 + d2_{min}\ ,\ min(D1 + d2_{max}, max(D1, D3) + d4_{max})]$$

Nous allons maintenant voir comment utiliser ces dates de tir symboliques.

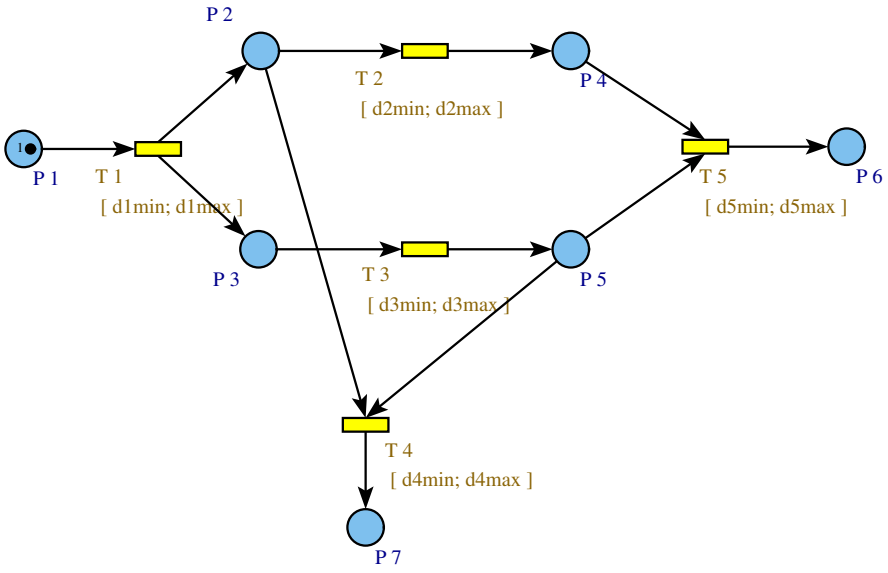


FIG. 4.1 – Exemple de réseau de Petri pour l'analyse des dates symboliques

4.2 Application numérique

Le calcul des intervalles de tir numériques doit être effectué dans l'ordre de tir des dates. Il pourrait être effectué directement au cours de la preuve.

La valeur minimum d'une date est calculée à l'aide du minimum des durées de sensibilisation et des valeurs minimum des dates apparaissant dans l'expression symbolique. La valeur maximum est elle calculée à l'aide des valeurs maximum et des durées de sensibilisation maximum.

Ce calcul nous fournit les dates de tir possibles d'une transition. Cependant, il ne permet pas de statuer sur la validité d'un franchissement. Il se peut en effet que l'on trouve une transition avec un intervalle de tir numérique correct (sa valeur minimum est inférieur à sa valeur maximum), mais dont le tir est dans tout les cas impossibles.

Par exemple, pour le réseau de la figure 4.1, prenons les valeurs numériques suivantes :
- T_1 : $[0,10]$,
- T_2 : $[1,1]$,
- T_4 : $[2,2]$,
- et pour toutes les autres $[0,0]$.

On analyse les deux scénarios T_1, T_2, T_3, et T_1, T_3, T_4, on calcule les valeurs numériques des dates de tir de chaque transition et on obtient les résultats suivants :
- $D1 \in [0,10]$
- $D3 \in [0,10]$
- $D2 \in [1,11]$
- $D4 \in [2,12]$

Ainsi la transition T4 serait tirable entre 2 et 12 unités temps. Ceci est faux car dans ce cas la transition T4 n'est jamais tirable à cause du conflit avec T2. L'application numérique n'est donc

pas suffisante pour déterminer la validité d'un conflit. Pour obtenir une condition suffisante, il est nécessaire d'analyser les valeurs symboliques trouvées.

4.3 Validation de scénarios

4.3.1 Problème

Un scénario est **valide** si tout d'abord il a pu être prouvé par l'algorithme de preuve. On vérifie de cette façon l'acessibilité. Mais dans les réseaux de Petri T-temporels, en sémantique forte, cela n'est pas suffisant.

En effet, en cas de conflit entre transitions, il se peut qu'une des transitions ne soit jamais franchissable. C'est le cas sur le réseau de la figure 4.3.1, dans lequel la transition T2 ne peut pas être franchie car T1 l'est nécessairement avant.

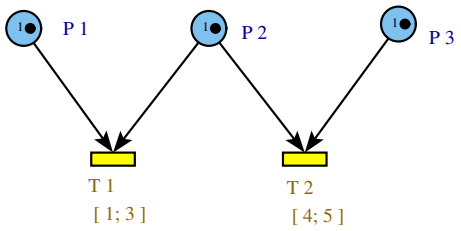


FIG. 4.2 – Conflit avec déterministe

Ainsi, pour qu'un scénario soit valide, il faut de plus, que toutes les transitions du scénario soient franchissables. Cette vérification ne doit toutefois être effectuée que pour les transitions en conflit. Et comme nous l'avons précédemment évoqué, elle doit être effectuée en manipulant les valeurs symboliques des dates.

4.3.2 Méthode

Pour vérifier qu'une transition en conflit est franchissable, il faut vérifier que la valeur minimum de la date de tir soit inférieure à la valeur maximum. Cependant, le calcul doit être effectué en symbolique jusqu'à un certain point.

Les valeurs numériques sont trop imprécises pour effectuer cette vérification et ceci est due au passé commun entre les transitions en conflit. En effet, dans le calcul numérique, la date de tir des transitions précédentes n'est pas connue précisément. Pour la valeur minimum, on prend les valeurs minimum des dates de tir et pour la valeur maximum les valeurs maximum. Mais lorsque l'on veut comparer les deux valeurs, on oublie que la date de tir des transitions précédentes n'a qu'une seule valeur.

Sur la figure 4.1, le passé commun entre T2 et T4 est constitué du tir de T1. Avec les valeurs données dans la partie 4.2, la durée de sensibilisation de T1 est très imprécise puisque comprise entre 0 et 10. Dans le calcul numérique de la date de tir de T2, la valeur 0 est utilisée pour la borne minimum et la valeur 10 pour la borne maximum. Si le franchissement de T2 est validé, l'intervalle calculé correspond effectivement à toutes les dates de tir possibles de T2. L'imprécision de la date de tir de T1 est de cette manière propagée sur celle de T2. Par contre, si l'on veut valider le franchissement

en comparant les deux valeurs numériques 0 et 10, nous commettons une erreur, car la valeur utilisée pour $D1$ n'est pas la même dans les deux valeurs. L'imprécision de $D1$ peut gommer sur les valeurs numériques la non franchissabilité de T2.

Nous allons donc raisonner sur ces dates symboliques en faisant attention à la prise en compte du passé commun. Nous considérons un cas de conflit entre deux transitions t et t'.

i. La valeur minimum de la date de tir de la transition t est de la forme :

$$A + dt_{min}$$

A et B sont deux expressions qui font apparaître des dates de tir Di et éventuellement des opérateurs $max(\ )$.

ii. La valeur maximum est elle de la forme :

$$min(A + dt_{max}, B + dt'_{max})$$

iii. Nous devons donc vérifier :

$$A + dt_{min} \leq min(A + dt_{max}, B + dt'_{max})$$

iv. Puisque $dt_{max} \geq dt_{min}$, nécessairement $A + dt_{min} \leq A + dt_{max}$. Il suffit donc de vérifier :

$$A + dt_{min} \leq B + dt'_{max}$$

v. **Élimination des dates** Di **:** nous devons maintenant supprimer toutes les dates Di qui apparaissent dans A et B, soit en les remplaçant, soit en les éliminant.
   - Nous commençons par la date de la transition tirée en dernier dans le scénario (pour pouvoir effectuer le plus d'éliminations possible).
   - Nous essayons de factoriser cette date dans les opérateur $max(\ )$.
   - Si elle apparaît dans les deux membres de l'inéquation, nous essayons de la supprimer.
   - Si elle apparaît dans les deux membres mais n'est pas supprimable, cela signifie qu'elle n'a pas pu être factorisée dans un opérateur $max(\ )$. Dans ce cas, il faut en toute rigueur élaborer une condition différente pour chaque valeur du $max(\ )$.
   - Si elle apparaît dans un seul membre, on la remplace par sa valeur minimum dans le membre gauche, ou par sa valeur maximum dans le membre droit. En effet, valider d'un scénario c'est déterminer une condition nécessaire pour son exécution. Cette condition doit donc être valable pour au moins une exécution possible, on prend donc la moins sévère, d'où cette substitution.

vi. Lorsqu'il n'y a plus de date Di nous obtenons une (ou plusieurs) condition symbolique, exprimée à l'aide des variables di_{min} et di_{max}, que nous pouvons vérifier numériquement pour valider le franchissement de la transition.

Cette méthode se généralise avec plusieurs transitions en conflit et notamment des transitions en de conflit indirect.

Invalidation de scénario

En utilisant une méthode similaire, nous pouvons également déterminer des conditions d'invalidation de scénarios. Pour invalider un scénario il faut trouver une condition de non franchissabilité d'une transition en conflit. Pour cela il faut vérifier que la date de tir maximum de cette transition est strictement inférieur à sa date de tir minimum.

On peut utiliser la même méthode pour déterminer cette condition. La seule différence est dans la substitution des dates *Di*. Il faut alors remplacer les remplacer par leur valeur minimum dans le membre droit, et par leur valeur maximum dans le membre gauche. Il faut en effet que cette condition soit vérifiée quelque soit l'exécution, d'où cette sévérité. Remarquons que les deux membres étant alors inversés, la substitution est donc finalement la même que pour la validation de scénarios.

4.4 Exemple

Nous allons appliquer cette méthode sur l'exemple de la figure 4.1. Dans cet exemple deux scénarios sont en concurrence : SC1 : $T_1,\ T_2,\ T_3,\ T_5$ et SC2 : $T_1,\ T_3,\ T_4$. Nous voulons déterminer les conditions de validation et d'invalidation de ces scénario.

Pour cela nous devons étudier le conflit entre les deux transitions T2 et T4. Nous calculons donc les dates de tir en symbolique de ces deux transitions et étudiant temporellement, par l'algorithme de preuve en sémantique forte, les deux scénarios SC1 et SC2.

Les dates de tir obtenues pour le scénario SC1 sont :
- $D1 \in [D0 + d1_{min}\ ,\ D0 + d1_{max}]$ date de tir de T1,
- $D3 \in [D1 + d3_{min}\ ,\ D1 + d3_{max}]$ date de tir de T3,
- $D2 \in [D1 + d2_{min}\ ,\ min(D1 + d2_{max}, max(D1, D3) + d4_{max})]$ date de tir de T2 qui est en conflit avec T4.
- Pour valider le scénario, la condition est :

$$D1 + d2_{min} \leq min(D1 + d2_{max}, max(D1, D3) + d4_{max}).$$

- Comme précisé au point iv. de la méthode, seule l'infériorité du membre de gauche avec le deuxième élément du $min()$ est à vérifier, c'est à dire :

$$D1 + d2_{min} \leq max(D1, D3) + d4_{max}.$$

- Nous devons éliminer D3. Elle n'est présente que dans le membre de droite, nous pouvons donc la remplacer par son expression maximum :

$$D1 + d2_{min} \leq max(D1, D1 + d3_{max}) + d4_{max}.$$

- Pour supprimer D1, nous commençons par la factoriser dans le $max()$ qui devient $max(0, d3_{max})$ et peut donc être remplacer par $d3_{max}$
- Nous pouvons alors éliminer D1 des deux membres de gauche et de droite. Il ne reste alors plus de dates de tir dans l'expression. La condition de validation de SC1 est donc :

$$\boxed{d2_{min} \leq d3_{max} + d4_{max}}$$

Si cette condition est vérifiée nous sommes assuré de l'exécution possible du scénario SC1. Mais il se peut que cette exécution reste indéterministe car le scénario SC2 peut également être franchi. La condition d'exécution nécessaire de SC1, est l'invalidation de SC2. Nous étudions donc le scénario SC2. Les dates de tir de SC2 sont les suivantes :

- $D1 \in [D0 + d1_{min}\ ,\ D0 + d1_{max}]$ date de tir de T1,
- $D3 \in [D1 + d3_{min}\ ,\ D1 + d3_{max}]$ date de tir de T3,
- $D4 \in [max(D1,D3) + d4_{min}\ ,\ min(max(D1,D3) + d4_{max}, D1 + d2_{max})]$ date de tir de T4 qui est en conflit avec T2.
- Pour invalider le scénario, la condition est cette fois inversée :

$$min(max(D1,D3) + d4_{max}, D1 + d2_{max}) \leq max(D1,D3) + d4_{min}.$$

- C'est à dire en simplifiant le $min()$:

$$D1 + d2_{max} \leq max(D1,D3) + d4_{min}.$$

- Nous devons éliminer D3 en la remplaçant par son expression minimum :

$$D1 + d2_{max} \leq max(D1, D1 + d3_{min}) + d4_{min}.$$

- Nous factorisons et éliminons D1, ce qui nous donne la condition d'invalidation de SC2 par SC1 :

$$\boxed{d2_{max} \leq d3_{min} + d4_{min}}$$

Cette condition nous assure que SC1 sera toujours exécuté au dépend de SC2. On remarque qu'elle est plus restrictive que la condition de validation de SC1. Elle permet donc également de valider SC1, mais ça ne serait pas forcément le cas si plusieurs transitions étaient en conflit.

4.5 Applications possibles

Les dates symboliques calculées par l'algorithme de preuve ont plusieurs utilités. On peut tout en faire une application numérique comme cela est expliqué dans la partie 4.2. On obtient ainsi les instants de tir possibles pour une transition dans un scénario donné. Si l'on étudie tous les scénarios amenant à une transition, on peut obtenir l'ensemble des instants de tir possibles.

On peut conserver dans l'expression des valeurs minimum et maximum de tir des valeurs symboliques. Cela peut alors être utile pour faire varier les durées de certaines transitions et analyser par exemple la durée finale du scénario.

Si l'on reste en symbolique on peut obtenir des conditions de validité de scénarios. On peut utiliser ces conditions pour modifier le système afin de s'assurer que les "bons" scénarios puissent être exécutés. Cependant, cela n'assure pas qu'il seront nécessairement exécutés. Pour vérifier qu'un scénario est obligatoirement exécuté au dépend d'un autre, il faut vérifier que le scénario concurrent n'est pas valide.

Chapitre 5

Enrichissement de scénarios

L'analyse d'un scénario nous fournit des informations temporelles sur le tir d'une transition. Ces informations ne sont cependant pas complètes, c'est à dire qu'elles ne couvrent pas l'ensemble des instants de tir possibles. Pour cela il faut analyser l'ensemble des scénarios amenant à la transition considérée. Nous allons présenter dans ce chapitre les méthodes qui permettent d'enrichir les scénarios pour obtenir des informations plus complètes.

On va d'une part rajouter des transitions dans les scénarios pour obtenir des résultats plus conformes, qui prennent en compte tous les conflits, et d'autre part construire de nouveaux scénarios qui représenteront d'autres alternatives temporelles.

5.1 Analyse arrière

Pour obtenir l'ensemble des dates de tir possibles d'une transition, il faut déterminer et analyser tous les scénarios qui peuvent amener, à partir du marquage initial, au tir de cette transition. Pour cela, il faut effectuer une analyse de diagnostic du réseau de Petri, c'est à dire partir de la transition étudiée pour remonter vers le marquage initial.

On traite ainsi les cas de conflits de jetons, dans lesquels un ou des jetons peuvent être produits de différentes manières. Pour chaque manière de créer ces jetons, on fait correspondre un scénario différent à étudier. C'est ce qui est déjà fait dans l'étude des dépendances : dans l'analyse arrière les conflits de jetons deviennent des conflits de transitions, et l'on détermine un scénario différent pour chaque alternative.

Raisonnement dans un contexte inconnu

En effet, dans la méthode de détermination des dépendances, nous effectuons déjà une analyse arrière. Cependant, nous ne prenons pas en compte le contexte inconnu. Dans cette méthode, pour effectuer l'analyse arrière nous possédons au départ les jetons produits par un scénario direct et éventuellement des jetons de contexte mais qui nous sont connus. Comme nous l'avons signalé au paragraphe 3.3.2, nous sommes de cette manière favorisés pour l'analyse arrière, car nous avons déjà examiné des scénarios avant.

Dans notre cas, nous ne connaissons pas de scénarios directs (nous sommes entrain d'essayer de les rechercher par une analyse arrière), nous ne connaissons donc qu'une partie du marquage final : le marquage final de la transition considérée. Le marquage qui est produit au cours du scénario et

non consommé nous est inconnu. Si l'on veut réaliser une accessibilité arrière ce marquage nous sera pourtant nécessaire.

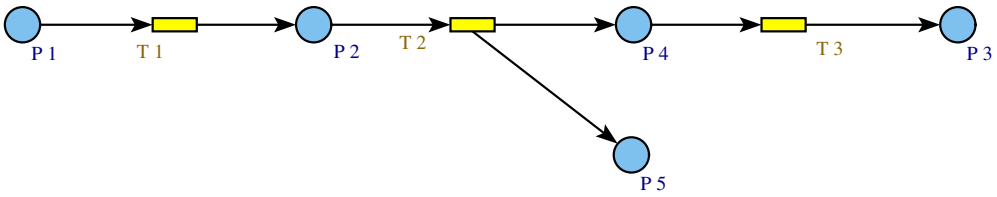


FIG. 5.1 – Réseau avec contexte final inconnu

Dans un réseau tel que celui de la figure 5.1, le marquage produit par T2 en P5 est pour l'analyse arrière un contexte inconnu qu'il sera pourtant nécessaire d'utiliser.

Nous ne sommes pas encore capable de raisonner dans un contexte inconnu. Nous considérons cependant dans la suite que nous possédons une méthode pour déterminer les scénarios qui permettent d'atteindre une place du réseau donnée.

5.2 Prise en compte des transitions en conflit

Dans un scénario, si une transition est en conflit avec d'autres transitions il faut connaître les dates de tir de ces autres transitions, pour pouvoir prendre en compte la sémantique forte. Pour cela il faut que les conflits soient effectifs. Il faut donc que soit présent dans le scénario les transitions qui vont permettre de rendre effectif les conflits (s'ils peuvent le devenir). On va donc enrichir le scénario à l'aide de ces transitions.

Dans la figure 5.2, le scénario que l'on veut étudier est constitué de T1 seule. Cependant, T1 est en conflit avec T2. Pour prendre en compte la sémantique forte, il faut connaître la date de tir de T2, et donc connaître la date de production du jeton P3, c'est à dire la date de tir de T3, qui doit donc être rajoutée dans le scénario.

Méthode d'enrichissement

Dans le scénario de base, nous considérons donc tour à tour toutes les transitions en conflit du scénario. Nous les examinons dans l'ordre inverse du scénario, pour éviter des recoupements.

Dans chaque groupe de conflit, nous étudions chaque place amont d'une des transition (sauf celles de la transition qui figure dans le scénario de base). Pour chacune de ces places, nous déterminons les scénarios qui permettent de la produire. Ces scénarios vont être destinés à enrichir le scénario de base. Si pour une place plusieurs scénarios sont possibles, nous allons créer un nouveau scénario de base pour chacun de ces scénarios. Cela correspond aux cas de conflits de jetons.

En considérant alors un scénario de base et un scénario pour l'enrichir nous réalisons les opérations suivantes :

– on doit insérer les transitions du scénario dans le scénario de base, avant le conflit en question.

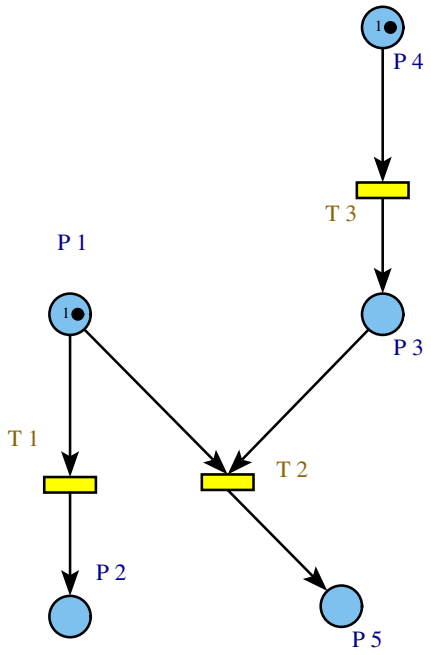


FIG. 5.2 – Activation d'un conflit

Cependant : si le scénario a des transitions communes avec le scénario de base, il y a deux façons de traiter ces transitions :

- si le scénario possède des transitions en conflit avec celles du scénario de base, alors les transitions communes sont insérées une nouvelle fois dans le scénario de base. En effet, dans ce cas, le scénario rajouté a besoin de nouveaux jetons pour être exécuté. Il faut donc reproduire les jetons et donc rajouter les transitions déjà présentes.
- si le scénario n'est pas en conflit avec le scénario de base, alors il n'est pas nécessaire de rajouter les transitions déjà présentes. Cela signifie que le scénario utilise des jetons produits par le scénario de base mais non utilisés.

– le scénario de base est donc enrichi de nouvelles transitions. Cela constitue désormais le nouveau scénario de base.

On réalise donc tous les enrichissement nécessaires en agrandissant au fur et à mesure le scénario de base jusqu'à ce qu'il n'y ait plus de cas de conflit à traiter. Et on fait cela pour toutes les duplications du scénario de base que l'on a du créer à cause de conflits de jetons.

Remarque Il est possible que l'on rajoute plus de transitions que nécessaire ou que l'on duplique certaines transitions. Cela est parfois difficilement évitable. Toutefois, cela ne modifiera pas l'analyse temporelle du scénario. En effet, ces transitions supplémentaires ne seront pas tirées si elles ne possèdent de jetons pour les sensibiliser. Elles resteront par contre à la fin de la preuve, qui ne sera donc plus formellement correcte.

5.3 Permutations de transitions

Pour obtenir des résultats complets, il est également nécessaire de réaliser dans les scénarios des permutations entre transitions.

Un des intérêts de l'approche par la logique linéaire est la prise en compte du parallélisme vrai. Ainsi, pour analyser deux transitions en parallèles, il n'est pas nécessaire d'effectuer des entrelacements. L'étude dans un ordre arbitraire (qui est dans notre méthode l'ordre lexicographique) suffit. L'ordre de tir des transitions dans la preuve est donc un ordre partiel entre les transitions.

Cependant, en cas de conflits nous avons dit dans la section 2.3 que nous devions fixer un ordre total entre les transitions d'un même groupe de conflit. Dans la plupart des cas, seule une des transitions d'un groupe de conflit figurera dans un scénario, car l'autre ne sera généralement pas franchissable après le tir de la première.

Mais, il se peut que plusieurs transitions d'un même groupe de conflit figurent dans un scénario. C'est le cas par exemple dans les conflits indirects (voir la section 2.3 et la figure 2.2). Deux transitions qui sont en conflit indirect entre elles peuvent en effet être tirées toutes les deux, en parallèle. Même en cas de conflit direct, si plusieurs jetons sont disponibles, deux transitions en conflit peuvent être toutes les deux tirées.

Pour obtenir des résultats complets il faut réintroduire des entrelacements là où le parallélisme a été supprimé. Une fois que l'on a obtenu l'ensemble des scénarios enrichis par la méthode décrite précédemment, il faut donc de nouveau analyser chacun de ces scénarios.

Si dans un scénario figure plus d'une transition d'un même groupe de conflit, on crée, à partir de ce scénario, un nouveau scénario pour chaque permutation possible entre ces transitions. On peut pour cela se contenter de permuter ces transitions en intervertissant leur place. En effet, leur place dans le scénario n'a pas d'importance, car un scénario n'est pas ordonné, seul compte leur position relative qui fixera l'ordre entre elles.

5.4 Conflits de jetons

Il y a conflit de jetons lorsque plusieurs jetons (plus que nécessaire) sont disponibles pour le tir d'une transition. Ces jetons peuvent avoir des estampilles temporelles différentes, et donc ne pas être tous présents simultanément pour le tir de la transition. Alors, selon les jetons utilisés pour le tir, la date de tir sera différente.

Si l'on effectue une analyse de diagnostic par une analyse arrière, le problème des conflits de jetons doit être normalement réglé. En effet, comme nous l'avons déjà signalé au début du chapitre, en analyse arrière les conflits de jetons deviennent des conflits de transitions, et selon l'alternative choisie on produira deux scénarios différents.

Les conflits de jetons indiquent qu'il existe plusieurs manières de produire un jeton. Avec l'analyse arrière nous créons un scénario différent pour chacune de ces manières. Dans ces scénarios il ne doit pas subsister de conflits de jetons.

Cependant, certains conflits de jetons peuvent persister, voir être rajoutés. En effet, lorsque l'on enrichit le scénario, nous rajoutons des transitions. Ces transitions peuvent produire des jetons supplémentaires, dont l'utilisation n'est pas prévue mais pourrait avoir lieu.

On crée donc de nouveaux conflits de jetons. Encore une fois, pour obtenir des résultats complets

il faut les traiter. Cela ne peut ce faire qu'au cours de la preuve. Si on rencontre un conflit de jetons au cours de la preuve il faut créer plusieurs preuves différentes selon les jetons employés.

5.5 Application sur un exemple

Dans l'exemple de la figure 2.3, nous voulons étudier le scénario correspondant à la tâche 1_1. Nous avons déjà déterminé le scénario de base à l'aide de l'étude des dépendances et trouver le scénario suivant : `task1_1-M1`, `acces CAN task1_1`, `send CAN task1_1`, `task1_1-M2`. Nous allons maintenant enrichir ce scénario.

Nous constatons dans ce scénario que la transition `acces CAN task1_1` est en conflit avec une autre transition : `acces CAN task1_2`. Nous devons donc enrichir le scénario de base pour que ce conflit devienne effectif s'il le peut.

Nous nous apercevons que pour devenir effectif, il faut produire un jeton `send id CAN task1_2`. Le scénario permettant de produire ce jeton est constitué de la seule transition `task1_2-M1`. Nous devons donc rajouter cette transition dans le scénario, avant le conflit, ce qui donne pour nouveau scénario de base : `task1_1-M1`, `send id CAN task1_2`, `acces CAN task1_1`, `send CAN task1_1`, `task1_1-M2`.

Ce scénario permet d'étudier temporellement la tâche 1_1.

Maintenant, nous considérons de nouveau le scénario comportant l'ensemble des tâches. Dans ce scénario figure les deux transitions en conflit. Un ordre est fixé entre ces deux transitions, et nous avons dans le chapitre 2 étudié l'ordre : `acces CAN task1_1`, `acces CAN task1_2`. Pour obtenir des résultats complets, il faut étudier le scénario dans lequel ces deux transitions sont permutées.

Nous obtenons alors comme date de tir de la première transition en conflit :

```
Date: D5        Tir de: accesCANtask1_2
Date min: max(D0,D3)+d(accesCANtask1_2)
Date max: min(max(D0,D2)+d(accesCANtask1_1),max(D0,D3)+d(accesCANtask1_2))
```

Si l'on effectue le calcul numérique cette date a pour valeur $[3, 1]$.

Nous en déduisons que la transition n'est pas tirable. Le scénario inversé n'est donc pas valide. Il existe donc un ordre total entre les deux transitions en conflit, qui est `acces CAN task1_1`, `acces CAN task1_2`. Et le résultat sur la durée totale des tâches est donc complet.

5.6 Perspectives pour raisonner dans un contexte inconnu

Nous donnons dans ce paragraphe des pistes pour raisonner dans un contexte inconnu.

L'idée est de créer un processus d'enrichissement, qui rajoute des ressources au cours de l'analyse lorsqu'elles sont nécessaires.

Concrêtement, dans l'analyse arrière nous partons du marquage final de la transition considérée et nous franchissons les transitions en arrière, comme cela est fait pour étudier les dépendances. Cependant, pour franchir une transition nous devons utiliser au moins un des jetons produits précédemment, et nous pouvons également utiliser tous les jetons supplémentaires qui nous sont nécessaires.

Cela peut cependant poser des problèmes en cas de parallélisme. Sur la figure 5.6, nous présentons un réseau de Petri inversé que nous sommes entrain de parcourir. Dans l'état actuel, il n'y a pas de

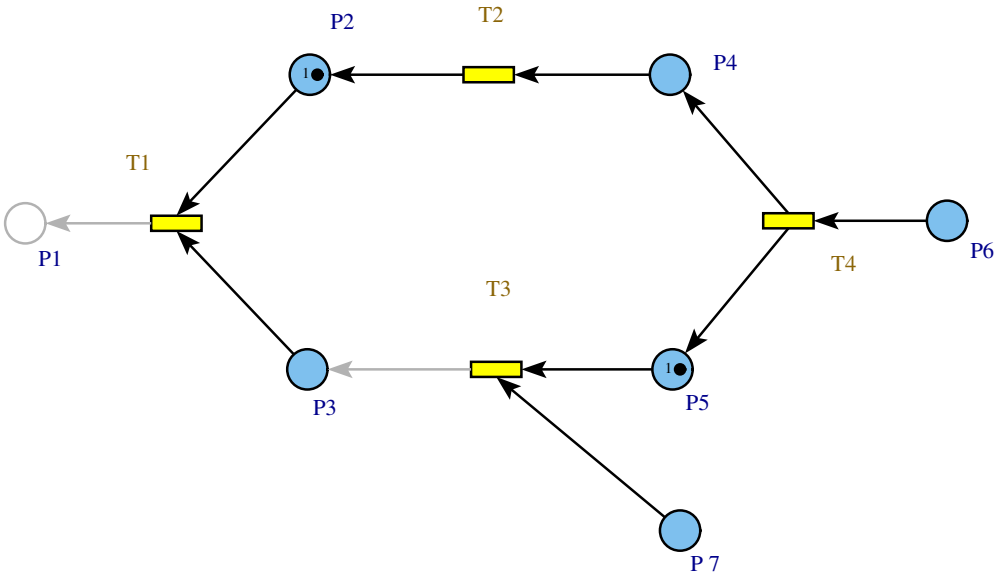


FIG. 5.3 – Réseau inversé en cours d'analyse avec contexte inconnu

transition directement franchissable, seules deux transitions T1 et T3 seront franchissables avec un enrichissement. Lorsque l'on regarde le réseau, nous savons qu'il vaut mieux tirer T3 d'abord, en enrichissant la place P7, car cela produira un jeton P3 et T1 sera alors franchissable sans enrichissement.

Cependant, dans une analyse automatique, il n'y a pas moyen de différencier les 2 transitions T1 et T3. Nous proposons donc de tirer au hasard l'une des deux transitions et de retenir le fait que la place P3 a été enrichie. Si l'on est amené à tirer ultérieurement une transition qui produise un jeton en P3, il faudrait alors se souvenir que P3 avait été enrichie et donc supprimer ce nouveau P3 car il a déjà été utilisé auparavant. L'ordre des transitions dans le scénario est bien entendu dans ce cas incorrect, mais un scénario n'est pas ordonné.

Conclusion

Nous avons présenté dans ce rapport plusieurs méthodes d'analyses de fractions de réseaux de Petri. Le but est de construire petit à petit un ensemble complet d'outils pour cette thématique. L'étude des dépendances et l'analyse arrière regroupent plusieurs objectifs d'étude : réduction d'un réseau de Petri, étude des liens de causalité, détermination d'informations temporelles complètes.

Dans ces objectifs, ces méthodes nous permettent de déterminer les parties du réseau à analyser et construisent les scénario à étudier. La construction des scénarios peut être complétée par un enrichissement de ces scénarios.

Nous pouvons alors effectuer l'analyse temporelle de ces scénarios grâce l'algorithme de preuve en sémantique forte. Cela nous fournit de manière symbolique les dates de tir des transitions.

Pour terminer l'étude et arriver aux conclusions, nous pouvons soit se contenter d'une application numérique des résultats, soit analyser symboliquement les résultats obtenus.

Il y a en perspective plusieurs points qui doivent être améliorés. Dans la détermination des dépendances nous avons soulevé quelques problèmes. Il faudrait notamment créer un mécanisme de détection des boucles. L'analyse arrière est encore une piste à explorer.

Une des tâches importantes à accomplir serait de rapprocher cette approche des méthodes déjà existantes sur les réseaux de Petri, pour les comparer voir les associer.

Bibliographie

[1] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time petri nets. *IEEE Transactions on Software Engineering*, 17(3) :259–273, March 1991. NewsletterInfo : 39.

[2] Beatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier (H.) Roux. Comparison of different semantics for time Petri nets. In *Automated Technology for Verification and Analysis (ATVA'05)*, volume 3707 of *Lecture Notes in Computer Science*, Taiwan, October 2005. Springer.

[3] Florent Frizon de Lamotte. *Logique linéaire et réseaux de Petri Temporels*. Rapport de dea, Institut de Recherche Cybernétique de Nantes, Ecole Doctorale STIMM, 2003.

[4] Guillaume Gardey, Olivier (H.) Roux, and Olivier (F.) Roux. Using Zone Graph Method for Computing the State Space of a Time Petri Net. In *Formal Modeling and Analysis of Timed Systems (FORMATS'2003)*, volume 2791 of *Lecture Notes in Computer Science*, pages 246–259, Marseille, France, September 2004. Springer–Verlag.

[5] Jean-Yves Girard. *Linear Logic*. *Theorical Computer Science*, vol. 50, 1987.

[6] François Girault. *Formalisation en logique linéaire du fonctionnement des réseaux de Petri*. Thèse de doctorat, Université Paul Sabatier, Toulouse, Décembre 1997.

[7] 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.

[8] Didier Lime. *Vérification d'applications temps réel à l'aide de réseaux de Petri temporels étendus*. Thèse de doctorat, Université de Nantes, 2004.

[9] P. M. Merlin. *A Study of the Recoverability of Computing Systems.* PhD thesis, Irvine : Univ. California, PhD Thesis, 1974. available from Ann Arbor : Univ Microfilms, No. 75–11026.

[10] 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.

[11] 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.

[12] 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.

[13] Nicolas Rivière. *Modélisation et analyse temporelle par réseaux de Petri et logique linéaire*. Thèse de doctorat, Université Paul Sabatier, Toulouse, Novembre 2003.

[14] Médésu Sogbohossou. *Raisonnement temporel sur un réseau de Petri t-temporel à l'aide de la logique linéaire*. Rapport de dea, Laboratoire d'Electrotechnique, de Télécommunications et d'Informatique Appliquée (LETIA), 2005.

[15] 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.