Mémoire pour l'obtention du diplôme de magistère en informatique.
Option : génie logiciel.
Thèse :

Présenter par :
Messaouda Bouneb
Equipe Formel
Loratoire LIRE, Université de Mentouri, 25000 Constantine

Table des matières

Table des figures

Chapitre 1

Introduction générale

1.1 Problématique :

Aux cours des décennies passées, les informaticiens, en collaboration avec les mathématiciens, ont eu pour défi d'élaborer des théories et des techniques garantissant le bon fonctionnement des systèmes, c'est-à-dire, selon des caractéristiques prescrites exprimant leurs comportements désirés. Les méthodes traditionnelles assurant de telles « garanties » ont été la simulation et le test. Ces méthodes permettent d'obtenir un certain degré de confiance dans la conformité du fonctionnement des systèmes, néanmoins elles souffrent de plusieurs problèmes à savoir :

-Ces méthodes de validation classiques sont généralement basées sur une couverture plus ou moins grande des exécutions possibles, par génération et validation de séquence de test, par conséquent elles prennent excessivement de temps.

-Ces méthodes nécessitent de se placer au plus prêt des conditions réelles d'utilisations pour être réaliste, ce qui est impossible dans certaines situations.

-Ces méthodes sont inexactes et imprécises, car elle fournissent souvent une évaluation probabiliste de la conformité.

Pour répondre à ces problèmes, des méthodes d'analyse formelle, apportant une certitude mathématique, ont été développées [Sai05]. Elles permettent de décrire sans ambiguïté et d'une manière rigoureuse le comportement des systèmes, ainsi elles peuvent garantir leur bon fonctionnement avant leur mise en service, c'est-à-dire de valider l'ensemble des comportements possibles du système par rapport aux services qui lui sont demandés.

Le succès relatif remporté par l'utilisation des méthodes formelles fait quelle commencent purement universitaire [Sai05]. Depuis de grande compagnie oeuvrant dans le domaine des systèmes concurrents, commencent également à réaliser l'importance de ces méthodes et tentent d'en imposer l'utilisation.

Ces méthodes formelles sont basées sur les modèles de spécification du parallélisme ainsi que les modèles sémantiques du parallélisme. Les modèles de spécification du parallélisme offrent les moyens formels pour décrire sans ambiguité les applications concurrentes, parmi ces modèles nous pouvons citer les réseaux de Petri

[Rei85], les algèbres de processus (ACP, CCS, CSP,..)[BK85] [Mil80] [Mil83] [Mil89] [Hoa85] et les techniques de description formels (ESTELLE, LOTOS, SDL)[ISO88] [BB87] [CCI88]. Quand aux modèles sémantiques du parallélisme, ils sont utilisés pour exprimer la sémantique du parallélisme des modèles de spécification.

Le modèle des réseaux de Petri est le modèle le plus élégant pour décrire le parallèlisme. En plus du parallèlime, il exprime aussi les autres aspects de conflit, de séquencement..etc. De ce fait les réseaux de Petri peuvent donner une description claire du système spécifié. Ils ont prouvés leurs aptitudes pour modéliser des classes importantes de systèmes qui recouvrent des classes de systèmes de production, de systèmes automatisés, de systèmes informatiques et de systèmes de communication, pour n'en citer que quelques-uns, afin de permettre leur évaluation et leur amélioration. Un caractère très intéressant de ce modèle est qu'ils est basé sur une représentation mathématique graphique. Ce qui facilite son utilisation par des personnes dont la formation scientifique n'est pas forcement poussée.

Malgré les avantages sus-cités du modèle des réseaux de Petri, un point crucial peut limiter son utilisation : la vérification de plusieurs propriétés comportementales, telles que les bissimulations, nécéssitent la traduction des réseaux de Petri à analyser dans d'autres structures du genre systèmes états/transitions. Afin de capturer le parallèlisme, dans le présent travail on s'interesse à la traduction des réseaux de Petri en terme de systèmes de transitions étiquetées maximales.

D'autre part, dans ce travail on va traiter un problème important, à savoir le problème de l'explosion combinatoire du graphe d'état. Pour cela nous proposons une technique d'agrégation des transitions modulo la relation de bissimulation de maximalité. Cette technique permet de réduire la taille d'un système de transitions étiquetées maximales.

Afin de concrétiser notre étude théorique nous avons développé l'outil MSOS-PN (Maximality-based Structured Operational Semantics for Petri Net). Cet outil permet de générer selon le choix soit un système de transitions étiquetées maximales, soit un système de transitions étiquetées maximales avec agrégation de transitions.

1.2 Contributions :

Nos contributions peuvent se résumer en :

- Proposition d'une sémantique opérationelle de maximalité pour les réseaux de Petri.
- Proposition d'une méthode de réduction à la volé des systèmes de transitions étiquetées maximales, basée sur l'agrégation des transitions modulo la relation de bissimulation de maximalité.
- Concrétisation des études théoriques par le développement de l'outil MSOS-PN (Maximality-based Structured Operational Semantics for Petri Net) .

- Intégration de l'outil MSOS-PN dans l'environnement de vérification formelle FOCOVE[BDS02].

1.3 Plan du document :

Le mémoire est organisé de la manière suivante :

- Le chapitre 1 représente une introduction générale qui permet d'exposer le but de ce travail dans le contexte des réseaux de Petri.
- Le chapitre 2 est consacré à une présentation générale des réseaux de Petri ainsi que les concepts de base nécessaires pour leur exploitation. Différentes méthodes d'analyse des réseaux de Petri seront abordées, ainsi que quelques variantes de ce modèle.
- Le chapitre 3 met l'accent sur les limites de la sémantique d'entrelacement vis à vis de l'expression du pararallélisme, ainsi que le problème des sémantiques d'ordre partièl et de la ST-sémantique. La sémantique de maximalité sera présentée comme alternative pour répondre aux limites des sémantiques précédentes.
- Le chapitre 4 aborde les contributions théoriques de ce travail, entre autre :
  - Il introduit, à travers des exemples simples, les définitions et notations utilisées dans la méthode de génération d'un graphe de marquage dans le contexte de la sémantique de maximalité. Par ailleurs une méthode de génération de systèmes de transitions étiquetées maximales relatives aux réseaux de Petri sera proposée.
  - Il présente une méthode de réduction à la volé des systèmes de transitions étiquetées maximales basée sur l'agrégation des transitions modulo la relation de bissimulation de maximalité.
- Le chapitre 5 est consacré à l'outil logiciel MSOS-PN implémantant les sémantiques opérationnelles sus-cités.
- Le chapitre 6 donne la conclusion et quelques perspectives de ce travail.

1.4 Publications de l'auteur :

- D. E. Saidouni, N. Belala and M. Bouneb. Réseaux de Petri et Sémantique de maximalité . Siménaire National d'Informatique Biskra (SNIB'2008)[SBB08b].
- D. E. Saidouni, N. Belala and M. Bouneb. Aggregation of transitions in marking graph generation based on maximality semantics for Petri nets. Verification and Evalution of Computer and Communication systems (VECOS'2008)[SBB08a].

- D. E. Saidouni, N. Belala and M. Bouneb and A. Boudjaadar and B. Ouchenne. Using maximality- based labeled transitions as model for Petri nets. The International Arab Conference on Information Technology (ACIT'2008)[SBB08c].

Chapitre 2

Modèle des réseaux de Petri

2.1 Introduction :

Les systèmes concurrents et distribués sont connus par leur niveau de complexité, posant ainsi plusieurs problèmes importants pour leurs développement. Une question fondamentale est comment réussir à développer un système qui remplit, son comportement attendu ? En effet la description du comportement attendu d'un système n'est rien d'autre que son cahier de charge. Celui-ci couvre plusieurs aspects à savoir l'aspect fonctionnel, les contraintes d'environnement, les besoins utilisateurs, les contraintes financières... etc. Afin de prendre en considération tous ces aspects il est indispensable de disposer d'outils et de méthodologies de conception. Parmi ces outils nous trouvons les outils de modélisation. Ces derniers prennent une part importante pour l'achèvement de cet objectif. Par définition, un modèle est une représentation d'un système , qui permet de le décrire de façon claire [SB06]. Cependant pour comprendre pourquoi le faire et pourquoi un tel intérêt est attribué à leur utilisation, il faut bien rappeler que le développement d'un système complexe est le résultat de la coopération de toute une équipe sous des points de vue différents (spécification, conception, réalisation)... etc. Ceci justifie l'utilisation de modèle formels décrivant le système sans ambiguité. D'autre part, dans certain cas, il est impossible de réaliser des expériences réelles sur le système (par exemple : système spatial), parfois cela est dangereux. L'approche de modélisation permet de surmonter ces problèmes car le modèle est employé tôt dans le cycle de conception, ce qui permet de détecter les erreurs de conception logique avant que le système ne soit mis en œuvre.

Dans la littérature il a été proposé un grand nombre de techniques dotées d'un support mathématique pour raisonner sur la conformité des systèmes. L'idée générale est de décrire dans un cadre formel les systèmes à étudier, puis d'appliquer des méthodes rigoureuses pour démontrer que la description des systèmes est correcte, qu'elle répond bien à certains exigences formellement définies. Parmi les techniques les plus populaires nous trouvons le modèle des réseaux de Petri défini en 1962 par Carl Adam Petri dans sa thèse "Communication mit automaten" [**?**].

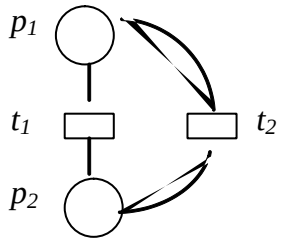


FIG. 2.1 – Exemple d'un réseau de Petri.

Les réseaux de Petri est un outil de modélisation graphique et mathématique qui peut être appliqué à plusieurs systèmes [Mur89]. D'un point de vue graphique les réseaux de Petri peuvent être considérés comme étant une méthode de communication visuelles très simple, compréhensible même par des personnes non spécialistes dans le domaine. Quand au point de vue mathématique, ce modèle offre une théorie bien fondée semblable aux systèmes équationnels et d'autre modèles mathématiques. De ce fait on peut dire que le modèle des réseaux de Petri et un bon médium de communication entre le niveau réalisation où les développeurs sont plus réaliste et le niveau conception où les théoriciens sont plus méthodiques [Mur89].

2.2 Représentation graphique :

Un réseau de Petri est un graphe biparti orienté dont les cercles symbolisent les places et les rectangles symbolisent les transitions. Un arc relie soit une place à une transition, soit une transition à une place. Mais jamais une place à une place ou une transition à une transition. Il est à noter que tous les arcs sont étiqueté par des entiers qui correspond à leurs poids. Par défaut, le poids d'un arc non étiquetés vaut 1, un exemple d'un réseau de Petri est donné par la figure 2.1. Chaque place et chaque transition peut être étiquetée par un nom, par exemple dans la figure 2.1 les places sont nommées p_1 et p_2 alors que les transitions sont nommées t_1 et t_2 .

Intuitivement les places permettent de décrire l'état instantané du système, les transitions correspondent aux actions que le système peut effectuer et les arcs expriment les conditions requises pour qu'une action puisse être exécutée, ainsi que l'effet de cet action sur l'état du système. L'évolution du système est modélisée par le tir d'une transition. D'une manière générale on peut dire que les réseaux de Petri fournissent une représentation statique des états du système et une modélisation dynamique de son évolution.

2.3 Définition formelle : [BD87][BDKP91]

Formellement, un réseau de Petri (en abrégé rdp) est un triplet (S, T, W) où S est l'ensemble des places, T est l'ensemble des transitions tel que $S \cap T = \varnothing$, et

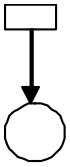


FIG. 2.2 – Transition source

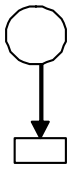


FIG. 2.3 – Transition puit

$W : ((S \times T) \cup (T \times S)) \rightarrow N = \{0, 1, 2, ..\}$ est la fonction de flux. On suppose que tous les réseaux de Petri sont finis, c'est à dire $|S \cup T| \in N$.

2.4 Concept de base :

2.4.1 Pré-ensemble et post-ensemble :

Pour tout réseau de Petri, le pré-ensemble d'un nœud x est l'ensemble des nœuds se trouvant à l'extrémité des arcs entrant à x, respectivement le post-ensemble d'un nœud x est l'ensemble des nœuds se trouvant à l'extrémité des arcs sortant de x. Formellement, pour tout $x \in S \cup T$, le pré-ensemble $^{\cdot}x$ est défini par $^{\cdot}x = \{y \in S \cup T / W(y, x) \neq 0\}$ et le post-ensemble $x^{\cdot}$ est défini par $x^{\cdot} = \{y \in S \cup T / W(x, y) \neq 0\}$.

Dans l'exemple de la figure 2.1 le pré-ensemble de $t_1$est $^{\cdot}t_1 = \{p_1\}$ on dit alors que $p_1$est en amont ou encore en entrée de t_1, et le post-ensemble de $t_1^{\cdot}$ est $t_1^{\cdot} = \{p_2\}$ et on dit que p_2 est en avale ou en sortie de t_1. De même pour les places, le pré-ensemble de p_1, $^{\cdot}p_1 = \{t_2\}$ et le post-ensemble de p_1, $p_1^{\cdot} = \{t_1\}$.

Cas particuliers :

- **Transition source :** pas de place en entrée de la transition, c'est-à-dire $^{\cdot}t = \varnothing$.

- **Transition puit :** pas de place en sortie de la transition, c'est-à-dire $t^{\cdot} = \varnothing$.

2.4.2 Notion de marquage :

On appel marquage M d'un réseau de Petri, un vecteur de nombre de marques (ou jetons) dans chaque place, de façon à ce que la $i^{éme}$ composante correspond à la $i^{éme}$ place. Un marquage permet de décrire l'état d'un réseau de Petri à un instant donné. Formellement, le marquage d'un réseau (S, T, W) est défini comme étant une fonction $M : S- > \mathbb{N}$.

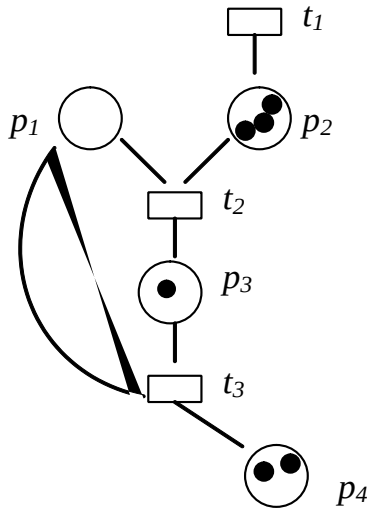


FIG. 2.4 – Machine de coupe de bois.

2.4.3 Réseau marqué :

un réseau marqué (S, T, W, M_0) est un réseau (S, T, W) avec un marquage initial M_0.

Exemple d'un réseau marqué : l'atelier de coupe de bois : [SB06]

Un atelier est constitué d'une machine de coupe et d'un stock. Quand une commande arrive et que la machine de coupe est disponible, la commande peut être traitée (découpée). Une fois le traitement terminé, la commande qui a été traitée est stockée. Sinon la commande doit attendre que la machine de coupe se libère avant de pouvoir être traitée. La modélisation de ce système est donnée par le réseau marqué de la figure 2.4.

Le nombre de jetons dans une place indique la valeur d'une variable d'état du système, par exemple les trois jetons dans p_2 représentent les trois commandes en attente. Mais parfois le nombre de jetons dans une place peut s'interpréter comme le nombre de ressources disponibles par exemple, dans la place p_1 un jeton indique qu'une machine est disponible, pas de jeton indique que la machine n'est pas disponible. Ces ressources sont consommées et ou produites au cours du temps. Par exemple dans la place p_2, le nombre de jetons indique le nombre de commandes à traiter par la machine de coupe. C'est-à-dire le nombre de ressources qui vont être consommées. Un jeton dans p_3 indique qu'une commande est entrain d'être traitée par la machine de coupe. Le nombre de jetons dans la place p_4 indique le nombre de commandes qui ont été traitées et stockées.

2.4.4 Changement d'état :

Le comportement d'un système peut être décrit en terme des états par lesquels il passe et les transitions entre ces états. Un changement d'état dans un réseau de Petri est décrit par le mouvement de jeton. Ce mouvement est causé par le franchissement d'une transition. Le franchissement d'une transition se fait selon la règle suivante :

- Une transition t est dite sensibilisée par le marquage M, si chacune des places contient aux moins $w(s,t)$ jetons où $w(s,t)$ est le poids de l'arc qui relie s à t. Formellement , t est sensibilisée par M si et seulement si $M(S) >= w(s,t)$ pour tout $s \in S$ [BD87][BDKP91].
- Une transition sensibilisée, peut être franchie immédiatement. Comme elle peut être franchie dès la satisfaction d'autres conditions tel que le cas d'autres variantes des réseaux de Petri [Mur89].
- Le franchissement d'une transition t se traduit par la suppression de $w(s,t)$ jetons de chacune des places en entrée de t et le rajout de $w(t,s)$ jetons à chacune des places en sortie de t , où $w(t,s)$ est le poids de l'arc qui relie t à s. Donc le franchissement d'une transition t produira un nouveau marquage $M^{'}$ défini par : $M^{'}(s) = M(s) - w(s,t) + w(t,s)$ pour tout $s \in S$. l'occurrence de t est notée $M[t > M^{'}$.

Remarque :

- Le franchissement d'une transition source se fait sans conditions.
- Une transition puit consomme des jetons et ne produit aucun nouveau jeton.

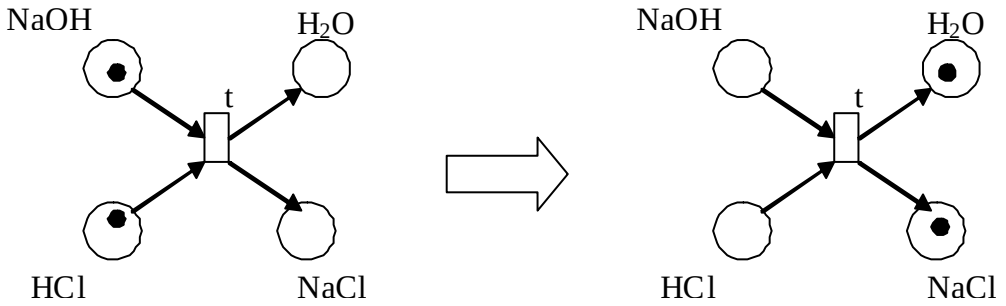
Exemple :

La règle de franchissement d'une transition est illustrée par la figure 2.5. Dans cet exemple nous considérons la réaction chimique suivante : $NaOH + HCl- > NaCl + H_2O$.

On suppose qu'au départ on a une particule de l'hydroxyde de sodium ($NaOH$) et une particule d'acide de clore (HCl). On obtient alors le réseau de Petri de la figure 2.5.(a). En remarque que le nombre de jetons dans chaque place correspond ici à la quantité de chaque produit chimique. Quand la réaction se produit, l'hydroxyde de sodium libère l'ion OH^- , cependant l'acide de clore libère l'ion H^+, ce qui donne l'eau (H_2O) et le sel ($NaCl$). Cette réaction chimique se traduit au niveau de notre modèle par la consommation d'un jeton de la place $NaOH$, et un autre de la place HCl et le rajout d'un jeton à la place H_2O, ainsi qu'un jeton à la place $NaCl$ (voir la figure 2.5.(b)).

2.4.5 Un système étiqueté :

Un alphabet $\mathcal{A}$ est un ensemble fini, tel que $\tau \notin \mathcal{A}$ (τ désignera l'action invisible, dite aussi action silencieuse).



(a) Avant réaction **(b) Aprés réaction**

FIG. 2.5 – Illustration de franchissement de transitions.

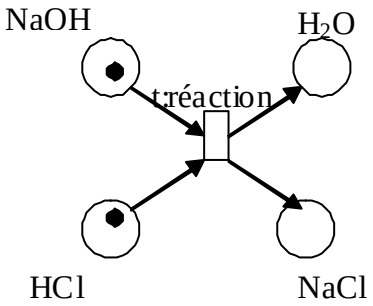


FIG. 2.6 – Système étiqueté

L'étiquetage d'un réseau $\mathcal{N} = (S, T, W)$ est une fonction $\lambda : T \to \mathcal{A} \cup \{\tau\}$ où $\tau \notin \mathcal{A}$. Si $\lambda(t) \in \mathcal{A}$ alors t est dite observable ou externe ; dans le cas contraire, t est dite silencieuse ou invisible.

$\Sigma = (S, T, W, M_0, \lambda)$ est un système étiqueté si et seulement si (S, T, W, M_0) est un réseau marqué et λ est une fonction d'étiquetage de (S, T, W).

Exemple :

Reprenons l'exemple de la figure 2.5, si on associe à la transition t l'action "réaction" , on obtient le système étiqueté de la figure 2.6.

2.5 Eléments de modélisation d'un système :

Les réseaux de Petri est un outil de modélisation très puissant apte de représenter plusieurs comportements. Dans cette section nous introduirons quelque concepts de base des réseaux de Petri, utilisables pour la modélisation des comportements des systèmes : séquence, parallélisme, synchronisation, indéterminisme, partage de ressource, mémorisation...etc. Ainsi nous donnerons quelques exemples illustratifs

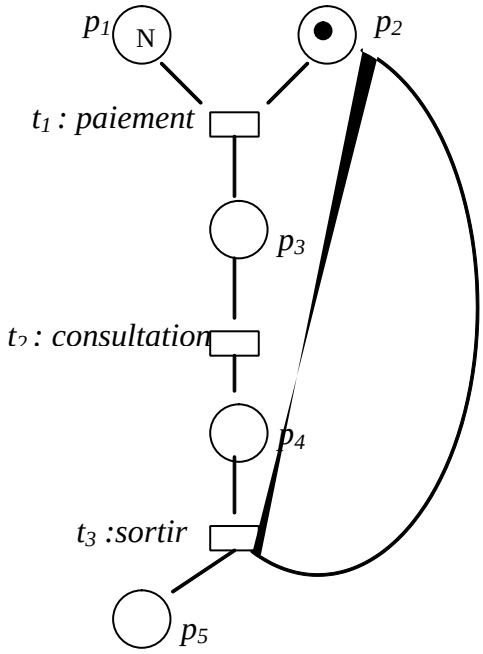


FIG. 2.7 – Système de consultation médicale

représentant une modélisation des systèmes réels qui appartiennent à des domaines diffèrents.

2.5.1 Séquence :

La séquence est l'un des comportements que l'on trouve dans la plupart des systèmes, elle représente une suite d'exécutions d'actions.

Formellement une séquence, notée $\sigma = M_0 t_1 M_1 t_2 \ldots$ est une séquence d'occurrence si et seulement si $M_{i-1}[t_i > M_i$ pour $i \succeq 1$.

Une séquence d'occurrence est une séquence $t_1 t_2 \ldots$de transition débutant par M si et seulement si : il existe une séquence d'occurence $M t_1 M_1 t_2 \ldots$

Si une séquence finie $t_1 t_2 .. t_n$ mène au marquage $M^{'}$ à partir du marquage M, on écrit $M[t_1 t_2 \ldots t_n > M^{'}$.

La figure 2.7 montre un exemple d'une exécution séquentielle d'un ensemble de transitions, il correspond à une modélisation du système de consultation de médecin dans une cabine médicale, on suppose que chaque malade doit fixer un rendez vous au préalable. A l'arrivée de son tour, il procède au paiement, consulte le médecin, prend l'ordonnance et sort.

2.5.2 Parallélisme :

Le parallélisme représente la possibilité que plusieurs activités s'exécutent simultanément au sein du même système [SB06], cette exécution concurrente ne peut

avoir lieu que lorsque ces activités sont causalement indépendantes. Cependant dans [Pet78], la concurrence a été vue comme une relation binaire notée *co* définie sur un ensemble d'évènements $A = \{e_1, e_2, ..e_n\}$: réflexive (c'est-à-dire e_1 *co* e_1 ce qui correspond à l'autoconcurrence), symétrique (c'est-à-dire e_1 *co* e_2 implique e_2 *co* e_1). Mais pas transitive, c'est-à-dire e_1 *co* e_2 et e_2 *co* e_3 n'implique pas e_1 *co* e_3. Par exemple : une personne peut conduire une voiture (e_1) et chanter (e_2) en même temps. Comme il peut chanter et marcher (e_3) en même temps, mais il ne peut pas conduire la voiture et marcher en même temps [Mur89]. De façon générale dans le contexte des réseaux de Petri, on peut dire qu'à partir d'un état donné (marquage M), deux transitions t_1, t_2 peuvent être franchie concurremment s'il y a suffisamment de jetons dans chaque place pour le franchissement de t_1 et t_2 en même temps, ceci se traduit formellement par $M(s) \succeq w(s, t_1) + w(s, t_2)$ pour tout $s \in S$.

Lors de la modélisation d'un système on peut provoquer le parallélisme de différentes manières : par le nombre de jetons, par une séparation de préconditions ou par une transition qui donne naissance à plusieurs branches parallèles. A titre illustratif nous considérons la machine à remplir et à boucher des bouteilles [SB06]. Cette machine se compose de trois postes qui travaillent en parallèles. Le poste 1 sert au transfert et au chargement : dans un premier temps, un vérin B permet de décaler le convoyeur d'une position vers la droite, ensuite le vérin A sert au chargement d'une nouvelle bouteille vide. Le poste 2 sert au remplissage des bouteilles à l'aide d'une vanne D. Quand au poste 3, il se charge du bouchage. Les actions de chargement, de remplissage et de bouchage d'une bouteille sont effectuées en parallèle. Cependant le transfert par le vérin B n'est effectué que lorsque les trois opérations se terminent. La modélisation de ce système en terme de réseau de Petri est donnée par la figure 2.8.

2.5.3 L'indéterminisme :

Parfois l'indéterminisme est une propriété intrinsèque dans le comportement du système que l'on doit prendre en compte. L'indéterminisme représente l'existence de plusieurs chemins, où chaque chemin correspond à un scénario possible dans le comportement du système. En général, la décision de prendre tel ou tel chemin dépend souvent, soit de l'état du système ou soit à des contraintes environnementales, c'est le cas des protocoles de communication. A titre d'exemple nous considérons un protocole de la classe sender-initiated où la source est responsable de la détection de la perte et de la retransmission du paquet perdu. Quand au récepteur il est chargé de signaler le succès de réception de chaque paquet de donnée. Ceci est illustré par la figure 2.9, la source envoie le paquet de donnée, garde une copie dans le cache et initialise un délai de garde. Dans ce cas, deux scénarios sont possibles : soit le récepteur reçoit le paquet, envoie l'acquittement et ce dernier est reçu par la source ; soit le délai de garde expire avant que la source reçoit l'acquittement, dans ce cas elle retransmet le paquet.

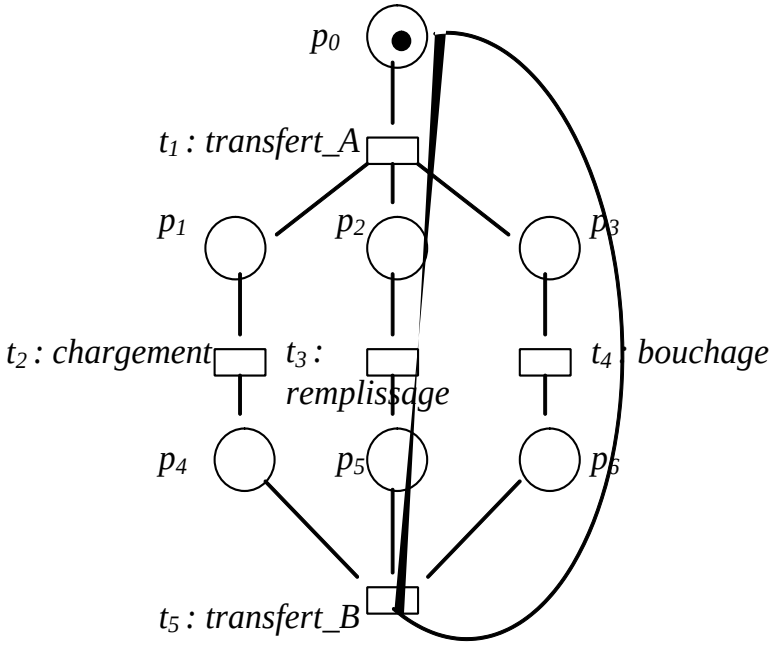


FIG. 2.8 – Machine à remplir et à boucher des bouteilles

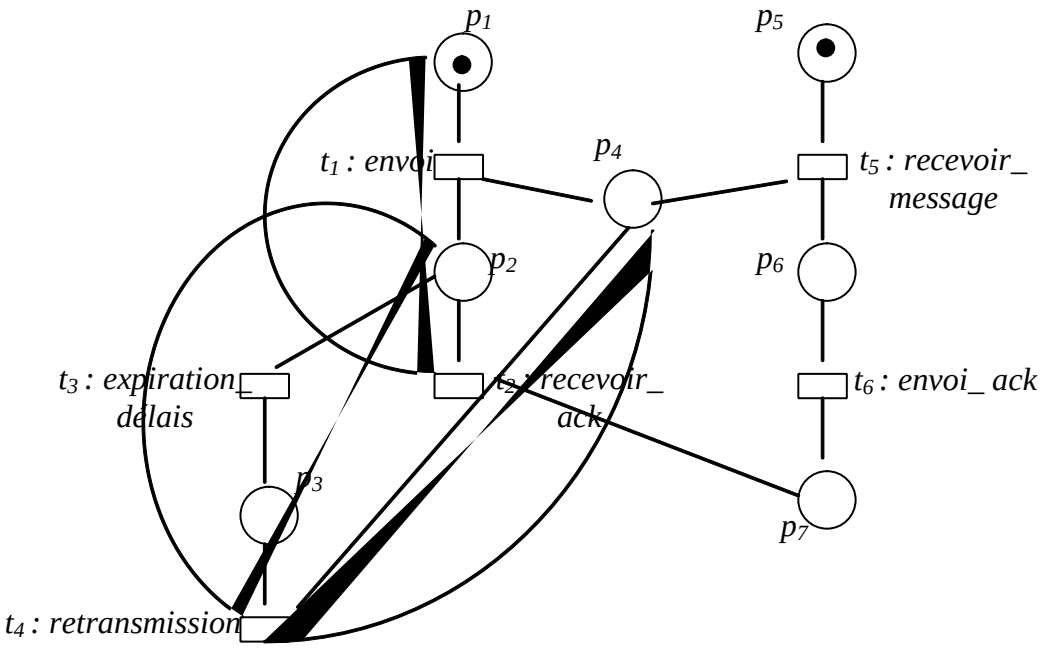


FIG. 2.9 – Protocole de la couche transport.

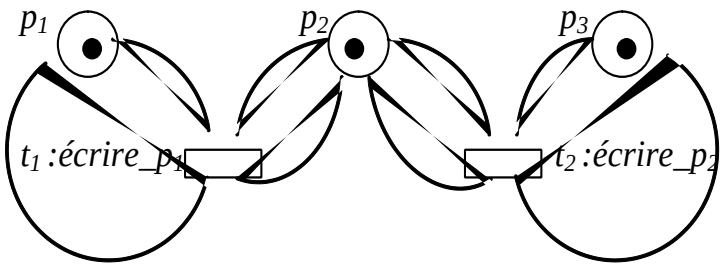


FIG. 2.10 – Deux rédacteurs qui partage une mémoire commune.

2.5.4 Synchronisation :

Souvent dans les systèmes, deux ou plusieurs processus évoluent indépendamment les uns des autres et se rencontrent sur un point donné pour réaliser une tâche commune. Ceci est appelée synchronisation. Par définition la synchronisation est un point de rendez-vous qui permet de coordoner les opérations de deux ou plusieurs processus [SB06]. L'exemple de la figure 2.8 illustre la synchronisation de trois processus. Nous remarquons que les processus de chargement, de remplissage et de bouchage évoluent indépendamment l'un de l'autre, cependant ces trois processus doivent se synchroniser sur la tâche t_5 parceque le transfert du vérin B n'est effectué que lorsque ces trois opérations se terminent.

2.5.5 Partage de ressource :

Dans les systèmes distribués, les processus peuvent s'exécuter concurremment, mais si ces processus utilisent une ressource ou une information commune non partageable, il faut contrôler son utilisation pour assurer le bon fonctionnement du système. Ce contrôle est appelé l'exclusion mutuelle. Nous avons déjà énoncé dans la section 2.4.3 que les jetons dans une place peuvent représenter le nombre de ressources disponibles dans le système. Donc la modélisation d'une ressource commune sera représentée par un jeton dans une place, cette dernière est reliée par un arc à chaque tâche qui l'utilise. De ce fait, on peut assurer que ces tâches soient conflictuelles. C'est-à-dire lorsque une tâche est entrain d'utiliser la ressource aucune autre tâche ne peut l'utiliser, d'où l'assurance du bon fonctionnement du système.

La figure 2.10 représente l'exemple classique de deux rédacteurs qui partagent une mémoire commune. Les places $p_1$et p_3 correspondent respectivement au processus $pross_1, pross_2$. Quand à la place p_2, elle représente la mémoire commune. Cette dernière joue le rôle de verrou. Elle est reliée par un arc vers la transition t_1 et un autre arc vers la transition t_2. Par conséquent les deux actions $écrire\_p_1$ et $écrire\_p_2$ sont conflictuelles.

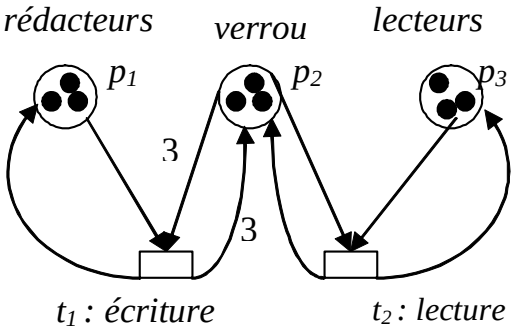


FIG. 2.11 – Modélisation de système lecteur / rédacteur

2.5.6 La mémorisation :

La mémorisation des informations sur les différentes tâches du système peut être modélisée par les réseaux de Petri. A titre d'exemple nous reconsidérons le réseau de Petri de la figure 2.4, on remarque que le nombre de jetons dans la place p_4 représente le nombre de commandes traitées et stockées. De même dans l'exemple 2.7, le nombre de jetons dans la place p_5 représente le nombre de malades qui ont consulté le médecin.

2.6 Approche de modélisation par les réseaux de Petri :

Afin de maîtriser la complexité des systèmes, lors de la modélisation, il est préférable d'adopter des approches permettant de procéder par étape. A ce niveau on distingue deux approches :

2.6.1 Approche par raffinement successif de transition :

La modélisation peut être vue comme un ensemble d'étapes obtenues par des raffinements successifs. Le raffinement est la transformation d'un réseau de Petri en un autre qui lui équivalent, en lui apportant des modifications dont le but d'introduire de nouvelles fonctionnalités. De ce fait, dans la première étape le système est décrit par un réseau de Petri simple, tout en considérant les transitions comme étant des tâches complexes. Par la suite toute transition complexe sera remplacé par le réseau de Petri qui décrit son comportement. Pour clarifier les idées on considère l'exemple classique des lecteurs/ rédacteurs. En premier lieu nous considérons deux tâches complexes : $lecture, écriture$ modélisées respectivement par les transitions $t_1$et t_2. La description du système est donnée par le réseau de Petri de la figure 2.11

Par la suite dans la deuxième étape on remplaçe respectivement les deux transitions *lecture* et *écriture* par les sous réseaux de Petri : $\{t_1 : requête\_d'écriture, p_4, t_2 : début\_écriture, p_5, t_3 : fin\_d'écriture\}$ et $\{t_4 : requête\_lecture, p_6, t_5 : début\_lecture,$

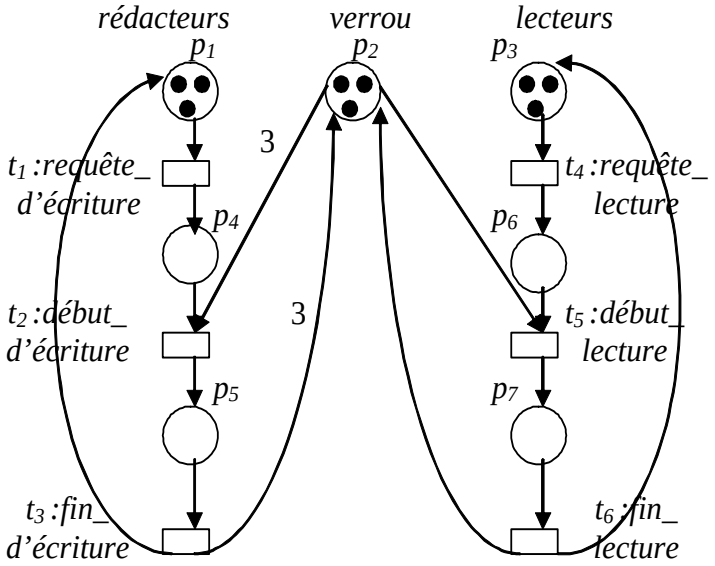


FIG. 2.12 – Modélisation détaillée du système des lecteurs/rédacteurs.

$p_7, t_6 : fin\_lecture\}$. On obtient alors un réseau de Petri plus détaillé, voir la figure 2.12. Remarquant que ce réseau de Petri donne une description détaillée du système. Chaque lecteur fait sa requête de lecture et attend l'obtention de l'autorisation de lecture, dans ce cas il commence sa lecture, lit dans la mémoire et à la fin il libère le verrou et regagne l'état prêt. La même chose pour les rédacteurs, chaque rédacteur fait sa requête d'écriture et rejoint la file d'attente jusqu'à ce que aucun lecteur ou rédacteur ne soit en cours. Dans ce cas là il commence l'écriture et à la fin il libère le verrou et regagne l'état prêt.

2.6.2 Approche par composition de réseaux de Petri :

Cette approche consiste en la décomposition d'un système complexe en des sous systèmes. Chaque sous système est représenté par un réseau de Petri. La modélisation d'un système global revient alors à l'assemblage de tous les réseaux de Petri correspondant aux sous systèmes. Ceci est fait soit par la fusion des places communes, où ces dernières modélisent en général les ressources communes, soit par la fusion des transitions communes entre les réseaux de Petri, celles ci représentent une synchronisation entre les différents sous systèmes.

Afin de mieux clarifier cette approche, nous considérons le système de coopération entre une société et un fournisseur. La société de production commande la matière première, dès sa réception elle la stocke dans un dépôt, sachant que cette matière sera transformée en un produit final. Notant qu'une commande n'est lancée que lorsqu'il y a rupture de stock. Concernant le fournisseur, il livre le produit dès qu'il reçoit la commande.

Dans notre modélisation le système est décomposé en deux entités, la société

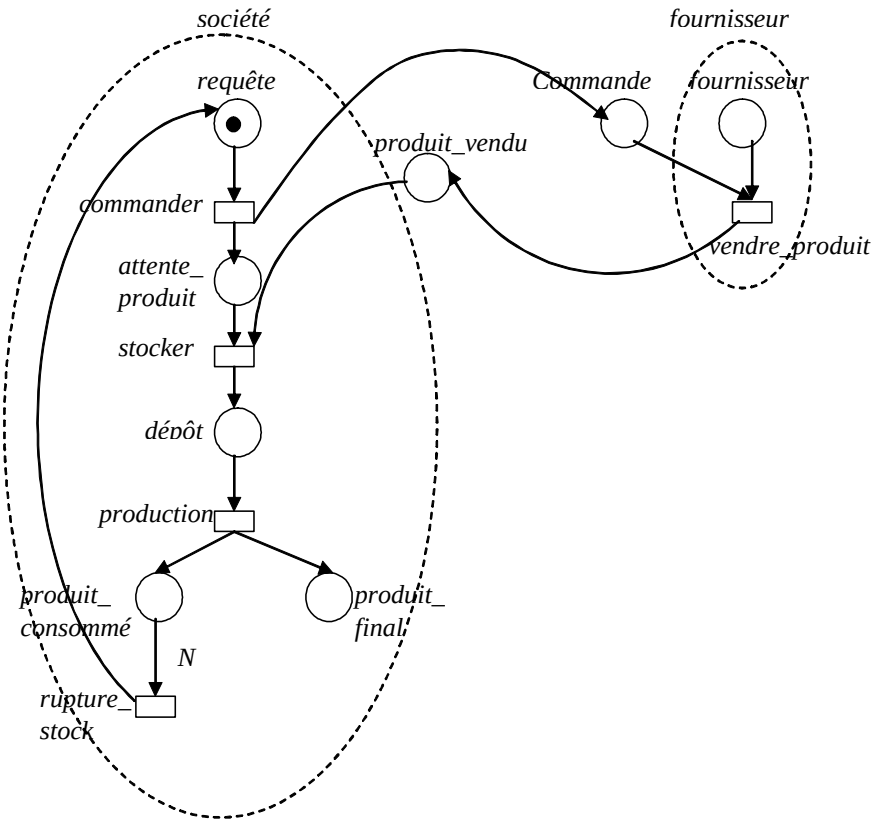


FIG. 2.13 – Modélisation du système société/ fournisseur.

et le fournisseur. Chaque entité est modélisée par un réseau de Petri séparé. La modélisation du système global est le résultat de l'assemblage des deux réseaux de Petri en fusionnant les places communes : $produit\_vendu, commande$.

2.7 Réseaux de Petri particuliers :

2.7.1 Réseau de Petri généralisé :

Un réseau de Petri généralisé est un réseau de Petri dans lequel les poids associés aux arcs sont des nombres entiers strictement positif. Si un arc (p_i, t_i) a un poids k la transition t_j n'est franchise que si la place p_i possède au moins k jetons. Le franchissement consiste à retirer k jetons de la place p_i. Si un arc (t_j, p_i) a un poids k : le franchissement de la transition rajoute k jetons à la place p_i.

2.7.2 Réseau de Petri à choix libre :

Un réseau de Petri à choix libre est un réseau dans lequel pour tout conflit $[p_i, \{t_1, t_2, \ldots, t_n\}]$ aucune des transitions $t_1, t_2, .., t_n$ ne possède de place d'entrée autre que p_i.

Exemple :

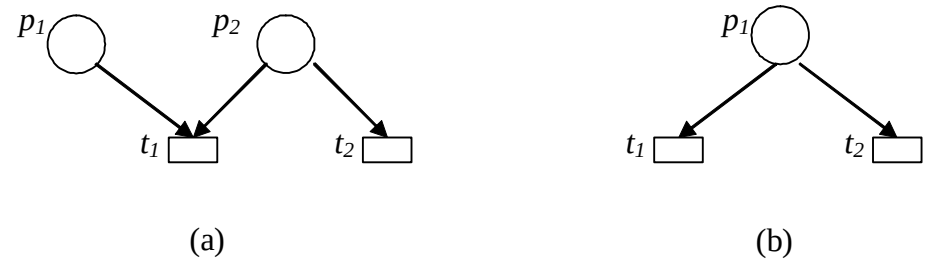


FIG. 2.14 – (a) Réseau sans choix libre. (b) Réseau à choix libre.

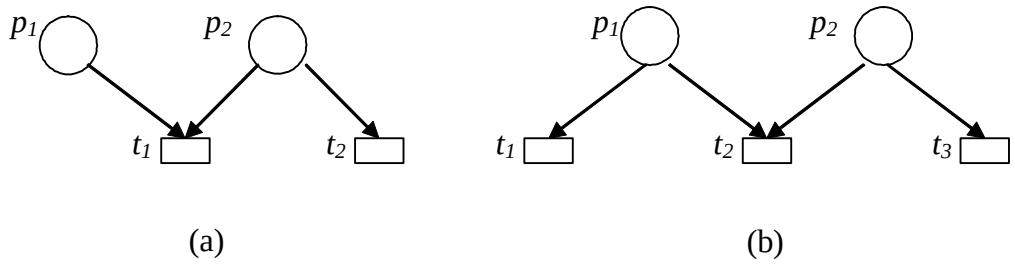


FIG. 2.15 – (a) Réseau simple. (b) Réseau non simple

Le réseau de la figure 2.14.(a) est sans choix libre car pour le conflit $[p_2, \{t_1, t_2\}]$ la transition t_1 possède en plus de la place p_2, la place d'entrée p_1. Quand au réseau de la figure 2.14.(b), il est à choix libre puisque le conflit $[p_1, \{t_1, t_2\}]$ ne possède aucune place d'entrée autre que p_1.

2.7.3 Réseau de Petri simple :

Un réseau de Petri est dit simple lorsque chaque transition ne peut être concernée par plus d'un conflit.

Exemple :

On remarque dans le réseau de la figure 2.15 .(a) que la transition t_1 est en conflit avec t_2 et réciproquement, donc ce réseau est simple car les transitions $t_1$et t_2 sont concernées par un seul conflit. Cependant dans le réseau de la figure 2.15.(b) on remarque que t_2 est en conflit avec t_1 et t_3 à la fois, par conséquent ce réseau n'est pas simple.

2.7.4 Réseau de Petri pur :

Dans un réseau de Petri, une transition est dite pure si elle ne possède aucune place qui est à la fois place d'entrée et place de sortie. Si toutes les transitions du réseau de Petri sont pures le réseau de Petri est dit pur.

Exemple :

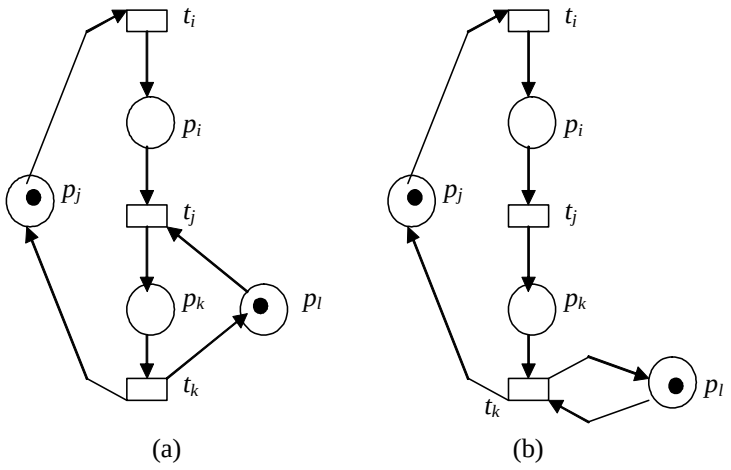


FIG. 2.16 – (a) Réseau pur. (b) Réseau impur.

Le réseau de Petri de la figure 2.16.(a) est pur car toutes les transitions sont pures, cependant celui de la figure 2.16.(b) n'est pas pur, puisque p_l est à la fois place d'entrée et place de sortie.

2.7.5 Réseau de Petri à capacité :

Dans un réseau de Petri ordinaire (ou généralisé) il n'y a aucune limitation quant au nombre de jetons pouvant être déposé dans une place. Pour la modélisation des systèmes, lorsqu'une place représente une ressource dont la capacité est limitée (par exemple : un stock pour lequel la capacité est limitée), il est nécessaire de pouvoir représenter cette propriété.

Dans un réseau de Petri à capacité, des capacités (nombre entier > 0) sont associées aux places. Le franchissement d'une transition d'entrée d'une place P_i, dont la capacité est $Cap(P_i)$, n'est possible que si le franchissement de cette transition ne conduit pas à un nombre de jetons dans P_i qui dépasse sa capacité.

Exemple :

Dans l'exemple suivant il semble évident que les transitions ne seront franchissables que lorsqu'elles ne conduisent pas à un dépassement de capacité égale à 2 pour la place $p_1$et 1 pour la place p_2. $\sigma = t_1t_3t_1t_1t_4$ est un exemple de séquence de transitions qui peut être franchie à partir de ce réseau de Petri.

2.7.6 Réseau de Petri à priorité :

Dans un tel réseau si on atteint un marquage à partir du quel plusieurs transitions sont franchissables, on doit franchir la transition qui a la plus grande priorité.

Exemple :

La figure 2.18 .(a) représente l'état initial d'un réseau de Petri, dans cet exemple

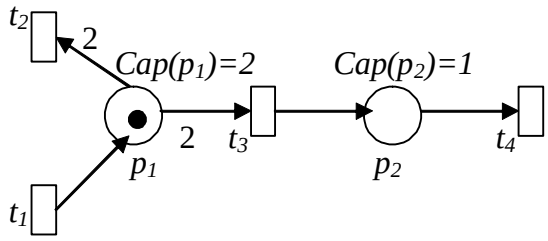
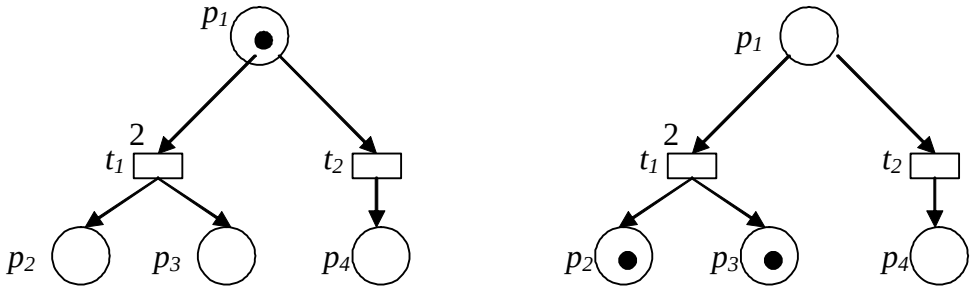


FIG. 2.17 – Réseau à capacité.



(a) Avant franchissement (b) Après franchissement

FIG. 2.18 – Réseau à priorité.

on a considéré que la transition t_1 est plus prioritaire que la transition t_2. On constate en premier lieu que les deux transitions t_1 et t_2 peuvent être franchie de manière indéterministe, mais comme on a supposé que t_1 a la plus grande priorité alors on doit franchir t_1. L'état du système après le franchissement de t_1 est donné par la figure 2.18 .(b).

Il à noter que dans le contexte de ce travail, notre étude est concerne pour les réseaux de Petri ordinaires.

2.8 Méthodes d'analyse d'un réseau de Petri :

Il existe trois méthodes principales d'analyse pour les réseaux de Petri : l'approche basée sur l'équation matricielle, l'approche basée sur l'arbre de couverture et l'approche basée sur la réduction.

Dans [Mur89], il a été montré que les deux premières approches sont efficaces pour l'étude des propriétés d'un réseau de Petri. Cependant dans certains cas elles se limitent à une sous classe spéciale de réseaux de Petri. Quand à l'arbre de couverture c'est une très bonne méthode pour détecter les marquages accessibles, néanmoins elle souffre du problème de l'explosion combinatoire du graphe d'état pour les systèmes

complexes. Dans cette section nous donnerons un aperçu sur ces trois approches.

2.8.1 Matrice d'incidence :

Le comportement dynamique de plusieurs systèmes peut être étudié en terme d'équations algébriques. Dans cette section nous nous intéressons à l'analyse des réseaux de Petri par les équations matricielles. On étudiera la représentation matricielle d'un réseau de Petri ainsi que l'équation fondamentale de ce système. Cette dernièrre est utilisable pour l'étude des propriétés comportementales.

2.8.2 Représentation matricielle :

Il est possible de représenter par des matrices la fonction de poids W [VNCG92] :

On appelle matrice d'incidence avant, nommée *pre*, la matrice de précondition de dimension (n, m) où n est le nombre de places et m est le nombre de transitions et $pre(i, j) = w(S_i, t_j)$.

On appelle matrice d'incidence arrière, nommée *post*, la matrice de postcondition de dimension (n, m) où n est le nombre de places et m est le nombre de transitions et $post(i, j) = w(t_j, S_i)$.

La matrice d'incidence $A = [a_{ij}]$ est une matrice (n, m) d'entiers tel que $a_{ij} = post[i, j] - pre[i, j]$.

Intuitivement $pre(i, j)$ indique le nombre de jetons que doit contenir chaque place pour le franchissement de la transition t_j. De façon duale, $post(i, j)$ contient le nombre de jetons déposées dans la place S_i suite à un franchissement de la transition t_j. Quand à a_{ij} , elle représente le changement final du nombre de jetons dans la place S_i aprés le franchissement de la transition t_j.

Exemple :

La représentation matricielle du réseau de la figure 2.7 est la suivante :

$$pre = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{array} \quad post = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \quad A = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{bmatrix} -1 & 0 & 0 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

Vecteur caractéristique d'une transition :

Le vecteur caractéristique de la transition t_i, noté $\bar{t}_i$, est défini comme suit [VNCG92] :

$t_i(j) = 0$ pour $j \neq i$ avec $1 \preceq j \preceq m$, où m est le nombre de transitions et $t_i(i) = 1$.

Soit M un marquage, si $pre \cdot \bar{t} = w(\cdot, t)$ où le "$\cdot$" dans $pre \cdot \bar{t}$ dénote le produit matricielle de pre et $\bar{t}$ alors, t est franchissable pour M. Le franchissement de t aboutit au marquage M' défini par : $M' = M + post \cdot \bar{t} - pre \cdot \bar{t} = M + A^T \cdot \bar{t}$.

Exemple :

Pour le réseau de Petri de la figure 2.7.

Le marquage initial $M_0 = \begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix}$, $\bar{t_1} = \begin{bmatrix}1\\0\\0\end{bmatrix}$

$$M' = M_0 + A^T \cdot \bar{t_1} = \begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix} + \begin{bmatrix}-1 & 0 & 0\\-1 & 0 & 1\\1 & -1 & 0\\0 & 1 & -1\\0 & 0 & 1\end{bmatrix}^T \begin{bmatrix}1\\0\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\\0\end{bmatrix}$$

Vecteur caractéristique d'une séquence :

Le vecteur caractéristique d'une séquence de franchissement S, noté $\bar{S}$, est le vecteur de N^m tel que : $\bar{S_j}$ est le nombre de franchissements de la transition t_j dans la séquence S [SB06].

Exemple :

Pour le réseau de Petri de la figure 2.7, et pour $S = t_1 t_2$

$\bar{S} = \begin{bmatrix}1\\1\\0\end{bmatrix}$

Equation fondamentale :

Si S est une séquence franchissable à partir du marquage Mi qui mène au marquage $M_k : (M_i[S > M_k)$ alors $M_k = M_i + A^T \cdot \bar{S}$ où $\bar{S}$ est le vecteur caractéristique associé à la séquence de franchissement S et A la matrice d'incidence [SB06].

Exemple :

Pour le réseau de Petri de la figure 2.7, la séquence $S = t_1 t_2 t_3$ est franchissable à partir de M_0, à quel marquage mène telle ?

Le vecteur caractéristique qui correspond à la séquence de franchissements $S = t_1 t_2 t_3$ est le suivant :

$\bar{S} = \begin{bmatrix}1\\1\\1\end{bmatrix}$

Soit M' le marquage accessible à partir de M_0, en franchissant la séquence $S = t_1 t_2 t_3$.

$$M' = M_0 + A^T \cdot \bar{S} = \begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix} + \begin{bmatrix}-1 & 0 & 0\\-1 & 0 & 1\\1 & -1 & 0\\0 & 1 & -1\\0 & 0 & 1\end{bmatrix}^T \cdot \begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix} + \begin{bmatrix}-1\\0\\0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\1\\0\\0\\1\end{bmatrix}$$

L'équation fondamentale est utilisable pour l'analyse des propriétés d'un réseau de Petri. Il est à noter que l'utilisation des équations ne permet pas l'analyse de toutes les propriétés d'un réseau de Petri. Ceci est dû principalement à la nature indéterministe de ce modèle.

2.8.3 Techniques de réduction pour l'analyse :

Le problème majeur des réseaux de Petri est sa complexité. Pour faciliter l'analyse de telles structures, une technique consiste à réduire le modèle du système à vérifier. Inversement, ces techniques sont utiles pour transformer hiérarchiquement un modèle abstrait à un autre plus raffiné [Mur89] :

Les six opérations suivantes permettent la réduction d'un réseau de Petri tout en préservant les propriétés du système à vérifier [SIL85][JM81] :

Fusion de places en série (voir la figure 2.19.(a))
Fusion de transitions en série (voir la figure2.19.(b))
Fusion de places parallèles (voir la figure2.19.(c))
Fusion de transitions parallèles (voir la figure 2.19.(d))
Elimination des boucles sur les places (voir la figure 2.19.(e))
Elimination des boucles sur les transitions (voir la figure 2.19.(f))

2.8.4 Graphe de marquage et arborescence de couverture :

Soit un réseau de Petri, M_0 étant son marquage initial. A partir de ce marquage on peut obtenir de nouveaux marquages égales au nombre de transitions franchises. Le résultat de ce processus est un graphe de marquages dont les nœuds représentent les marquages et dont les arcs représentent les transitions franchises menant le système d'un marquage à un autre. L'approche utilisant le graphe de marquages est utile lorsque le nombre de marquages accessibles est fini. Dans la cas où il est infini, on fait recourt au graphe dit arborescence de couverture fini. Dans ce cas, un symbole spécial ω est introduit, il désigne un nombre de jetons dans une place S_i , qui peut être indéterminé. Ce symbole a les propriétés suivantes :

1. $\forall n \in \mathbb{N}, \omega \succ n$
2. $\omega \pm n = \omega$
3. $\omega \succeq \omega$

L'algorithme suivant est un algorithme général qui permet de générer, suivant le cas, un graphe de marquages ou une arborescence de couverture [Mur89] :

Algorithme 2.1 *:*

- *Etape 1 : Mettre le marquage initial M_0 comme racine et l'étiqueter comme « nouveau ».*

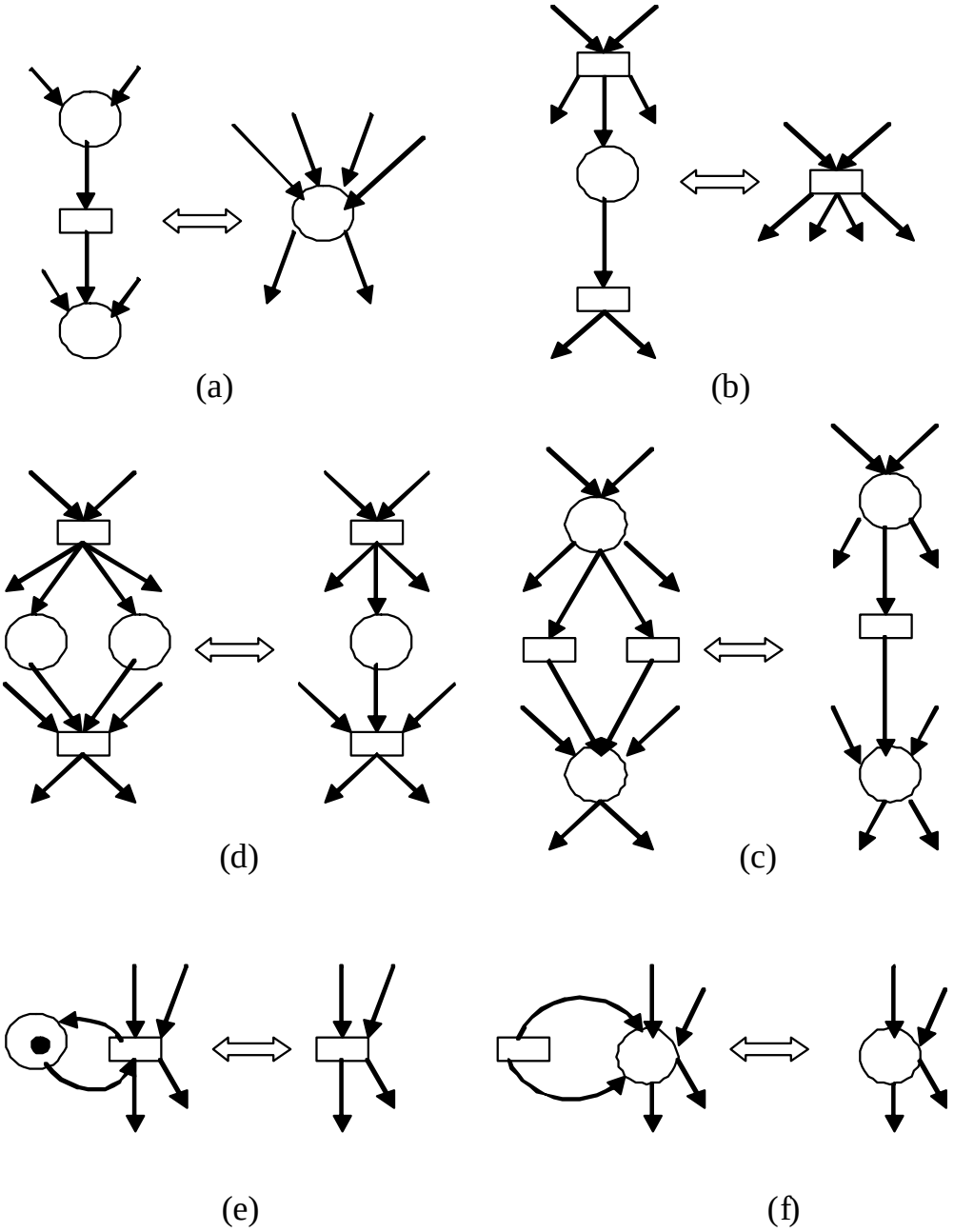


FIG. 2.19 – Les six opérations de réduction d'un réseau de Petri.

- *Etape 2 : Tant qu'il y a un « nouveau » marquage faire :*
  - *Etape 2.1 : Choisir un nouveau marquage M ;*
  - *Etape 2.2 : Si M est identique à un marquage se trouvant dans le chemin qui lie la racine à M alors étiqueter M comme « ancien » et choisir un autre nouveau marquage.*
  - *Etape 2.3 : Si aucune transition n'est franchissable à partir de M alors étiqueter M comme état d'interblocage «$dead-end$ ».*
  - *Etape 2.4 : Tant que des transitions sensibilisées par M existent, faire ce qui suit pour chaque transition t franchise à partir de M.*
    - *Etape 2.4.1 : Calculer $M^{'}$ résultat du franchissement de t à partir de M.*
    - *Etape 2.4.2 : S'il existe un marquage $M^{''}$ dans le chemin qui lie la racine vers M tel que $M^{'}(s) \succeq M^{''}(s)$ pour chaque place s et $M^{'} \neq M^{''}$ alors remplacer $M^{'}(s) > M^{''}(s)$.*
    - *Etape 2.4.3 :*
      · Introduire $M^{''}$ comme nœud.
      ·Dessiner un arc étiqueté par t de M vers $M^{'}$.
      ·Etiqueter $M^{'}$ comme nouveau marquage.

Exemple d'un système borné :

On considère un système informatique qui se compose de deux tâches cycliques T_1, T_2 qui partagent un processeur unique. Les tâches peuvent être soit en attente du processeur, soit en cours d'exécution. Nous faisons abstraction de la politique d'allocation du processeur. La modélisation de ce système est donnée par le réseau de Petri de la figure 2.20. Il est à noter que les places p_1, p_2 et p_3 correspondent à la tâche T_1 , le processeur et à la tâche T_2. Les places p_4 et p_5 signifient respectivement que T_1 est en cours d'exécution et T_2 est en cours d'exécution.

L'application de l'algorithme donne le graphe de marquages de la figure 2.21 :

Exemple d'arbre de couverture :

Soit l'exemple classique du producteur/consommateur qui partagent le même buffer. Sachant qu'un consommateur ne peut retirer un élément que lorsque le buffer est non vide. Ce système est modélisé par le réseau de Petri de la figure 2.22. La place p_1 correspond au producteur, la place p_3 correspond au consommateur. Cependant la place $p2$ représente le buffer. On remarque que la place p_2 est non bornée parceque le producteur peut produire un nombre infini d'éléments, d'où la construction de l'arbre de couverture de la figure 2.23.

2.9 Conclusion :

Dans ce chapitre nous avons présenté le modèle des réseaux de Petri. La première partie a introduit les différents concepts de bases nécessaires pour la compréhention

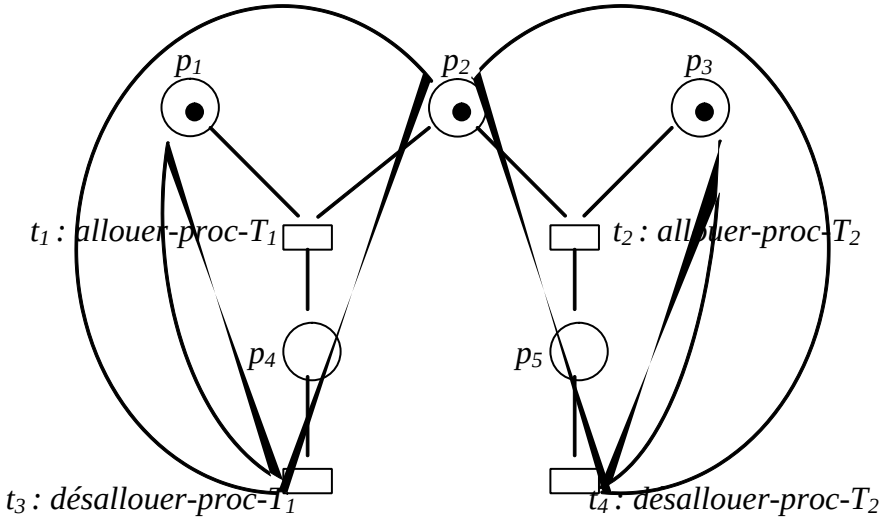


FIG. 2.20 – Allocation du processeur.

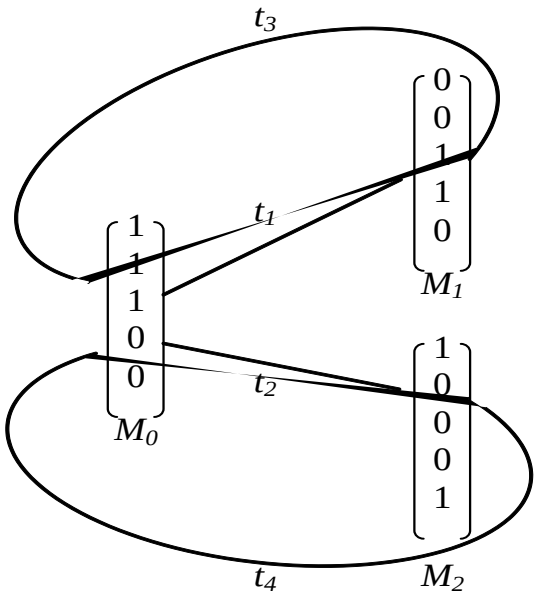


FIG. 2.21 – Graphe de marquages

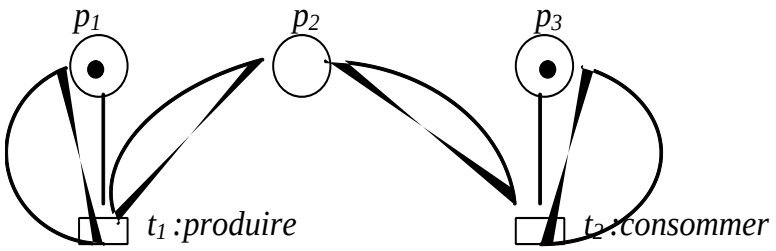


FIG. 2.22 – L'exemple du producteur/consommateur.

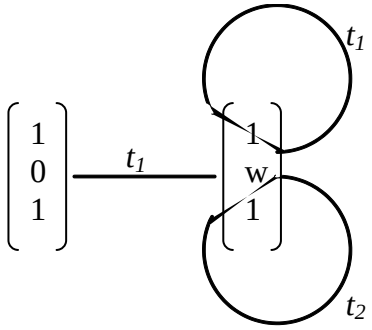


FIG. 2.23 – Arbre de couverture.

de ce modèle. A travers des exemples simples, nous avons montré que ce modèle est un modèle puissant qui peut intervenir dans différents domaines et apte de représenter différents comportements : séquencement, parallèlisme, indéterminisme, synchroniation...etc. Par la suite nous avons présenté quelques cas particuliers des réseaux de Petri. Cependant la dernière partie a été consacrée à la présentation de trois approches d'analyse d'un réseaux de Petri. L'approche basée sur la matrice d'incidence et l'équation d'état, l'approche de réduction pour l'analyse et l'approche basée sur la construction de l'arbre de couverture. Les deux premières approches sont efficaces, mais elle sont limitées à une sous classe des réseaux de Petri. La troisième approche est utile pour l'étude de l'accessibilité, néanmoins elle souffre du problème de l'explosion combinatoire du graphe d'états.

Chapitre 3

Système de transitions étiquetées maximales :

3.1 Problématique :

Durant les vingt dernières années, plusieurs sémantiques du parallélisme ont été étudiées dans le cadre de la théorie de la concurrence [Sai05]. Comme leur nom l'indique, elles ont été utilisées pour donner une sémantique du parallélisme aux modèles de spécification, permettant de fournir une base pour différentes notions et définitions d'équivalences de comportements. L'une de ces sémantiques est la sémantique d'entrelacement. Dans les modèles d'entrelacement, l'exécution en parallèle de deux actions est représentée par l'exécutions entrelacées dans le temps de ces deux actions, c'est-à-dire par leurs exécutions alternées [SC]. Parmi ces modèles nous pouvons citer le modèle des systèmes de transitions étiquetées, celui-ci permet de représenter les systèmes concurrents à modéliser à l'aide de graphes orientés, dont les sommets et les arcs représentent respectivement les états et les transitions. Ces états et transitions sont associés avec des chaînes de caractères de manière à pouvoir distinguer entre eux. Ces chaînes de caractères s'appellent noms lorsqu'elles sont associées aux états et étiquettes lorsqu'elles sont associées aux transitions.

De manière générale on peut dire que la sémantique d'entrelacement est très attrayante vue sa simplicité, néanmoins son utilisation impose l'hypothèse d'atomicité structurelle et temporelle des actions, ceci est parfois impossible, surtout qu'une étude précise d'un système nécessite une projection effective du système réel sur le modèle théorique. A titre d'exemple nous considérons deux machines de distribution de café $M1$ et $M2$. La machine $M1$ est une machine séquentielle, à la commande de deux tasses de café elle les fournit d'une manière séquentielle. La machine $M2$ est une machine parallèle qui peut délivrer les deux tasses de café simultanément (voir la figure 3.1). En appliquant la sémantique d'entrelacement, on obtient le même système de transitions étiquetées donné par la figure 3.2. Donc, de point d'un vue de la sémantique d'entrelacement, ces deux machines ne sont pas distinguables, ceci est valable sous l'hypothèse que la livraison des tasses est instantanée. Malheureu-

FIG. 3.1 – Machines de distribution de café

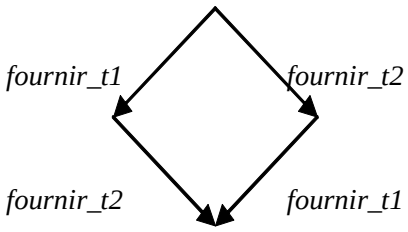


FIG. 3.2 – Système de transitions étiquetées pour les machines M1 et M2.

sement, dans la réalité ceci n'est pas le cas, il semble évident que ces deux machines sont complètement différentes surtout que l'action de distribution d'une tasse de café dure dans le temps. Supposant que la durée de l'action de livraison d'une tasse de café est égale à d, il est clair que la machine $M2$ peut terminer la tâche dans une durée d. Cependant la machine $M1$ ne peut achever la tâche qu'après un temps $2d$. En d'autres termes, les deux machines n'ont pas les mêmes performances. Une autre manière d'observer la différence entre les deux machines consiste en la substitution de l'action de livraion d'une tasse de café par deux actions (raffinement des actions fournir_t1, fournir_t2 respectivement par le processus $a1$ suivie de $a2$ et le processus $a3$ suivie de $a4$). Les systèmes de transitions étiquetées résultants des machines $M1$ et $M2$ sont respectivement donnés par les figures 3.3 .(a) et 3.3.(b). On peut constater que dans le système de transitions étiquetées correspondant à la machine $M2$, l'exécution de l'action $a3$ peut survenir entre $a1$ et $a2$, cependant ceci n'est pas le cas dans la machine $M1$.

Dans le but de lever l'hypothèse d'atomicité des actions, plusieurs sémantiques ont été définies dans la littérature. Un premier intérêt de ces sémantiques est de permettre une conception hiérarchique des systèmes par remplacement d'actions, considérées comme des entités entières dans un certain niveau d'abstraction, par des

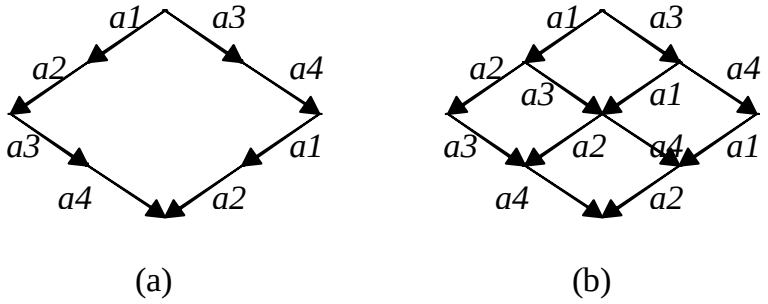


FIG. 3.3 – Système de transition étiquetée pour les systèmes rafinés.

processus dans le niveau d'abstraction inférieur[1]. Un deuxième intérêt, moins apparent de ces sémantiques est la caractérisation des possibilités d'exécutions parallèles des actions non instantanées dans le comportement d'un système. Parmi ces sémantiques on peut citer la sémantique de causalité, la ST-sémantique et la sémantique de maximalité [Saï96].

3.2 Sémantique de causalité :

Dans l'approche basée sur la causalité les transitions sont des évènements qui correspondent aux occurrences d'actions. A chaque transition sont associés :

- Le nom d'un évènement.
- Le nom d'action.
- L'ensemble des noms des évènements qui ont conditionné l'occurrence de cette transition.

Parmi les modèles basés sur cette sémantique nous pouvons citer : les arbres causaux [DD89], les arbres causaux dynamiques [Saï96] et les systèmes de transitions étiquetées causaux [Coe93].

3.2.1 Arbres causaux :

Les arbres causaux ont été définis par Darondeau et Degano [DD89]. Dans un arbre causal, à chaque transition observable est associé un ensemble de références. Une référence est un entier naturel qui correspond au nombre de transitions observables qui sépare cette transition de la transition qui la causée.

Pour illustrer l'intuition derrière les arbres causaux prenant l'exemple des machines de distribution de café. Les arbres causaux qui représentent les comportements des machines $M1$ et $M2$, sont donnés respectivement par les figures 3.4 .(a) et 3.4.(b).

[1] Des hypothèses sont à considérer sur les processus de raffinement. Pour des études détaillées sur le raffinement d'action, le lecteur est invité à consulter les références sus-citées.

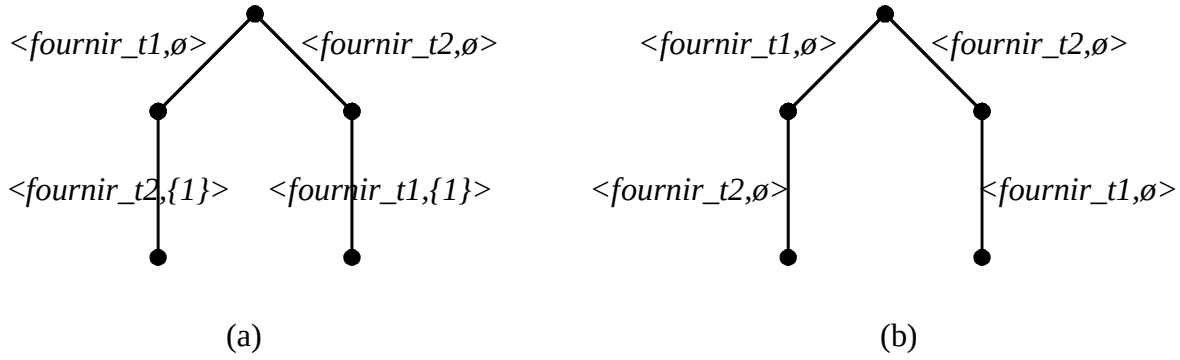


FIG. 3.4 – Intuition derrière les arbres causaux

3.2.2 Systèmes de transitions étiquetées causaux :

Le modèle des systèmes de transitions étiquetés causaux a été introduit par Rosvelter Coelho da Costa [Coe93]. Il peut être considéré comme un modèle de vrai parallélisme très proche du modèle des arbres causaux [DD89]. Il a été utilisé pour exprimer la sémantique de CCS [CS92][Coe93], d'un sous ensemble de LOTOS [CC91][CC93], des réseaux de Petri [CC92] et pour l'étude du raffinement d'actions dans LOTOS [CS92][CS94].

3.2.3 Les arbres causaux dynamiques :

On peut considérer que les arbres causaux dynamiques sont un enrichissement des arbres causaux. En effet, un nom d'évènement est associé à chaque transition de l'arbre causal, qui identifie son occurrence, les noms de ces évènements peuvent être alors utilisés pour faire référence aux causes des transitions.

Pour illustrer le principe de ce modèle, on reprend l'exemple précédent des machines de distribution de café. L'arbre causal qui correspond à la machine $M1$ (respectivement à la machine $M2$) est donné par la figure 3.5.(a) (respectivement par la figure 3.5.(b)).

Ce modèle a été introduit par Saidouni et Al dans [Saï96] dont le but d'une étude comparative, cette dernière a clarifié le résultat de Van Glabbeek qui a montré à travers des exemples que les sémantiques d'ordre partiel ne sont ni nécessaires ni suffisantes lorsque les actions ne sont pas atomiques ou bien qu'elles ont des durées non nulles [van90a].

3.3 ST-Sémantique :

L'une des sémantiques les plus intéressantes vis à vis de l'hypothèse d'atomicité des actions est la ST-sémantique qui a été définie pour la première fois sur les réseaux de Petri [vV87] et généralisée par la suite aux structures d'évènements primaires

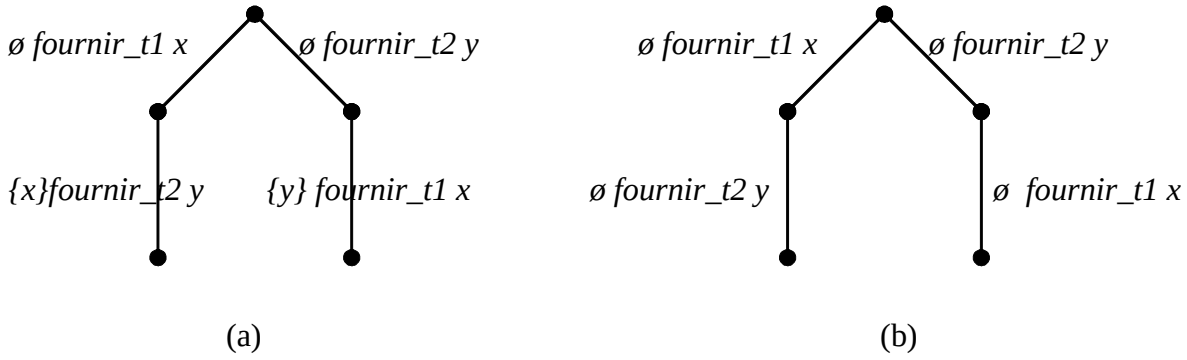


FIG. 3.5 – Intuition derrière les arbres causaux dynamiques.

[van90b]. L'utilisation de la ST-sémantique a permi la définition de relations de bissimulation qui sont les plus appropriées lorsque les actions ne sont pas atomiques [Vog91][Vog93], dans le sens où elles sont les plus larges relations préservées par le raffinement d'actions [Saï96].

L'idée principale de la ST-sémantique pour capturer les évolutions parallèles dans un système consiste à éclater toute action non instantanée en deux actions consécutives, la première désigne son début et la deuxième désigne sa fin. Un état du système dans lequel seules certaines actions ont débuté leur exécution signifie que ces actions sont en exécution parallèle dans cet état là. Pour mieux clarifier le principe de cette sémantique, on va reprendre l'exemple des machines de distribution de café. Les figure 3.6 .(a) et 3.6.(b) correspondent respectivement aux arbres de dérivations des machines $M1$ et $M2$.

L'avantage principal de la ST-sémantique est évidemment l'exploitation de la simplicité des modèles d'entrelacement ainsi que tous les résultats et travaux qui ont été déjà réalisés sur ces modèles [Saï96]. Mais d'un point de vue pratique, cette technique contribue de manière significative à l'explosion combinatoire du graphe d'état [CS94][CS95].

3.4 Sémantique de maximalité :

La présence des relations de causalité ou d'indépendance entre les occurrences d'actions dans les modèles de vrai parallélisme a conduit à la définition de la sémantique de maximalité [Saï96][SC].

L'avantage de cette sémantique réside dans le fait que d'une part elle a le même pouvoir d'expression que la ST-sémantique et d'autre part elle échape au problème de l'explosion combinatoire du graphe d'états induit par l'éclatement des actions en débuts et fins.

Dans l'approche basée sur la maximalité, les transitions sont des évènements qui ne représentent que le début de l'exécution des actions.

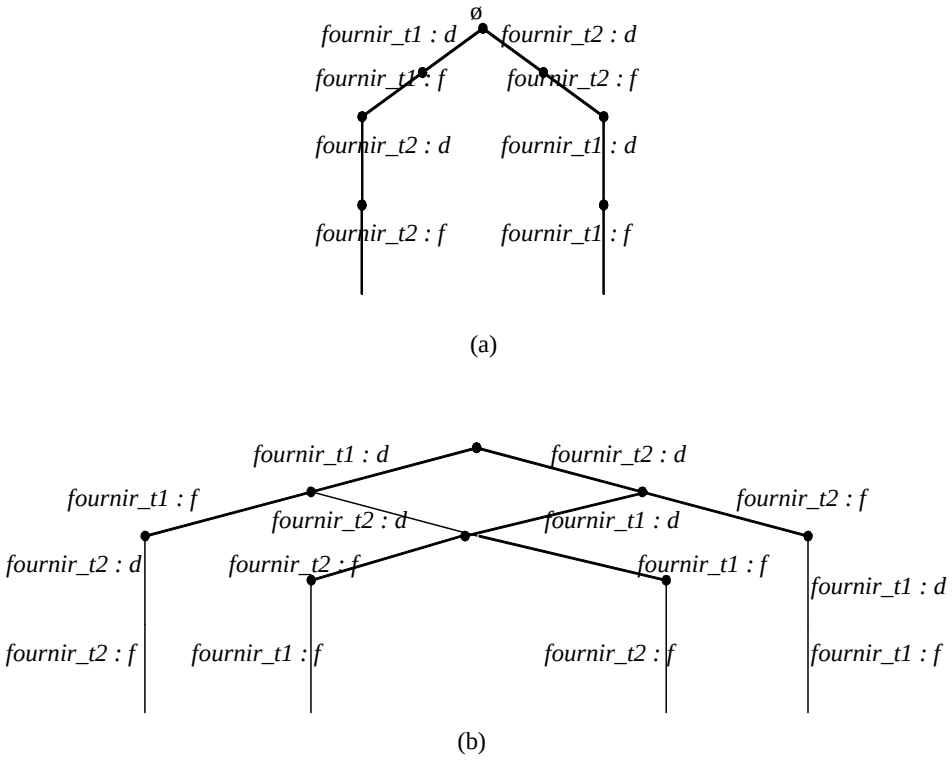


FIG. 3.6 – Arbres de dérivation basés sur la ST-sémantique

Etant donnée que plusieurs actions qui ont le même nom peuvent s'exécuter en parallèle (auto-concurrence), on associe, pour distinguer les exécutions de chacune des actions, un identificateur à chaque début d'exécution d'action, c'est-à-dire à la transition ou à l'évènement associé. Dans un état, un évènement est dit maximal s'il correspond au début de l'exécution d'une action qui peut éventuellement être toujours en train de s'exécuter dans cet état là.

L'applicabilité de la sémantique de maximalité à des modèles réels a déjà été démontrée plusieurs fois, sur les réseaux de Petri par Deviller en 1992 [Dev92b]. Il a définie un modèle basé sur la notion de processus. Ce dernier est considéré un bon modèle pour l'étude des relations d'équivalence, mais il reste un modèle théorique non implémentable. Cette sémantique a été appliquée sur les algèbres des processus [CS95]. Dans ce qui suit nous allons présenté le modèle des systèmes de transitions étiquetées maximales [SBB08b].

3.4.1 Systèmes de transitions étiquetées maximales :

Dans un système de transitions étiquetées maximales chaque état est étiqueté par un ensemble de noms d'événements. Chaque nom d'événement identifie le début d'exécution d'une action qui s'est produite avant cet état et que cette action peut être encore en cours d'exécution. Une transition entre deux états Si et Sj est étiquetée par un triplet (M, a, x) (noté ${}_M a_x$) ; x étant le nom d'événement identifiant le début d'exécution de l'action a, M désigne l'ensemble des noms d'événements représentant des causes de l'action a, les éléments de M appartiennent à l'état Si. L'occurrence de cette transition met fin aux actions identifiées par l'ensemble M, de ce fait, l'ensemble des noms d'événements associé à l'état Sj n'est autre que celui de l'état Si à qui on soustrait l'ensemble M et on rajoute le nom d'événement x. La définition formelle d'un système de transitions étiquetées maximales est comme suit :

Définition 3.1 *$\mathcal{M}$ étant un ensemble dénombrable de noms d'événements, un système de transitions étiquetées maximales de support $\mathcal{M}$ est un quintuplet $(\Omega, \lambda, \mu, \xi, \psi)$ avec :*

- *$\Omega = \langle S, T, \alpha, \beta, s_0 \rangle$ est un système de transitions tel que :*
  - *S est l'ensemble des états dont lesquels peut se trouver le système, cet ensemble peut être fini ou infini.*
  - *T est l'ensemble de transitions indiquant le changement d'état que peut réaliser le système, cet ensemble peut être fini ou infini.*
  - *α et β sont deux applications de T dans S tel que pour toute transition t on a : $\alpha(t)$ est l'origine de la transition t et $\beta(t)$ son but.*
  - *s_0 est l'état initial du système de transitions Ω.*
- *(Ω, λ) est un système de transitions étiqueté par la fonction λ sur un alphabet A appelé support de (Ω, λ). $(\lambda : T \to A)$.*

- $\psi : S \rightarrow 2^{\mathcal{M}}_{fn}$ est une fonction qui associe à chaque état l'ensemble fini des noms des événements maximaux présents au niveau de cet état.
- $\mu : T \rightarrow 2^{\mathcal{M}}_{fn}$ est une fonction qui associe à chaque transition l'ensemble fini des noms des événements correspondant aux actions qui ont commencé leur exécution et dont la terminaison sensibilise cette transition.
- $\xi : T \rightarrow \mathcal{M}$ est une fonction qui associe à chaque transition le nom de l'événement qui identifie son occurrence.

Definition 1 *tel que* $\psi(s_0) = \emptyset$ *et pour toute transition* t, $\mu(t) \subseteq \psi(\alpha(t))$, $\xi(t) \notin \psi(\alpha(t)) - \mu(t)$ *et* $\psi(\beta(t)) = (\psi(\alpha(t)) - \mu(t)) \cup \{\xi(t)\}$.

Notation 3.1 *Soit* $stem = (\Omega, \lambda, \mu, \xi, \psi)$ *un système de transitions étiquetées maximales tel que* $\Omega = \langle S, T, \alpha, \beta, s_0 \rangle$. $t \in T$ *étant une transition tel que* $\alpha(t) = s$, $\beta(t) = s'$, $\lambda(t) = a$, $\mu(t) = E$ *et* $\xi(t) = x$. *La transition* t *sera notée aussi* $s \xrightarrow{E a_x} s'$.

3.4.2 Intuition derrière la sémantique de maximalité :

Pour illustrer la sémantique de maximalité, on considère les expressions de comportement des machines M_1et M_2 écrite en Basic LOTOS :

$M_1 = \ fournir\_t1; fournir\_t2; stop[]fournir\_t2; fournir\_t1; stop$

$M_2 = fournir\_t1; stop|||fournir\_t2; stop$

Dans l'état initial, aucune action n'a encore été exécutée, donc l'ensemble des évènements maximaux est vide, d'où les configurations initiales suivantes associées à M_1et M_2 : $_{\varnothing}[M_1]$ et $_{\varnothing}[M_2]$. En appliquant la sémantique de maximalité, les transitions suivantes sont possibles :

$_{\varnothing}[M_2] \overset{\varnothing\ fournir\_t1x}{\longrightarrow_m}_{\{x\}} [stop]|||_{\varnothing}[fournir\_t2; stop] \overset{\varnothing\ fournir\_t2y}{\longrightarrow}_{\{x\}} [stop]|||_{\{y\}}[stop]$

x (respectivement y) étant le nom de l'évènement identifiant le début de l'action $fournir\_t1$ (respectivement $fournir\_t2$). Etant donné que rien ne peut être conclu à propos de la terminaison des deux actions $fournir\_t1$ et $fournir\_t2$ dans la configuration $_{\{x\}}[stop]|||_{\{y\}}[stop], x$ et y sont alors maximaux dans cette configuration. Notant que x est également maximal dans l'état intermédiaire représenté par la configuration $_{\{x\}}[stop]|||_{\varnothing}[fournir\_t2; stop]$.

Pour la configuration initiale associée à l'expression de comportement M_1, la transition suivante est possible : $_{\varnothing}[M_1] \overset{\varnothing\ fournir\_t1x}{\longrightarrow_m}_{\{x\}} [fournir\_t2; stop]$. Comme précédemment x identifie le début de l'action $fournir\_t1$ et il est le nom du seul évènement maximal dans la configuration $_{\{x\}}[fournir\_t2; stop]$. Il est clair que, au vu de la sémantique de l'opérateur de préfixage, le début de l'exécution de l'action $fournir\_t2$ n'est possible que si l'action $fournir\_t1$ termine son exécution. Par conséquent, x ne reste plus maximal lorsque l'action $fournir\_t2$ commence son exécution. L'unique évènement maximal dans la configuration résultante est donc celui identifié par y, qui correspond au début de l'exécution de l'action $fournir\_t2$. L'ensemble des noms des évènements maximaux a donc été

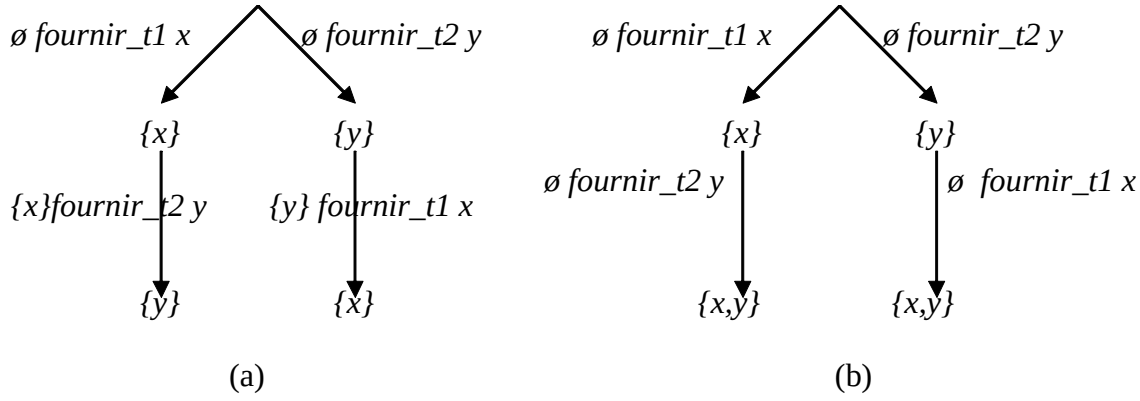


FIG. 3.7 – Arbres de dérivations obtenus par la sémantique de maximalité.

modifié par la suppression de x et l'ajout de y, ce qui justifie la dérivation suivante ${}_{\{x\}}[fournir\_t2; stop] \overset{\{x\}fournir\_t2y}{\longrightarrow_m} {}_{\{y\}}[stop]$.

La configuration ${}_{\{y\}}[stop]$ est différente de la configuration ${}_{\{x\}}[stop]|||_{\{y\}}[stop]$ car la première ne possède qu'un seul évènement maximal (identifié par y), alors que la deuxième en possède deux (identifiés par x et y). Les arbres de dérivation des expressions de comportement M_1et M_2 obtenus par l'application de la sémantique de maximalité sont représentés par la figure 3.7.

3.4.3 Notion de configuration :

Définition 3.2 *L'ensemble des noms des évènements est un ensemble dénombrable noté $\mathcal{M}$. Cet ensemble est parcouru par...$x, y, z.M, N$,..dénotent des sous ensembles finis de $\mathcal{M}$. L'ensembles des atomes de support Act est $Atm = 2^{\mathcal{M}}_{fn} \times Act \times \mathcal{M}, 2^{\mathcal{M}}_{fn}$ étant l'ensemble des parties finies de $\mathcal{M}$; Pour $M \in 2^{\mathcal{M}}_{fn}, x \in \mathcal{M}$ et $a \in Act$, l'atome (M, a, x) est noté ${}_M a_x$. Etant donné l'atome ${}_M a_x$, les fonctions suivantes sont définies :*

$$\mu({}_M a_x) = M \qquad |{}_M a_x| = a \qquad \zeta({}_M a_x) = x$$

Définition 3.3 *Soit $get : 2^{\mathcal{M}}_{fn} - \{\varnothing\} \longrightarrow M$ une fonction satisfaisant $get(M) \in M$ pour tout $M \in 2^{\mathcal{M}}_{fn} - \{\varnothing\}$. Pour clarifier les idées, donnons un exemple de la fonction get satisfaisant la définition précédente. Etant donné que l'ensemble$\mathcal{M}$ est dénombrable, il existe une bijection h entre $\mathcal{M}$ est l'ensemble des entiers naturels $\mathbb{N}$. Par conséquent, la fonction get peut être définie de la façon suivante : $get(M) = x$ tel que $h(x)$ est le plus petit élément dans $h(M)$, où $h(M)$ est l'image de M dans $\mathbb{N}$ par la fonction h.*

Définition 3.4 *L'ensemble des fonctions de substitution des noms des évènements est Subs (ie. $Subs = \mathcal{M} \longrightarrow 2^{\mathcal{M}}_{fn}$); $\sigma, \sigma_1, \sigma_2, ...$désignent les éléments de Subs. Etant donnés $x, y, z \in \mathcal{M}$ et $M \in 2^{\mathcal{M}}_{fn}$, alors*

- *l'application de* σ *à* x *sera écrite* σx;
- *la substitution identité* ι *est définie par* $\iota x = \{x\}$;
- $M\sigma = \cup_{x \in M} \sigma x$;
- $\sigma[y/z]$ *est une fonction de substitution définie par :* $\sigma[y/z]x = \begin{cases} \{y\} \ si \ z = x \\ \sigma x \ sinon \end{cases}$

Définition 3.5 *L'ensemble* C *des configurations des expressions de comportements de Basic LOTOS est le plus petit ensemble défini par induction comme suit :*

- $\forall E \in B, \forall M \in 2^{\mathcal{M}}_{fn} :_M [E] \in C$
- $\forall P \in PN, \forall M \in 2^{\mathcal{M}}_{fn} :_M [P] \in C$
- *Si* $\varepsilon \in C$ *alors hide* L *in* $\varepsilon \in C$
- *Si* $\varepsilon \in C$ *et* $F \in B$ *alors* $\varepsilon >> F \in C$
- *Si* $\varepsilon, \mathcal{F} \in C$ *alors* ε *op* $\mathcal{F} \in C$ $\quad op \in \{[], |||, ||, |[L]|, [>\}$
- *Si* $\varepsilon \in C$ *et* $\{a_1, ..., a_n\}, \{b_1, ..., b_n\}$ *deux sous ensembles de* G *alors* $\varepsilon[b_1/a_1, ..., b_n/a_n] \in C$

Etant donné un ensemble $M \in 2^{\mathcal{M}}_{fn}, {}_M[...]$ est appelée opération d'encapsulation. Cette opération est distributive par rapport aux opérations $[], |[L]|, hide, [>$ et le renommage de portes.

Définition 3.6 *Une configuration est dite canonique si elle ne peut plus être réduite par la distribution de l'opérateur d'encapsulation sur les autres opérateurs.*

Par exemple la configuration ${}_{\{x,y\}}[a; stop|||b; stop]$ n'est pas canonique, alors que la configuration ${}_{\{x,y\}}[a; stop]|||_{\{z\}}[b; stop]$ est canonique.

Proposition 3.1 *Toute configuration canonique est sous l'une des formes suivantes :*

${}_M[stop]$ ${}_M[exit]$ ${}_M[a; E]$ ${}_M[P]$ $\varepsilon[]\mathcal{F}$
$\varepsilon|[L]|\mathcal{F}$ $hide\ L\ in\ \varepsilon$ $\varepsilon >> F$ $\varepsilon[> \mathcal{F}$ $\varepsilon[b_1/a_1, ..., b_n/a_n]$
où ε et $\mathcal{F}$ sont des configurations canoniques.

Définition 3.7 *La fonction* $\psi : C \longrightarrow 2^{\mathcal{M}}_{fn}$, *qui détermine l'ensemble des noms évènement dans une configuration, est définie récursivement par :*

$\psi({}_M[P]) = M$ $\psi(\varepsilon[]\mathcal{F}) = \psi(\varepsilon) \cup \psi(\mathcal{F})$
$\psi(\varepsilon|[L]|\mathcal{F}) = \psi(\varepsilon) \cup \psi(\mathcal{F})$ $\psi(hide\ L\ in\ \varepsilon) = \psi(\varepsilon)$
$\psi(\varepsilon >> F) = \psi(\varepsilon)$ $\psi(\varepsilon[> \mathcal{F}) = \psi(\varepsilon) \cup \psi(\mathcal{F})$
$\psi(\varepsilon[b_1/a_1, ..., b_n/a_n]) = \psi(\varepsilon)$

Définition 3.8 *Soit σ une fonction de substitution, la substitution simultanée, $\sigma\xi$, de toutes les occurrences de x dans ξ par σx, est définie récursivement sur la configuration ξ comme suit :*

$(_M[E])\sigma \equiv_{M\sigma} [E]$ $\quad$ $(\xi[]\mathcal{F})\sigma \equiv \xi\sigma[]\mathcal{F}\sigma$
$(\xi|[L]|\mathcal{F})\sigma \equiv \xi\sigma|[L]|\mathcal{F}\sigma$ $\quad$ $(hide\ L\ in\ \xi)\sigma \equiv hide\ L\ in\ \xi\sigma$
$(\xi >> F)\sigma \equiv \xi\sigma >> F$ $\quad$ $(\xi[> \mathcal{F})\sigma \equiv \xi\sigma[> \mathcal{F}$
$\psi(\xi[b_1/a_1, ..., b_n/a_n])\sigma \equiv \xi\sigma[b_1/a_1, ..., b_n/a_n]$

Définition 3.9 *Soit ξ une configuration $\xi\backslash N$ dénote la configuration obtenue par la suppression de l'ensemble des noms d'évènements N de la configuration ξ.$\xi\backslash N$ est définie récurssivement sur la configuration ξ comme suit :*

$(_{M-N}[E])\backslash N =_{M-N} [E]$ $\quad$ $(\xi[]\mathcal{F})\backslash N = \xi\backslash N[]\mathcal{F}\backslash N$
$(\xi|[L]|\mathcal{F})\backslash N = \xi\backslash N|[L]|\mathcal{F}\backslash N$ $\quad$ $(hide\ L\ in\ \xi)\backslash N = hide\ L\ in\ \xi\backslash N$
$(\xi >> F)\backslash N = \xi\backslash N >> F$ $\quad$ $(\xi[> \mathcal{F})\backslash N = \xi\backslash N[> \mathcal{F}\backslash N$
$(\xi[b_1/a_1, ..., b_n/a_n])\backslash N = \xi\backslash N[b_1/a_1, ..., b_n/a_n]$

3.4.4 Sémantique opérationnelle de maximalité pour Basic LOTOS :

La sémantique opérationnelle de maximalité pour l'ensemble des configurations est donnée par la définition 3.10

Définition 3.10 *la relation de transition de maximalité $\longrightarrow_m \subseteq C \times Atm \times C$ définie comme étant la plus petite relation satisfaisant les régles suivantes :*

1. $\dfrac{}{{}_M[exit] \xrightarrow{M^{\delta}X}_m {}_{\{x\}}[stop]}\ x = get(\mathcal{M})$
2. $\dfrac{}{{}_M[a;E] \xrightarrow{M^{a}X}_m {}_{\{x\}}[E]}\ x = get(\mathcal{M})$
3. (a) $\dfrac{\xi \xrightarrow{M^{a}X}_m \xi'}{\mathcal{F}[]\xi \xrightarrow{M^{a}X}_m \xi'}$
   (b) $\dfrac{\xi \xrightarrow{M^{a}X}_m \xi'}{\xi[]\mathcal{F} \xrightarrow{M^{a}X}_m \xi'}$
4. $(a).i.$ $\dfrac{\xi \xrightarrow{M^{a}X}_m \xi' \quad a \notin L \cup \{\delta\}}{\xi|[L]|\mathcal{F} \xrightarrow{M^{ay}}_m \xi'[y/x]|[L]|\mathcal{F}\backslash M}\ y = get(\mathcal{M} - ((\psi(\xi) \cup \psi(\mathcal{F})) - M))$
   $ii.$ $\dfrac{\xi \xrightarrow{M^{a}X}_m \xi' \quad a \notin L \cup \{\delta\}}{\mathcal{F}|[L]|\xi \xrightarrow{M^{ay}}_m \mathcal{F}\backslash M|[L]|\xi'[y/x]}\ y = get(\mathcal{M} - ((\psi(\xi) \cup \psi(\mathcal{F})) - M))$
   $(b).$ $\dfrac{\xi \xrightarrow{M^{a}X}_m \xi' \quad \mathcal{F} \xrightarrow{N^{a}y}_m \mathcal{F}' \quad a \notin L \cup \{\delta\}}{\xi|[L]|\mathcal{F} \xrightarrow{M \cup N^{az}}_m \xi'[z/x]\backslash N|[L]|\mathcal{F}'[z\backslash y]\backslash M}$ z$= get(\mathcal{M} - ((\psi(\xi) \cup \psi(\mathcal{F})) - (M \cup N)))$
5. $(a).$ $\dfrac{\xi \xrightarrow{M^{a}X}_m \xi' \quad a \notin L}{hide\ L\ in\ \xi \xrightarrow{M^{ax}}_m hide\ L\ in\ \xi'}$
   $(b).$ $\dfrac{\xi \xrightarrow{M^{a}X}_m \xi' \quad a \in L}{hide\ L\ in\ \xi \xrightarrow{M^{i}i}_m hide\ L\ in\ \xi'}$

6. (a). $\dfrac{\xi \xrightarrow{M}{}^{a}_{m} X \xi' \qquad a \neq \delta}{\xi >> F \xrightarrow{M}{}^{a}_{m} X \xi' >> F}$

   (b). $\dfrac{\xi \xrightarrow{M}{}^{\delta}_{m} X \xi'}{\xi >> F \xrightarrow{M}{}^{i}_{m\{x\}} X [F]}$

7. $(a).$ $\dfrac{\xi \xrightarrow{M}{}^{a}_{m} X \xi' \qquad a \neq \delta}{\xi [> \mathcal{F} \xrightarrow{M}{}^{ay}_{m} \xi' [y \backslash x] [> \mathcal{F} \backslash M} \; y = get(\mathcal{M} - ((\psi(\xi) \cup \psi(\mathcal{F})) - M))$

   $(b).$ $\dfrac{\xi \xrightarrow{M}{}^{\delta}_{m} X \xi'}{\xi [> \mathcal{F} \xrightarrow{M}{}^{ay}_{m} \xi' [y \backslash x] [>_{\psi(\mathcal{F}) - M} [stop]} \; y = get(\mathcal{M} - ((\psi(\xi) \cup \psi(\mathcal{F})) - M))$

   $(c).$ $\dfrac{\mathcal{F} \xrightarrow{M}{}^{a}_{m} X \mathcal{F}'}{\xi [> \mathcal{F} \xrightarrow{M}{}^{ay}_{m \, \psi(\mathcal{F}) - M} [stop] [> \mathcal{F}' [y \backslash x]} \; y = get(\mathcal{M} - ((\psi(\xi) \cup \psi(\mathcal{F})) - M))$

8. $(a).$ $\dfrac{\xi \xrightarrow{M}{}^{a}_{m} X \xi' \qquad a \in \{a_1, ..., a_n\}}{\xi [b_1/a_1, ..., b_n/a_n] \xrightarrow{M}{}^{a}_{m} X \xi' [b_1/a_1, ..., b_n/a_n]}$

   $(b).$ $\dfrac{\xi \xrightarrow{M}{}^{a}_{m} X \xi' \qquad a = a_i \;\; (1 \leq i \leq n)}{\xi [b_1/a_1, ..., b_n/a_n] \xrightarrow{M}{}^{b_i}_{m} X \xi' [b_1/a_1, ..., b_n/a_n]}$

9. $\dfrac{P := E \qquad {}_{M}[E] \xrightarrow{M}{}^{a}_{m} X \mathcal{F}}{{}_{M}[P] \xrightarrow{M}{}^{ax}_{m} \mathcal{F}}$

3.5 Conclusion

La sémantique d'entrelacement admet comme hypothèse l'atomicité structurelle et temporelle des actions, c'est-à-dire que les actions sont à la fois indivisibles et de durées nulles. Mais dans certains cas de telles hypothèses ne sont pas acceptables, en particulier lorsqu'on considère des méthodologies de conception basées sur le raffinement d'actions. Dans ce chapitre nous avons présenté quelques sémantiques qui visent à surmonter ce problème. Parmi ces sémantiques, la sémantique de causalité sur laquelle sont basés quelques modèles d'ordre partiel, cependant des études ont montré que ces sémantiques ne sont ni nécessaires, ni suffisantes pour la levé de l'hypothèse d'atomicité des actions. Quand à la ST-sémantique, dont les propriétés sont intéressantes pour le raffinement d'actions, elle consiste à diviser les actions non instantanées en débuts et fins pour capturer le parallélisme. Cependant ceci contribue directement à l'explosion combinatoire du graphe d'états. Par ailleurs est la sémantique de maximalité peut être considérée comme étant la réconciliation des sémantiques d'ordre partiel et de la ST-sémantique. Le principe de cette sémantique consiste à utiliser les relations de dépendances entre les occurrences d'actions pour associer à chaque état du système les actions qui sont potentiellement en cours d'exécution. L'application de cette sémantique à des modèles réels a été déjà montrée plusieurs fois, à titre d'illustration nous avons présenté la sémantique opérationnelle de maximalité de Basic LOTOS.

Chapitre 4

Sémantique de maximalité pour les réseaux de Petri

4.1 Problématique :

Le modèle des réseaux de Petri est très attractif, non seulement par son aspect graphique mais aussi par sa capacité à capturer le comportement parallèle des systèmes. L'une des approches de vérification d'un réseau de Petri consiste à générer son graphe de marquages dont les sommets représentent les états dans lesquels le système peut se trouver, et dont les arcs représentent les transitions faisant passer le système d'un état à un autre. Après sa génération, le graphe de marquage peut être vu comme un système de transitions étiquetées [Arn92]. Le système de transitions étiquetées, ainsi généré, est utilisé pour la vérification des propriétés du système spécifié par le réseau de Petri (model checking, bissimilarité, test de conformité... etc. [CES86][FM][CH93]). Cependant, tel que nous l'avons souligné dans le chapitre 3, le modèle des systèmes de transitions étiquetées ne peut pas distinguer entre exécution séquentièlle et exécution parallèle des transitions. A titre illustratif considérons l'exemple des deux réseaux de Petri des figures 4.1.(a) et 4.1.(b). Le réseau de Petri de la figure 4.1.(a) représente un système pouvant exécuter les actions a et b en parallèle, alors que le réseau de Petri de la figure 4.1.(b) représente un système qui exécute les actions a et b entrelacées dans le temps. Malgré cela, ces deux réseaux de Petri ont une seule représentation sémantique donnée par la figure 4.1.(c). D'autre part ce modèle exige l'atomicité structurelle et temporelle des actions. Cependant, dans la réalité, cette hypothèse d'atomicité temporelle et structurelle est difficilement acceptable. À titre d'exemple, une instruction écrite dans un langage de haut niveau est traduite en une suite d'instructions machines interruptibles ; chaque instruction machine étant de durée non nulle.

La prise en compte de la non atomicité des actions dans un système a été étudiée dans la littérature à travers la définition de plusieurs sémantiques supportant le concept du raffinement d'actions [DLW94, CS95, DD89, DD91, DD93, AH91, BC88,

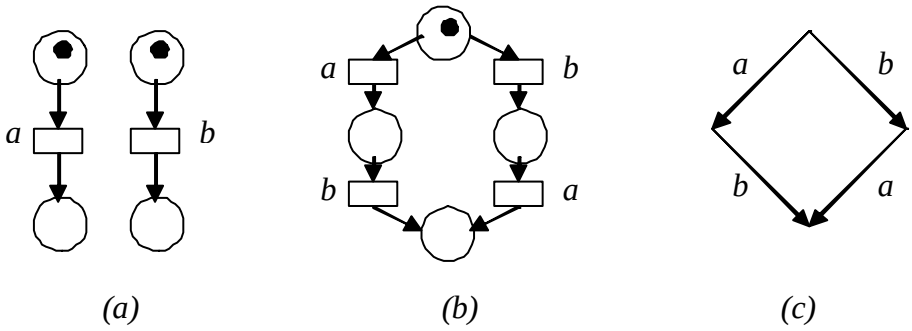


FIG. 4.1 – Modèle d'entrelacement.

BDKP91, DG91, Dev92a, Dev92b, Dev93, JPZ91, Saï96, SC94, van90b]. L'une des façons de caractériser l'exécution parallèle des actions consiste à constater que, dans un état donné, seules les occurrences d'actions maximales correspondentent aux actions peuvent s'exécuter en parallèle dans cet état. Une occurence d'action est dite *maximale* dans un état s'il n'existe pas une autre occurence d'action qui lui est causalement dépendante. Cette idée a conduit en la définition de la relation de bissimulation de préservation de la maximalité[1] sur les réseaux de Petri et sur les structures d'événements de manière indépendante [Dev92a, Vog93]. Cette relation de bissimulation est montrée équivalente à la ST-bissimulation [Dev92a, Vog93]. Par ailleurs, une sémantique opérationnelle et une sémantique dénotationnelle de maximalité ont été définies pour l'algèbre de processus Basic LOTOS, le modèle d'interprétation est appelé *modèle des arbres maximaux* [CS95, Saï96]. L'adéquation entre les deux sémantiques a été établie. La relation de bissimulation de maximalité a été définie entre les expressions Basic LOTOS ainsi qu'entre les arbres maximaux, de même pour le raffinement d'actions. La relation de bissimulation de maximalité, étendue à Basic LOTOS et aux arbres maximaux est prouvée préservée par le raffinement d'actions. Afin de vérifier les systèmes à comportements cycliques, le modèle des systèmes de transitions étiquetées maximales a été défini, ce dernier est sémantiquement équivalent au modèle des arbres maximaux [SL03]. D'autre part, l'intérêt de la sémantique de maximalité pour la prise en compte des durées d'actions a été mis en évidence dans [SC03]. En effet, une relation de performance a été définie sur un ensemble de processus exprimés dans une algèbre de processus à la LOTOS. Pour les processus liés par la relation de bissimulation de maximalité, il a été montré que ces processus sont liés par cette relation de performance et ceci quelque soit la fonction attribuant les durées aux actions. En d'autres termes, les processus liés par la relation de bissimulation de maximalité ont les mêmes performances indépendemment de la machine sur laquelle ils s'exécutent [SC03]. Dans [SL03, SB05], un algorithme de model checking et un algorithme de model checking symbolique ont été définis sur le modèle des systèmes de transitions étiquetées maximales ; les propriétés sont

[1] Cette relation sera appelée par la suite *bissimulation de maximalité.*

exprimées en logique temporelle arborescente CTL. Plus particulièrement, il a été montré que les algorithmes de model checking définis sur le modèle des systèmes de transitions étiquetées s'adaptent facilement au modèle des systèmes de transitions étiquetées maximales. Par ailleurs, des algorithmes de réduction des systèmes de transitions étiquetées maximales modulo des relations d'équivalence, entre autres la α-réduction et le graphe des pas maximaux, ont été définis dans [BS06]. Ces différents résultats sont intégrés dans l'environnement de vérification formelle FOCOVE[2] (FOrmal COncurrency Verification Environment).

Afin de tirer profit des différents résultats développés autour du modèle des systèmes de transitions étiquetées maximales, nous proposons une sémantique opérationnelle de maximalité pour les réseaux de Petri Places/Transitions. Cette sémantique permet d'interpréter tout réseau de Petri en terme des système de transitions étiquetées maximales. L'approche proposée est valable pour les réseaux de Petri à comportements cycliques, néanmoins nous considérons que les réseaux sont bornés et que toute transition à au moins une place en entrée et une place en sortie (c'est-à-dire ne contient pas des transitions sources ou des transitions puits). Ainsi dont le but de réduire la complexité des systèmes de transitions étiquetées maximales générés nouss allons proposer une nouvelle sémantique opérationnelle pour la génération des systèmes de transitions étiquetées maximales, cette dernière fait l'agrégation de dérivations redondantes et on montre que les systèmes de transitions étiquetées maximales obtenus par l'application des deux sémantiques opérationnelles sont équivalents modulo la relation de bissimulation de maximalité. L'intérêt de l'approche proposée réside dont le fait que c'est une méthode de réduction générés à la volé, ce qui permet de réduire de manière significative la taille de graphe d'états.

4.2 Système de transitions étiquetées maximales associé à un réseau de Petri

Considérons le réseau de Petri de la figure 4.2.(a). Dans cet exemple, seules les transitions t_1 et t_2, peuvent s'exécuter en parallèle, ce qui correspond à l'exécution parallèle de deux actions de nom a. À titre illustratif, nous admettons que les débuts d'exécution des actions a sont identifiés par les noms d'événements x et y alors que le début d'exécution de l'action b est identifié par le nom d'événement z.

Le système de transitions étiquetées maximales qui représente la sémantique du réseau de la figure 4.2.(a) est donné par la figure 4.2.(b). Dans l'état initial, aucune action n'a commencé son exécution, l'état initial est donc étiqueté par l'ensemble vide. À partir de cet état, chacune des actions a peut commencer son exécution, d'où les transitions identifiées respectivement par les noms d'événement x et y. L'état 1, étiqueté par l'ensemble $\{x\}$ signifie que l'action a est potentiellement en cours d'exécution au niveau de cet état. La transition identifiée par le nom d'événement y, correspond au début de l'exécution de l'autre action a. L'état 3, étiqueté par

[2] `http://www.focove.new.fr`.

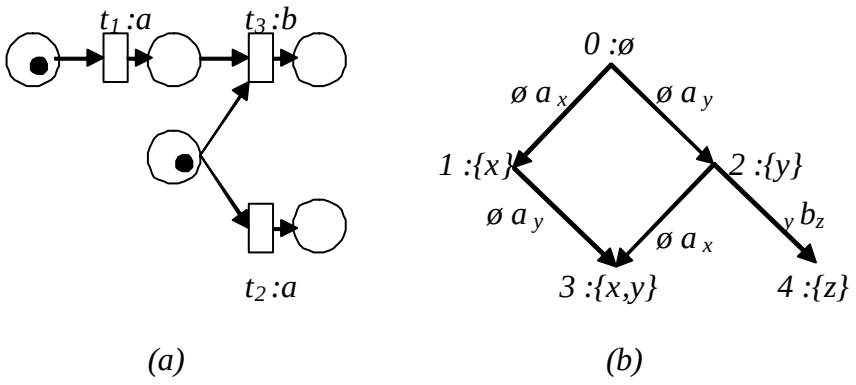


FIG. 4.2 – Système de transitions étiquetées maximales

l'ensemble $\{x, y\}$, montre que les deux actions a peuvent être en cours d'exécution *simultanément* ; alors que l'état 2, étiqueté par l'ensemble $\{y\}$, montre que dans cet état seule l'action a peut être en cours d'exécution. À partir de l'état 2, deux scénarios se présentent : soit la deuxième action a commence son exécution, dans ce cas on aboutit à l'état 3 ; soit c'est l'action b qui commence son exécution. Il est clair que l'action b ne peut commencer son exécution qu'après la fin de la première exécution de l'action a. Cette dépendance causale entre l'exécution de l'action a et l'exécution de l'action b est capturée par l'ensemble $\{y\}$ associé à la transition menant le système de l'état 2 à l'état 4. Dans l'état résultant, seule l'action b peut être en cours d'exécution, d'où l'étiquetage de cet état par l'ensemble $\{z\}$.

4.3 Réseaux de Petri et sémantique de maximalité

Dans cette section et à travers de simples exemples, nous introduisons les notations et les fonctions utiles à la définition du graphe de marquage associé à un système étiqueté dans une approche de maximalité, ce qui permettra de générer, pour tout système étiqueté, un système de transitions étiquetées maximales.

4.3.1 Graphe de marquage basé sur la sémantique de maximalité

Soit le réseau marqué de la figure 4.3.(a).

Après le tir de la transition t_1, il est évident que les tirs des transitions t_2 et t_3 sont conditionnés par la fin de l'action liée à la transition t_1. Pour capturer cette dépendance causale entre les tirs des transitions, nous considérons que les jetons produits par le tir de la transition t_1 sont liés à cette transition, entre autre le jeton de la place s_2 est le jeton de la place s_3. On peut remarquer que, dans l'état initial, le jeton dans s_1 n'est lié à aucune transition, ce jeton est dit libre dans cet état là. Dans le cas où la transition t_2 serait tirée, on pourrait déduire que l'action associée au tir de t_1 s'est terminée. De ce fait, le jeton dans s_3 deviendra libre. Le marquage

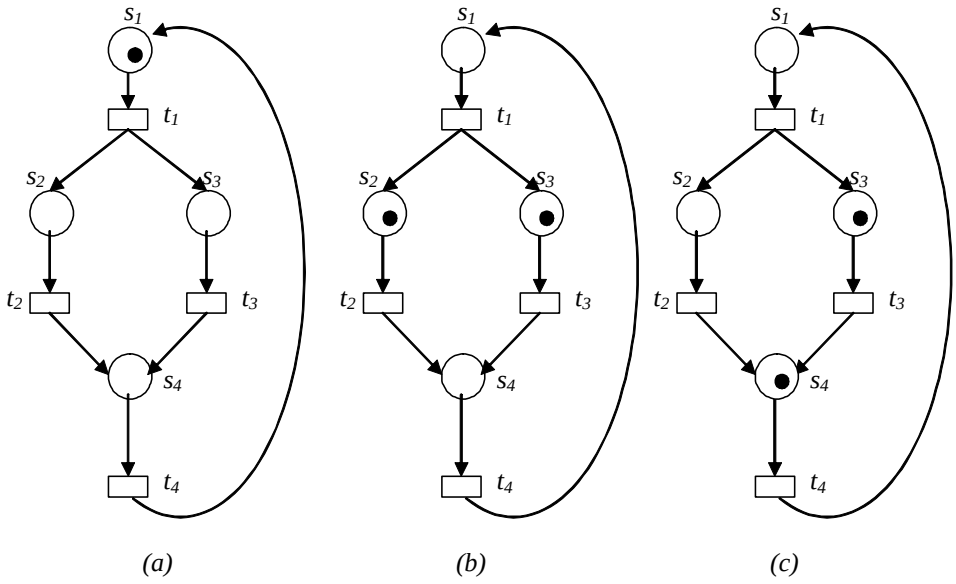


FIG. 4.3 – Réseau marqué

résultant du tir de la transition t_2 est donné par la figure 4.3.(c).

Pour distinguer entre jetons libres et jetons liés dans une place, on peut imaginer qu'une place est composée de deux parties disjointes. La partie de gauche contiendra les jetons libres tandis que celle de droite contiendra les jetons liés. Dans une place, le nombre des jetons libres sera désigné par $\mathcal{FT}$ (pour *free tokens*) alors que l'ensemble des jetons liés sera désigné par $\mathcal{BT}$ (pour *bound tokens*). On obtiendra ainsi la succession des marquages de la figure 4.4.

A partir de la configuration C_2 on aurait pu tirer la transition t_3. La configuration résultante serait C_4. (Voir figure 4.5).

Le jeton dans s_4 est maintenant lié à la transition t_3 et le jeton dans s_2 est désormais libre puisque la fin de l'action associée à la transition t_1 implique la libération du jeton de s_2.

Une question qui se pose est comment lier un jeton à une transition ? Pour y répondre, on considère le réseau marqué de la figure 4.6.(a).

Par un tir de la transition t_1, on obtiendra le réseau marqué de la figure 4.6.(b). A partir de ce marquage, on remarque que la transition t_1 est franchissable. Le franchissement de cette transition conduira à la configuration de la figure 4.6.(c).

Les deux jetons de la place s_2 sont liés. En effet, l'un est lié au premier tir de la transition t_1 alors que l'autre est lié au second tir de la même transition (deux actions associées à t_1 pouvant être en exécution parallèle). Pour lever cette ambiguïté, chaque tir de transition sera identifié par un nom d'événement. De ce fait, le lien d'un jeton sera caractérisé aussi bien par la transition qui la produite que par le nom d'événement identifiant le franchissement de cette transition. Les successions des franchissements de l'exemple précédent sont illustrées par la figure4.7.

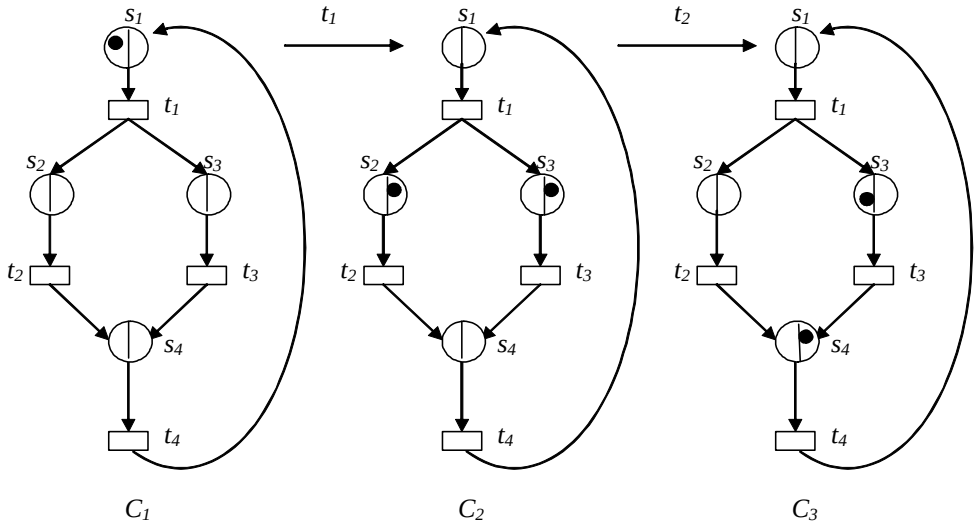


FIG. 4.4 – Jetons libres et jetons liés dans un marquage

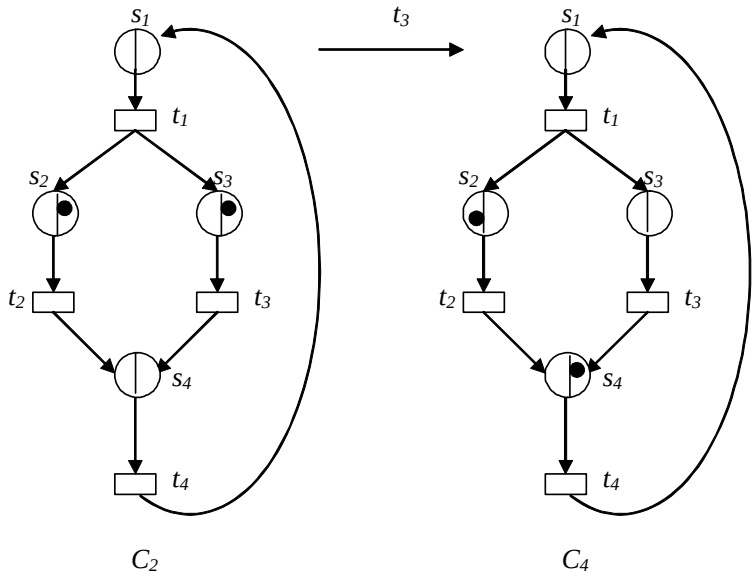


FIG. 4.5 – Franchissement de la transition t_3

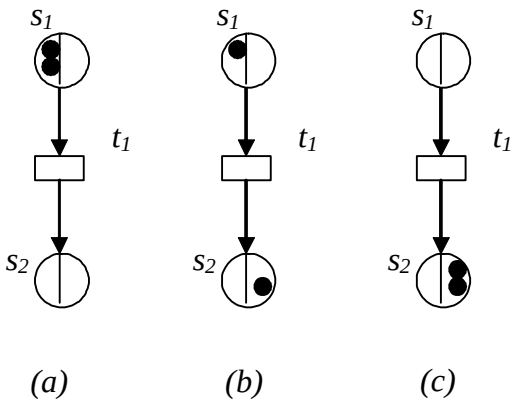


FIG. 4.6 – Liaison des jetons.

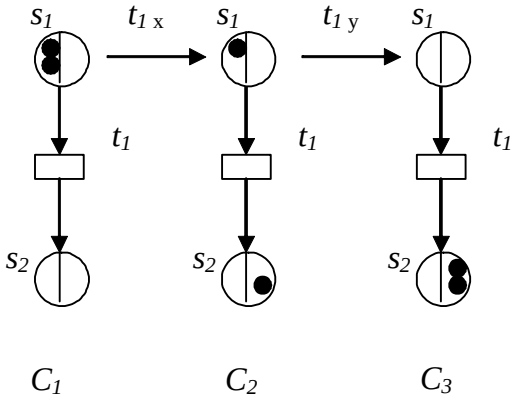


FIG. 4.7 – Successions de franchissements de t_1

Dans la configuration C_2, l'ensemble des jetons liés dans s_2 est $\mathcal{BT} = \{t_{1_x}\}$ alors que l'ensemble des jetons liés dans s_2 de la configuration C_3 est $\mathcal{BT} = \{t_{1_x}, t_{1_y}\}$. Le nom de l'événement x désigne le premier franchissement de la transition t_1 alors que y désigne le deuxième franchissement de cette même transition.

Un autre problème se pose quant aux jetons liés à un même tir de transition. Pour le voir, considérons le réseau de la figure 4.8.(a).

Par le franchissement de la transition t_1 on obtient la dérivation de la figure 4.8.(b).

La partie droite $\mathcal{BT}$ de la place s_2 contient deux jetons liés au tir t_{1_x}, c'est-à-dire $\mathcal{BT} = \{t_{1_x}, t_{1_x}\}$. Etant donné que $\mathcal{BT}$ est un ensemble, nous considérons qu'un jeton lié est un triplet (n, t, x) de $\mathbb{N}\times T \times \mathcal{M}$, (noté aussi nt_x), où n est le nombre d'instances, t est la transition productrice de ce jeton et x est le nom d'événement associé au franchissement de la transition t. On note $\mathcal{BT} = \{n_1 t_{1x_1}, n_2 t_{2x_2}, \ldots\}$

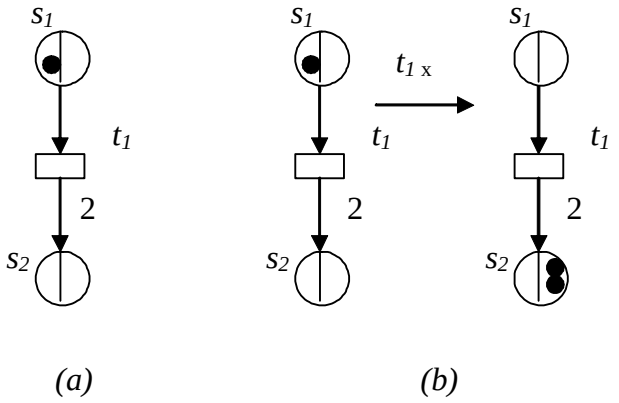


FIG. 4.8 – Réseau avec arc sortant de poids supérieur à un

l'ensemble (éventuellement vide) des jetons liés. Dans l'exemple précédent $\mathcal{BT} = \{2t_{1_x}\}$.

Par conséquent, le *marquage* d'une place s est un couple $(\mathcal{FT}, \mathcal{BT})$ où $\mathcal{FT}$ est le nombre de *jetons libres* dans s.

Définition 4.1 *Etant donné un réseau $N = (S, T, W)$, le marquage de N est désormais une fonction $M : S \rightarrow \mathbb{N} \times 2^{\mathbb{N} \times T \times \mathcal{M}}$. Entre autre le marquage $M(s)$ d'une place $s \in S$ est un couple $(\mathcal{FT}, \mathcal{BT})$ tel que $\mathcal{FT} \in \mathbb{N}$ et $\mathcal{BT} \in 2^{\mathbb{N} \times T \times \mathcal{M}}$ [3] désignent respectivement le nombre de jetons libres et l'ensemble (éventuellement vide) de jetons liés dans la place s. Dans ce qui suit un réseau muni d'un marquage sera appelé configuration. $|M(s)|$ désigne le nombre total de jetons dans la place s. Si $M(s) = (\mathcal{FT}, \mathcal{BT})$ tel que $\mathcal{BT} = \{n_1 t1_{x_1}, \ldots, n_m tm_{xm}\}$ alors $|M(s)| = \mathcal{FT} + |\mathcal{BT}|$ avec $|\mathcal{BT}| = \sum_{i=1}^{m} n_i$ est le cardinal de l'ensemble des jetons liés dans la place s. Parfois on utilisera l'écriture $\mathcal{FT}(s)$ et $\mathcal{BT}(s)$ pour désigner les composantes du marquage $M(s)$ de la place s.*

Après avoir montré les jetons produits par le franchissement d'une transition, un point reste à éclairer et qui concerne l'identification des jetons consommés par le franchissement de celle ci. Pour cela, considérons le réseau de la figure 4.9.

On admet qu'un jeton de s_3 est lié au franchissement de t_1 (t_{1_x}) et que l'autre jeton est lié au franchissement de t_2 (t_{2_y}). Par un premier franchissement de la transition t_3, on obtient la configuration C'_2, et par un second franchissement de cette même transition on obtient la configuration C'_3.

Parmi les jetons liés dans s_3, on veut connaitre le jeton consommé dans le premier franchissement de t_3 et celui consommé dans le second franchissement de cette même transition. Cette information est indispensable pour connaitre, dans chaque configuration, les actions (associées aux transitions) qui ont terminé leur exécution.

[3] 2^S est l'ensemble des parties de S

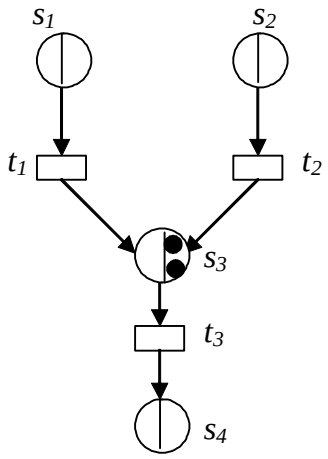


FIG. 4.9 – Identification des jetons consommés

Pour ce faire, nous associons au niveau d'un franchissement les noms d'événements identifiant les jetons liés consommés par ce franchissement. Ceci nous donne la succession des franchissements de la figure 4.10.

Dans l'exemple de la figure 4.4, nous avons noté qu'après le franchissement de la transition t_2, le jeton qui était lié à la transition t_1 dans la place s_3 devient libre dans le marquage résultant. Ceci est dû au fait que le franchissement de la transition t_3 est conditionné par la fin de l'action relative à la transition t_1. Dans le paragraphe suivant, nous donnons quelques définitions préliminaires qui nous permettront la proposition de la méthode de génération d'un graphe de marquage dans le contexte de la sémantique de maximalité.

Définitions préliminaires

Définition 4.2 *Etant donné un réseau (S, T, W) muni d'un marquage M :*

- *L'ensemble des noms des événements maximaux dans M est l'ensemble de tous les noms d'événements identifiant les jetons liés au niveau des places du réseau dans le marquage M. Formellement la fonction ψ sera utilisée pour le calcul de cet ensemble, elle peut être définie de la manière suivante : $\psi(M) = \bigcup_{s \in S} \bigcup_{i=1}^{ms} xs_i$ tel que $M(s) = (\mathcal{FT}, \mathcal{BT})$ avec $\mathcal{BT} = \{(ns_1, ts_1, xs_1), \ldots, (ns_{ms}, ts_{ms}, xs_{ms})\}$.*
- *$\mathcal{N} \subset \mathcal{M}$ étant une ensemble fini non vide de noms d'évenements. makefree$(\mathcal{N}, M)$ est définie récurssivement par :*
  - *makefree $(\{x_1, x_2, \ldots, x_n\}, M)$ = makefree $(\{x_2, \ldots, x_n\}$, makefree $(\{x_1\}, M))$*
  - *makefree $(\{x\}, M) = M'$ tel que pour tout $s \in S$, si $M(s) = (\mathcal{FT}, \mathcal{BT})$ alors :*

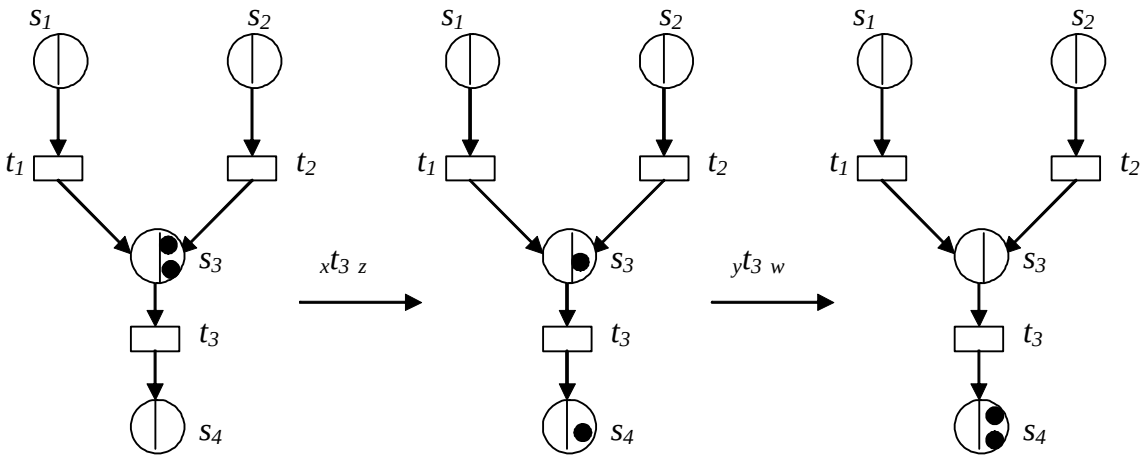


FIG. 4.10 – Noms d'évènements identifiant les jetons consommés

S'il existe $(n,t,x) \in \mathcal{BT}$ *alors* $M'(s) = (\mathcal{FT} + n, \mathcal{BT} - \{(n,t,x)\})$ *(Conversion des n jetons liés identifiés par le nom d'événement x en jetons libres).*

Sinon $M'(s) = M(s)$.

- *t étant une transition de T ; t est dite sensibilisée par le marquage M si et seulement si* $|M(s)| \geq W(s,t)$ *pour tout* $s \in S$. *L'ensemble de toutes les transitions sensibilisées par le marquage M sera désigné par enabled*(M).

- *Le marquage M est dit minimal pour le franchissement de la transition t si et seulement si* $|M(s)| = W(s,t)$ *pour tout* $s \in S$.

- *Soient* M_1 *et* M_2 *deux marquages du réseau* (S,T,W). $M_1 \Subset M_2$ *si et seulement si* $\forall s \in S$, *si* $M_1(s) = (\mathcal{FT}_1, \mathcal{BT}_1)$ *et* $M_2(s) = (\mathcal{FT}_2, \mathcal{BT}_2)$ *alors* $\mathcal{FT}_1 \leq \mathcal{FT}_2$ *et* $\mathcal{BT}_1 \Subset \mathcal{BT}_2$ *tel que la relation* $\Subset$ *et étendue aux ensembles des jetons liés de la manière suivante :*

  - $\mathcal{BT}_1 \Subset \mathcal{BT}_2$ *si et seulement si* $\forall (n_1,t,x) \in \mathcal{BT}_1$, $\exists (n_2,t,x) \in \mathcal{BT}_2$ *tel que* $n_1 \leq n_2$.

- *Soient* M_1 *et* M_2 *deux marquages du réseau* (S,T,W) *tel que* $M_1 \Subset M_2$. *La différence* $M_2 - M_1$ *est un marquage* M_3 $(M_2 - M_1 = M_3)$ *tel que pour tout* $s \in S$, *si* $M_1(s) = (\mathcal{FT}_1, \mathcal{BT}_1)$ *et* $M_2(s) = (\mathcal{FT}_2, \mathcal{BT}_2)$ *alors* $M_3(s) = (\mathcal{FT}_3, \mathcal{BT}_3)$ *avec :*

  - $\mathcal{FT}_3 = \mathcal{FT}_2 - \mathcal{FT}_1$
  - $\forall (n_1,t,x) \in \mathcal{BT}_1$, $(n_2,t,x) \in \mathcal{BT}_2$ *si* $n_1 \neq n_2$ *alors* $(n_2 - n_1, t, x) \in \mathcal{BT}_3$.

- $Min(M,t) = \{M' | M' \Subset M\}$ *et* M' *est minimal pour le franchissement de la transition* t*. En d'autre terme* $Min(M,t)$ *est l'ensemble de tous les marquages minimaux inclus dans* M *et vérifiant la propriété de sensibilisation de la transition* t*.*

Construction du graphe de marquage

Soit la donnée d'un système étiqueté $\Sigma = (S, T, W, M_0, \lambda)$. Le graphe de marquage GM étiqueté par λ associé à Σ est celui dans les états sont définis par tous les marquages accéssibles à partir du marquage initial M_0. La règle de dérivation des marquages est définie récurssivement comme il est donné par la définition 4.3 :

Définition 4.3 *M étant un marquage accéssible du réseau marqué (S, T, W, M_0), $t \in enabled(M)$ alors pour tout $M" \in Min(M,t)$,$E = \psi(M")$ et $M''' = makefree(E, M - M")$; la dérivation suivante est possible $M \xrightarrow{E t_x} M'$ (notée aussi $(M, {}_E\, t_x, M')$) tel que :*

- E étant l'ensemble des noms d'événements maximaux associés aux actions dont les fins d'exécution sont impératives au début de l'action associée au franchissement de la transition t.
- $x = get(\mathcal{M} - \psi(M'''))$. On choisit un nom d'événement de l'ensemble $\mathcal{M}$ qui n'est pas associé à une action pouvant rester en cours d'éxécution après le franchissement de la transition t. Notons que les noms d'événements appartrenant à E peuvent être réutilisés. Cette méthode de choix des noms d'événement permet la considération des systèmes à comportements cycliques.
- $M\prime = occur(t, x, M''')$ est le marquage résultant du rajout des jetons liés à la transition t au marquage M'''. De manière générale, étant donnés un marquage M, une transition t et un nom d'événement $x \notin \psi(M)$; la fonction $occur(t, x, M) = M\prime$ tel que $\forall s \in S$ si $M(s) = (\mathcal{FT}, \mathcal{BT})$ alors $M\prime(s) = (\mathcal{FT}, \mathcal{BT}')$ avec $\mathcal{BT}' = \mathcal{BT} \cup \{W(t,s), t, x)\}$ si $W(t,s) \neq 0$ et $\mathcal{BT}' = \mathcal{BT}$ sinon.

Propriétés

Proposition 4.1 *Etant donné un système étiqueté $\Sigma = (S, T, W, M_0, \lambda)$ dont le graphe de marquage GR est construit selon la définition 4.3, alors la structure $\Sigma_{stem} = (GR, \lambda, \mu, \xi, \psi)$ est un système de transitions étiquetées maximales avec :*

- $GR = \langle Sg, Tg, \alpha, \beta, M_0 \rangle$ *est le graphe de marquages associé à Σ tel que*
  - *Sg est l'ensemble des états défini par l'ensemble des marquages accessibles à partir du marquage initial M_0.*
  - *$Tg = \{(M, {}_E\, t_x, M') | M, M' \in Sg \wedge (M, {}_E\, t_x, M')$ est une dérivation valide dans le graphe des marquages$\}$*

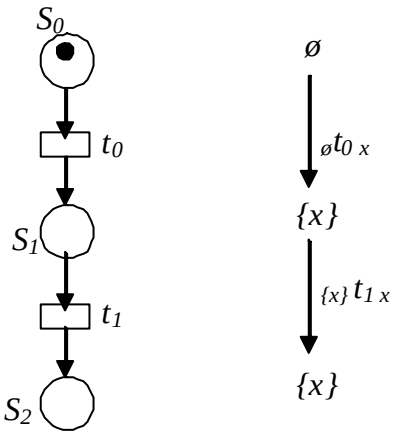


FIG. 4.11 – Séquence

– *Pour* $(M,_E t_x, M') \in Tg$, $\alpha((M,_E t_x, M')) = M$ *et* $\beta((M,_E t_x, M')) = M'$

- $\psi : Sg \rightarrow 2^{\mathcal{M}}$ *est la fonction définie précédement.*
- *Pour* $d = (M,_E t_x, M') \in Tg$ *on pose* $\lambda(d) = \lambda(t)$, $\mu(d) = E$ *et* $\xi(d) = x$.

Preuve. Remarquons tout d'abord que le marquage initial M_0 ne contient que des jetons libres, de ce fait $\psi(M_0) = \varnothing$. D'autre part pour $d = (M,_E t_x, M') \in Tg$, D'après la définition 4.3, $\mu(d) = E \subseteq \psi(\alpha(d)) = \psi(M)$, $\xi(t) \notin \psi(M) - \mu(d)$ et $\psi(\beta(t)) = \psi(M') = (\psi(M) - \mu(d)) \cup \{\xi(d)\}$. D'où le résultat. □

Proposition 4.2 *Etant donné un réseau marqué* (S, T, W, M_0)*, alors l'ensemble des séquences de transition généré dans une approche d'entrelacement est le même que celui généré dans l'approche de maximalité.*

Preuve. Dérive directement du fait que dans l'approche de maximalité, la condition de franchissement d'une transition ne prend en compte que le nombre de jetons dans les places et non la nature de ces jetons. Ce qui revient aux mêmes conditions de franchissement des transitions dans l'approche d'entrelacement. □

4.3.2 Exemples divers

Cette section montre quelques exemples de réseaux de Petri marqués accompagnés de leurs systèmes de transitions étiquetées maximales.

Séquence (dépendance)

La figure 4.11 montre que la transition t_1 doit attendre l'achèvement de l'exécution de l'action associée à t_0 pour qu'elle soit franchie.

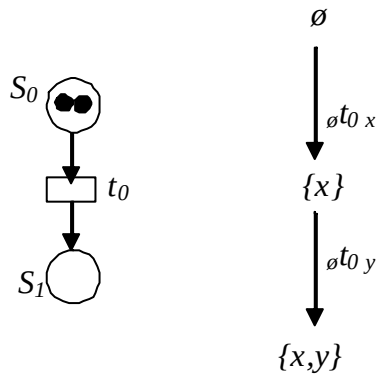


FIG. 4.12 – Auto-concurrence

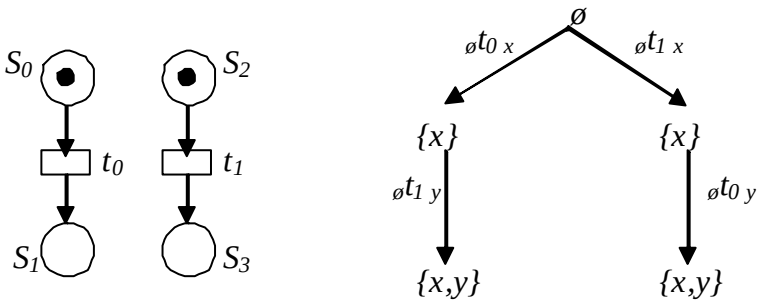


FIG. 4.13 – Parallélisme

Auto-concurrence

La figure 4.12 montre deux actions du même nom en train d'être exécutées en concurrence.

Parallélisme

Les deux actions liées à t_0 et t_1 dans la figure 4.13 peuvent être franchies en parallèle.

Conflits

La figure 4.14 montre une situation de conflit sur un jeton.

Comportements cycliques

La figure 4.15 montre un système de transitions étiquetées maximales correspondant à un réseau à comportement cyclique.

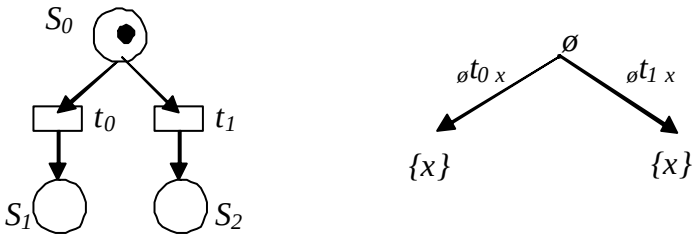


FIG. 4.14 – Conflit sur les jetons

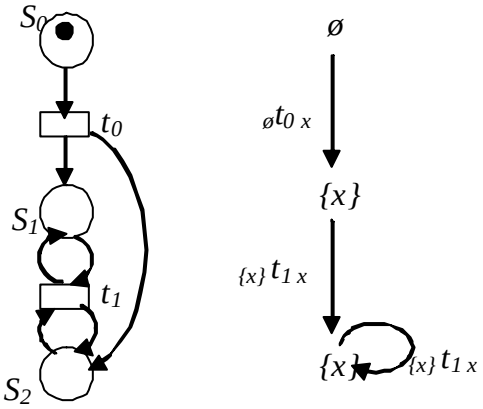


FIG. 4.15 – Comportement cyclique

4.4 Relation de bissimulation de maximalité

Dans cette section nous rappelons la relation de bissimulation de maximalité définie sur le modèle des systèmes de transitions étiquetées maximales et nous étalons cette relation aux systèmes étiquetés.

Notation 4.1 *Dans ce qui suit, nous admettons les notations suivantes :*

- *Soit $f : E \to F$ une fonction de domaine $Dom(f) = E$ et de co-domaine $Cod(f) = F$, et soit D (respectivement C) un sous ensemble de E (respectivement de F). Les restrictions de f vis à vis de son domaine et co-domaine sont définies par :*
  - $f\lceil D = \{(x, y) \in f | x \in D\}$
  - $f\lfloor C = \{(x, y) \in f | y \in C\}$
- $\mathfrak{F} \subseteq 2^{\mathcal{M}\times\mathcal{M}}$ *dénote toutes les fonctions bijectives entre les sous ensembles de* $\mathcal{M}$.

Définition 4.4 *Soient $Stem1 = (\Omega_1, \lambda_1, \mu_1, \xi_1, \psi_1)$ et $Stem2 = (\Omega_2, \lambda_2, \mu_2, \xi_2, \psi_2)$ deux systèmes de transitions étiquetées maximales tels que $\Omega_1 = \langle S_1, T_1, \alpha_1, \beta_1, s1_0\rangle$ et $\Omega_2 = \langle S_2, T_2, \alpha_2, \beta_2, s2_0\rangle$. $Stem1$ et $Stem2$ sont dits maximalement bissimilaires s'il existe une relation $\mathfrak{R} \subseteq S_1 \times S_2 \times \mathfrak{F}$ tel que :*

1. $(s1_0, s2_0, \emptyset) \in \mathfrak{R}$. *Les états initiaux de $stem1$ et $stem2$ sont liés par la relation. Etant donné que les ensembles des événements maximaux dans ces états initiaux sont vides, alors la fonction reliant ces deux ensembles est aussi égale à l'ensemble vide.*
2. *Si $(s1, s2, f) \in \mathfrak{R}$ alors*
   (a) *$Dom(f) \subseteq \psi(s1)$ et $Cod(f) \subseteq \psi(s2)$.*
   (b) *Si $s1 \xrightarrow{E a_x} s1'$ alors il existe $s2 \xrightarrow{F a_y} s2'$ tel que :*
      i. $\forall(u, v) \in f$, *si $u \notin E$ alors $v \notin F$*
      ii. $(s1', s2', f') \in \mathfrak{R}$ *avec* $f' = (f\lceil(\psi(s1') - \{x\}))\lfloor(\psi(s2') - \{y\}) \cup \{(x, y)\}$
   (c) *Si $s2 \xrightarrow{F a_y} s2'$ alors il existe $s1 \xrightarrow{E a_x} s1'$ tel que :*
      i. $\forall(u, v) \in f$, *si $v \notin F$ alors $u \notin E$*
      ii. $(s1', s2', f') \in \mathfrak{R}$ *avec* $f' = (f\lceil(\psi(s1') - \{x\}))\lfloor(\psi(s2') - \{y\}) \cup \{(x, y)\}$

Notation 4.2 *La fonction f met en correspondance les événements maximaux correspondant aux actions qui peuvent être en cours d'exécution dans les états correspondant (on n'a pas pu déduire par bissimulation l'information sur leur fin d'exécution).*

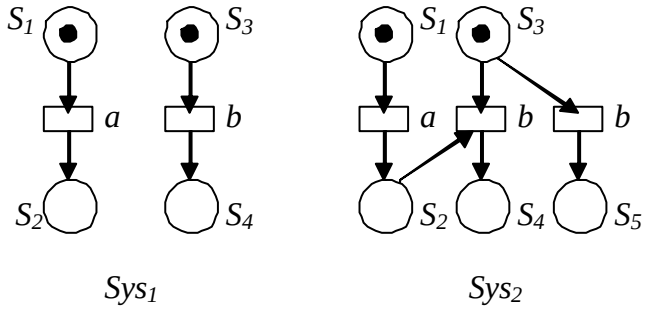


FIG. 4.16 – Systèmes maximalement bissimilaires

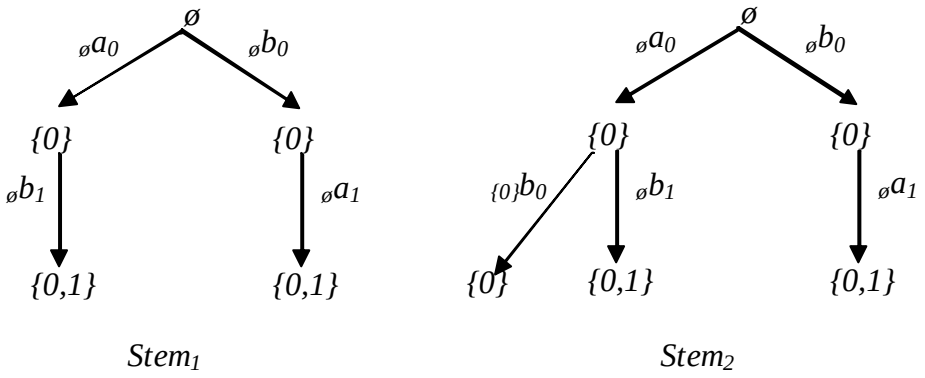


FIG. 4.17 – STEMs maximalement bissimilaires

Définition 4.5 *Soient* $\Sigma_1 = (S_1, T_1, W_1, M1_0, \lambda_1)$ *et* $\Sigma_2 = (S_2, T_2, W_2, M2_0, \lambda_2)$ *sont dits maximalement bissimilaires si et seulement si leur systèmes de transitions étiquetées maximales respectifs* Σ_{1stem} *et* Σ_{2stem} *le sont.*

Example 1 *Soit l'exemple des deux systèmes étiquetés Sys*1 *et Sys*2 *de la figure 4.16 (cet exemple est issu de [Dev92a]). En applicant l'approche proposée, les systèmes de transitions étiquetées maximales qui leur correspondent sont donnés respectivement par les figures 4.17.Stem1 et 4.17.Stem2. Le lecteur pourra facilement vérifier que ces deux systèmes sont maximalement bissimilaires.*

La relation de bissimulation définie précédement considère deux hypothèses. Dans la première hypothèse, aucune distinction n'est faite entre actions visibles et actions internes (toutes les actions sont traitées de la même manière). La deuxième hypothèse, considère que toutes les actions sont non atomiques. Il est à noter que dans [Dev92a], une relation de bissimulation de maximalité notée $AiMP$ a été définie sur les réseaux et considère la maximalité partièle sur les actions appartenant à l'ensemble A (entre autres, seules les actions de A sont non atomiques). Il est évident que cette relation peut être facilement étendue à notre contexte. De même,

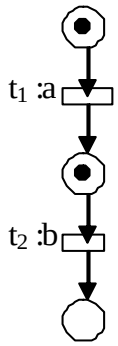


FIG. 4.18 – Système étiquetée

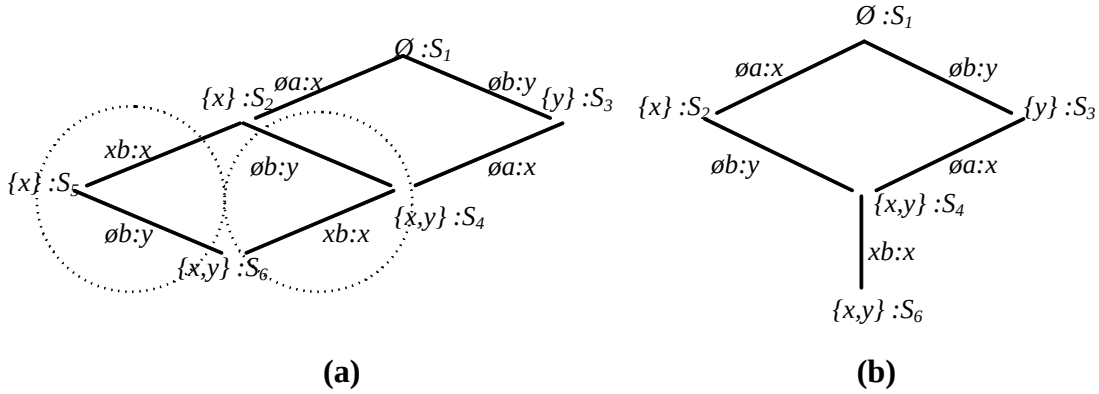


FIG. 4.19 – Systèmes de transition étiquetées maximales

dans [CS94][Saï96], une relation de bissimulation de maximalité faible a été définie sur l'algèbre de processus Basic-LOTOS et sur les arbres maximaux. Cette dernière fait abstraction des actions internes tout en restant préservée par le raffinement d'actions. L'adaptation de cette relation au contexte des systèmes de transitions étiquetées maximales est triviale.

4.5 Méthode de génération de graphe de marquage avec agrégation de transitions :

Dans cette section nous introduirons une nouvelle méthode de génération de graphes de marquages basée toujours sur la sémantique de maximalité, mais qui vise à la réduction des STEMs générés. La réduction consiste en l'agrégation des transitions redondantes modulo la relation de bissimulation de maximalité. Pour mieux clarifier les idées considérons l'exemple du système étiqueté de la figure 4.18. En appliquant l'approche proposée dans la section 4.3 alors nous obtenons le système de transitions étiquetés maximales donné par la figure 4.19.

Le système de transitions étiquetées maximales de la figure 4.19.(a) peut être expliqué de la manière suivante : Dans l'état initial (état $S1$) aucune action n'est en cours d'exécution, d'où l'association de l'ensemble vide à cet état. A partir de l'état $S1$, les actions a et b peuvent commencer leur exécution de manière indépendante, les débuts de leur exécution sont identifiés respectivement par les noms d'événements x et y. Les actions a et b peuvent commencer leur exécution dans un ordre quelconque (début de a suivi du début de b ou inversement). L'ensemble $\{x\}$ (resp. $\{y\}$) associé à l'état $S2$ (resp. $S3$) stipule que l'action a (resp. b) est potentiellement en cours d'exécution dans cet état. L'ensemble $\{x, y\}$ dans l'état $S4$ montre que les actions a et b peuvent être en exécution parallèle.

Remarquons que lorsque le système est dans l'état $S2$, tant que l'action a ne s'est pas terminée, la seule évolution concerne le début d'exécution de l'action b. Cependant, dès que l'action a temine son exécution, on peut commencer l'action b causée par a ou bien l'action b qui est indépendante de la fin de l'action a. Les états résultants sont respectivement $S4$ et $S5$. On observe qu'à partir de l'état $S5$ le début de l'action b est toujours possible. Cependant, la même contrainte de terminaison de l'action a est imposée pour l'exécution de l'action b au niveau de l'état $S4$. Notons que la dépendance causale entre l'exécution de l'action b par rapport à l'action a est capturée par la consommation du jeton produit par le franchissemnt de la transition $t1$ lors du franchissement de la transition $t2$ du réseau de Petri.

Le lecteur pourra remarquer qu'à partir de l'état $S2$, les transitions menant respectivement aux états $S4$ et $S5$ sont dûes au franchissement de la même transition $t2$. Dans le premier franchissement c'est le jeton du marquage initial qui est consommé, alors que dans le deuxième franchissement c'est le jeton produit par le franchissement de la transition $t1$ qui est consommé. D'autre part, tel que nous l'avons souligné précédement, la dérivation par b menant à l'état $S4$ n'est pas conditionnée par la fin de l'action a, alors que la dérivation menant à l'état $S5$ est conditionnée par la fin de l'action a. De ce fait, la question suivante se pose : Peut-on, dans le système de transitions étiquetées maximales, omettre les dérivations $S2 \rightarrow S5 \rightarrow S6$? En d'autre terme, le système de transitions étiquetées maximales de la figure 4.19.(b) préserve t'il la sémantique comportementale du réseau de Petri de la figure 4.18.

Dans ce qui suit nous montrerons que ces dérivations sont redondantes. Afin d'éliminer ces redondances, nous étalons la relation de bissimulation de maximalité aux transition.

Il est à noter que ce qu'on a vu dans l'exemple précédent est une réduction d'un STEM déjà généré. En effet, notre objectif est la génération à la volé d'un STEM réduit modulo la relation de bissimulation de maximalité étalée à un ensemble des transitions.

4.5.1 Sémantique opérationnelle de maximalité pour les réseaux de Petri avec agrégation de transitions :

L'approche de génération basée sur l'agrégation des transitions ressemble à celle vue précédement. De ce fait, les définitions et notations developpées dans la section 4.3 restent valable pour cette approche. Cependant l'agrégation des transitions redondantes se fera grâce à la nouvelle sémantique de la fonction Min. Etant donnés un marquage M et une transition $t, Min(M,t)$ désigne tous les marquages $M^{'}$ inclus dans M minimaux pour le franchissement de t tel qu'il n'existe pas un autre marquage minimal pour le franchissement de t et dont le nombre de jetons libres est supérieur à celui de $M^{'}$ pour une place donnée.

Formellement, $Min(M,t) = \{M^{'} \backslash M^{'} \Subset M\}$ tel que $M^{'}$ est minimale pour le franchissement de la transition t et $\forall s \in S$ avec $M^{'}(s) = (\mathcal{FT}, \mathcal{BT})$

$$M^{'}(s) = \begin{array}{l} (w(s,t), \varnothing) \;\; si \;\; \mathcal{FT} \geqslant w(s,t) \\ \left(\mathcal{FT}, \mathcal{BT}^{'}\right) \;\; sinon \;\; avec \;\; \mathcal{BT}^{'} \Subset \mathcal{BT} \;\; et \;\; \mathcal{FT} + \left|\mathcal{BT}^{'}\right| = w(s,t) \end{array}$$

4.5.2 Relation de bissimulation de maximalité sur les transitions :

Définition 4.6 *Soit $Marq$ l'ensemble des marquages, T l'ensemble des transitions et $\rightarrow$ la relation de dérivation entre les marquages définie dans la définition 4.3.*

1. Soit $\Re \subseteq 2^{Marq} \times 2^{Marq} \times \mathcal{F}$. Une relation $\Re$ est dite une relation de bissimulation de maximalité sur un ensemble de transitions d'un réseau $\mathcal{N} = (S,T,W)$ si et seulement si : $\forall \left(M_i, M_i^{'}, Id_{A_i}\right) \in \Re$, $A_i \subseteq \psi(M_i)$ et $A_i \subseteq \psi\left(M_i^{'}\right)$.

   (a) Si $M_i \overset{E_i t_i x}{\longrightarrow} M_j$ alors $\exists \;\; M_i^{'} \overset{E_i^{'} t_i y}{\longrightarrow} M_j^{'}$ / $\forall u \in A_i$ si $u \notin E_i$ alors $u \notin E_i^{'}$ et pour $z = get\left(\mathcal{M} - \left((\psi(M_i) - E_i) \cup \left(\psi\left(M_i^{'}\right) - E_i^{'}\right)\right)\right)$: $\left(M_j[z/x], M_j^{'}[z/y], Id_{A_{i+1}}\right) \in \Re /\; A_{i+1} = (A_i - E_i) \cup \{z\}$.

   (b) Si $M_i^{'} \overset{E_i^{'} t_i y}{\longrightarrow} M_j^{'}$ alors $\exists\; M_i \overset{E_i t_i x}{\longrightarrow} M_j$ /$\forall u \in A_i$ si $u \notin E_i^{'}$ alors $u \notin E_i$ et pour $z = get\left(\mathcal{M} - \left((\psi(M_i) - E_i) \cup \left(\psi\left(M_i^{'}\right) - E_i^{'}\right)\right)\right)$: $(M_j[z/x], M_j^{'}[z/y], Id_{A_{i+1}}) \in \Re /\; A_{i+1} = \left(A_i - E_i^{'}\right) \cup \{z\}$.

2. Soit $\Sigma_1 = \left(S_1, T, W_1, M_0^1, \lambda_1\right)$, $\Sigma_2 = \left(S_2, T, W_2, M_0^2, \lambda_2\right)$ deux systèmes marqués. Σ_1, Σ_2 sont dits maximalement bissimilaire par rapport à T noté $\Sigma_1 \approx_m^T \Sigma_2$ si et seulement si : $\exists\; \Re$ une relation de bissimulation de maximalité par rapport à T / $\left(M_0^1, M_0^2, \Re\right)$.

3. $M_1 \approx_m^T / f \;\; M_2$ dénote le fait que $(M_1, M_2, f) \in \Re$.

Corollaire 4.1 $\approx_m^T \subseteq \;\approx_m^{ACT}$

Preuve. Evident, car $\approx_m^T$est plus discriminante que $\approx_m^{ACT}$, en fait $\exists\; t_i \neq t_j / \lambda(t_i) = \lambda(t_j)$. Le cas échéant si λ est bijective alors $\approx_m^T = \;\approx_m^{ACT}$. □

Proposition 4.3 *Soit M un marquage avec $A \subseteq \psi(M)$ alors :*

1. $M \approx_m^T /Id_{(\psi(M))}M$
2. $M \approx_m^T /Id_{(A)}M$
3. Si $B = \psi(M) - A$ alors : $makefree(B, M) \approx_m^T /Id_A \ M$

Preuve. 1. Soit $\Re_{Id/M} = \left(M, M, Id_{\psi(M)}\right)$ on montre que $\forall M$, $\Re_{Id/M}$ est une relation de bissimulation de maximalité sur l'ensemble des transitions.

Soit $\Re_{Id/M} = \underset{i\geq 0}{\cup} \Re_i$ tel que :

$\Re_0 = \left(M, M, Id_{\psi(M)}\right)$

$\Re_1 = \left(M_1, M_1, Id_{\psi(M_1)}\right)$ tel que $\exists M \xrightarrow{Etx} M_1$

jusqu'à $\Re_{n+1} = \left(M_{n+1}, M_{n+1}, Id_{\psi(M_{n+1})}\right)$ tel que $\exists \left(M_n, M_n, Id_{\psi(M_n)}\right)$,

$\exists M_n \overset{E_{n+1}tx_{n+1}}{\longrightarrow} M_{n+1}$

$\forall \left(M_i, M_i, Id_{\psi(M_i)}\right) \in \Re_{Id/M}, \psi(M_i) \subseteq \psi(M_i)$

Si $M_i \overset{E_{i+1}tx_{i+1}}{\longrightarrow} M_{i+1}$ alors $\exists E_{i+1}^{'} = E_{i+1}$ et $\exists x_{i+1}^{'} = x_{i+1}$ tel que $M_i \overset{E_{i+1}^{'}tx_{i+1}^{'}}{\longrightarrow} M_{i+1}$. D'autre part $\forall u \in \psi(M_i)$, Si $u \notin E_{i+1}$ alors $u \notin E_{i+1}^{'}$ parceque $E_{i+1}^{'} = E_{i+1}$ alors $\left(M_{i+1}, M_{i+1}, Id_{A_{i+1}}\right) \in \Re_{Id/M}$ tel que $A_{i+1} = (\psi(M_i) - E_i) \cup \{x_{i+1}\} = \psi(M_{i+1})$ par conséquent $\left(M_{i+1}, M_{i+1}, Id_{\psi(M_{i+1})}\right) \in \Re_{Id/M}$. On déduit que $\Re_{Id/M}$ est une relation de bissimulation de maximalité sur l'ensemble des transitions.

2. Soit $\Re_{Id/A} = (M, M, Id_A)$ on montre que $\forall M, \forall A$, $\Re_{Id/A}$ est une relation de bissimulation de maximalité sur l'ensemble des transitions.

Soit $\Re_{Id/A} = \underset{i\geq 0}{\cup} \Re_i$ tel que :

$\Re_0 = (M, M, Id_A)$

$\Re_1 = (M_1, M_1, Id_{A_1})$ tel que $\exists M \xrightarrow{Etx} M_1$ et $A_1 = (A - E) \cup \{x\}$

jusqu'à $\Re_{n+1} = \left(M_{n+1}, M_{n+1}, Id_{A_{n+1}}\right)$ tel que $\exists (M_n, M_n, Id_{A_n})$ et $\exists M_n \overset{E_{n+1}tx_{n+1}}{\longrightarrow} M_{n+1}$ avec $A_{n+1} = (A_n - E_{i+1}) \cup \{x_{i+1}\}$.

$\forall (M_i, M_i, Id_{A_i}) \in \Re_{Id/A}, \forall A_i \subseteq \psi(M_i)$

Si $M_i \overset{E_{i+1}tx_{i+1}}{\longrightarrow} M_{i+1}$ alors $\exists E_{i+1}^{'} = E_{i+1}$ et $\exists x_{i+1}^{'} = x_{i+1}$ tel que $M_i \overset{E_{i+1}^{'}tx_{i+1}^{'}}{\longrightarrow} M_{i+1}$. D'autre part $\forall u \in A_i$, Si $u \notin E_{i+1}$ alors $u \notin E_{i+1}^{'}$ parceque $E_{i+1}^{'} = E_{i+1}$ alors $\left(M_{i+1}, M_{i+1}, Id_{A_{i+1}}\right) \in \Re_{Id/A}$ tel que $A_{i+1} = (A_i - E_i) \cup \{x_{i+1}\}$.On déduit que $\Re_{Id/A}$ est une relation de bissimulation de maximalité sur l'ensemble des transitions.

3. $B = \psi(M) - A$ avec $A \subseteq \psi(M)$.Soit $M^{'} = makefree(B, M)$: on construit une relation $\Re$ contenant $\left(M^{'}, M, Id_A\right)$, et on montre que $\Re$ est une relation de bissimulation de maximalité pour un ensemble de transitions T.

Soit $\Re = \{\left(M^{'}, M, Id_{\psi\left(M^{'}\right)}\right)\}/M^{'} = makefree\left(\psi(M) - \psi\left(M^{'}\right), M\right)$, $\psi(M) - \psi\left(M^{'}\right) = B$ et $\psi\left(M^{'}\right) = A$.

i. Si $M^{'} \xrightarrow{Ftu} M_1^{'}, \exists\, c \subseteq B/\ M \xrightarrow{C\cup F} M_1$ avec : $u = get\left(\mathcal{M} - \left(\psi\left(M^{'}\right) - F\right)\right)$

et $v = get(\mathcal{M} - (\psi(M) - (C \cup F)))$

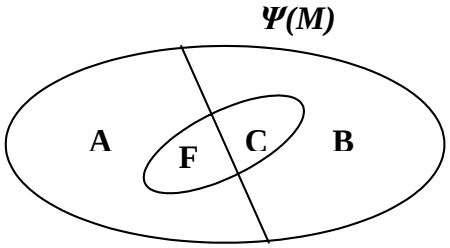


FIG. 4.20 – L'ensemble des noms des évènements correspondant au marquage M.

$\psi\left(M_1^{'}\right) = \left(\psi\left(M^{'}\right) - F\right) \cup \{u\}$
$\psi\left(M_1^{'}\right) = ((\psi(M) - B) - F) \cup \{u\}$
$\psi(M_1) = (\psi(M) - (C \cup F)) \cup \{v\}$
$\psi(M_1) = ((\psi(M) - C) - F) \cup \{v\}$
$\psi(M_1) - \{v\} = \left(\psi\left(M_1^{'}\right) - \{u\}\right) \cup (B - C)$
Soit : $w = get\left(\mathcal{M} - \left(\left(\psi\left(M^{'}\right) - F\right) \cup (\psi(M) - (C \cup F))\right)\right)$ or $\psi\left(M^{'}\right) - F \subseteq \psi(M) - (C \cup F)$
$w = get(\mathcal{M} - (\psi(M) - (C \cup F))) \implies w = v$
$\psi(M_1) = \left(\psi\left(M_1^{'}[v/u]\right) \cup (B - C)\right)$ or $(B - C) \cap \psi\left(M_1^{'}[v/u]\right) = \phi.$
$makefree((B - C), M_1) = M_1^{'}[v/u] \Longrightarrow \left(M_1^{'}[v/u], M_1, Id_{\psi\left(M_1^{'}[v/u]\right)}\right) \in \Re$
Soit : $x \notin A \cap F$ or $A \cap B = \phi \Longrightarrow x \notin A \cap (B \cup F)$ et puisque $C \subseteq B \Longrightarrow x \notin A \cap (C \cup F)$.
ii. Si $M \xrightarrow{C \cup F} M_1$ alors $\exists\ M^{'} \xrightarrow{Ftu} M_1^{'}/c \subseteq B$ avec :
$v = get(\mathcal{M} - (\psi(M) - (C \cup F)))$ et $u = get\left(\mathcal{M} - \left(\psi\left(M^{'}\right) - F\right)\right)$
en a montré dans 3.i que : $\psi(M_1) = \left(\psi\left(M_1^{'}[v/u]\right) \cup (B - C)\right)$
$makefree((B - C), M_1) = M_1^{'}[v/u] \Longrightarrow \left(M_1^{'}[v/u], M_1, Id_{\psi\left(M_1^{'}[v/u]\right)}\right) \in \Re$
soit $x \notin A \cap (C \cup F) \Longrightarrow x \notin (A \cap C) \cup (A \cap F)$ or $A \cap c = \phi \Longrightarrow x \notin A \cap F$
de (i) et (ii) $\Re$ est une ralation de bissimulation de maximalité par rapport à l'ensemble de transitions T. □

Proposition 4.4 *Soit Σ un système dont le marquage initial est M_0, et soit σ une séquence de transitions tel que $M_0[\sigma > M$ et $M_0[\sigma > M^{'}$ avec $D \subseteq \psi(M)$, $C \subseteq \psi\left(M^{'}\right)$ et $f : D \longrightarrow C$ une bijection ; alors :*

$M \approx_m^T /f\ M^{'} \Rightarrow makefree(\psi(M) - D, M) \approx_m^T /f\ makefree\left(\psi\left(M^{'}\right) - C, M^{'}\right)$.

Preuve. on a de 4.3 $M \approx_m^T /Id_D\ makefree(\psi(M) - D, M)$ et
$M^{'} \approx_m^T /Id_C\ makefree\left(\psi\left(M^{'}\right) - C, M^{'}\right)$. on a par hypothèse : $M \approx_m^T /f\ M^{'}$, donc il existe $f : \psi(M) \longrightarrow \psi\left(M^{'}\right)$. D'autre part on a $f : D \longrightarrow C$ alors

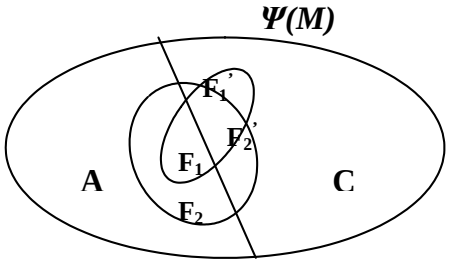


FIG. 4.21 – Division de l'ensemble M

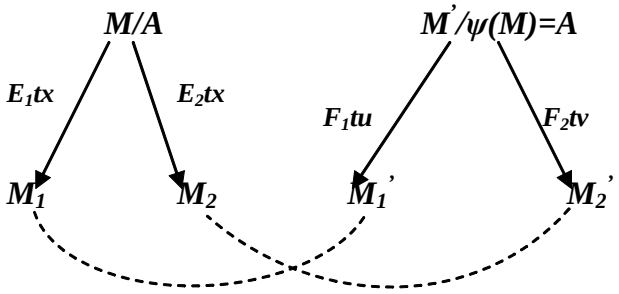


FIG. 4.22 – Technique de la preuve

$f : \psi(M) - D \longrightarrow \psi\left(M'\right) - C$. Par transitivité on déduit que $makefree(\psi(M) - D, M) \approx_m^T / f\ makefree\left(\psi\left(M'\right) - C, M'\right)$ □

Proposition 4.5 *Soit* $f = Id_A$ *tel que* $A \subseteq \psi(M)$. *Si* $M \xrightarrow{E_1 tx} M_1$ *et* $M \xrightarrow{E_2 ty} M_2$ *tel que* $A \cap E_1 \subseteq A \cap E_2$ *alors pour* $z = get(\mathcal{M} - ((\psi(M) - E_1) \cup (\psi(M) - E_2)))$ *et* $B = (A - E_2) \cup \{z\}$ *on a* $M_1[z/x] \approx_m^T / Id_B\ M_2[z/y]$.

Preuve. Soit $C = \psi(M) - A$, $M' = makefree(c, M)$

Proposition 4.3.(1) $\Longrightarrow M' \approx_m^T / Id_A\ M$. Soit $F_1 = A \cap E_1, F_2 = A \cap E_2$ alors $F_1 \subseteq F_2$.

Soient les noms d'événements x, y, u et v définis comme suit :

$x = get(\mathcal{M} - (\psi(M) - E_1))$

$y = get(\mathcal{M} - (\psi(M) - E2))$

$u = get\left(\mathcal{M} - \left(\psi(M') - F_1\right)\right)$

$v = get\left(\mathcal{M} - \left(\psi(M') - F_2\right)\right)$Par hypothèse de la proposition on a $\psi(M_1) =$

$(\psi(M) - E_1) \cup \{x\}$ d'où $\psi(M_1) = \left((C \cup F) - \left(F_1' \cup F_2'\right)\right) \cup \{x\}$

Les ensembles E1 et E2 peuvent être partitionnés de la manière suivante :

$F_1^{'} \cup F_1 = E_1$ et $F_1 \cap F_1^{'} = \varnothing$ tel que $\forall x \in F_1^{'}$ alors $x \in C$ et $F_1 \subseteq A$
$F_2^{'} \cup F_2 = E_2$ et $F_2^{'} \cap F_2 = \varnothing$ tel que : $\forall x \in F_2^{'}$, $x \in C$ et $F_2 \subseteq A$.
Alors :
$\psi(M_1) = \left(\left(C - F_1^{'}\right) \cup (A - F_1)\right) \cup \{x\}$
$\psi\left(M_1^{'}\right) = (A - F_1) \cup \{x\}$
$M_1^{'} = makefree\,(C - F_1, M_1)\,[u/x]$
Etant donné que $\psi(M_2) = (\psi(M) - E_2) \cup \{y\}$ alors $\psi(M_2) = \left((C \cup A) - \left(F_2^{'} \cup F_2\right)\right) \cup \{y\}$ d'où $\psi(M_2) = \left(\left(C - F_2^{'}\right) \cup (A - F_2)\right) \cup \{y\}$.

D'autre part $\psi\left(M_2^{'}\right) = (A - F_2) \cup \{v\}$ alors $M_2^{'} = makefree\left(\left(C - F_2^{'}\right), M_2\right)[v/y]$
Proposition 4.3 implique :
$M_1 \approx_m^T /Id_{((C-F_1)\cup\{x\})}\ makefree\left(\left(C - F_1^{'}\right), M_1\right)$ d'où $M_1 \approx_m^T /Id_{((A-F_1)\cup\{x\})}$ $M_1^{'}$ et
$M_2 \approx_m^T /Id_{((A-F_2)\cup\{y\})}\ makefree\left(\left(C - F_2^{'}\right), M_2\right)$ d'où $M_2 \approx_m^T /Id_{((A-F_2)\cup\{y\})}$ $M_2^{'}$
On veut montrer que $M_1 \approx_m^T /f\ M_2$ tel que $f = Id_{(A-F_2)} \cup \{(x, y)\}$. Pour cela il suffit de montrer que :
$M_1^{'} \approx_m^T /f^{'}\ M_2^{'}$ tel que $f^{'} = Id_{(A-F_2)} \cup \{(u, v)\}$. On remarque que $f^{'}$ est une bijection.
on a : $u = get\left(\mathcal{M} - \left(\psi\left(M^{'}\right) - F_1\right)\right) = get\,(\mathcal{M} - (A - F_1))$
$v = get\left(\mathcal{M} - \left(\psi\left(M^{'}\right) - F_2\right)\right) = get\,(\mathcal{M} - (A - F_2))$
Soit $w = get\,(\mathcal{M} - (A - (F_1 \cup F_2)))$ or $F_1 \subseteq F_2$ alors :
$w = get\,(\mathcal{M} - (A - F_2)) = v$
On remarque donc que $v \notin \left(\psi\left(M_1^{'}\right) - \{u\}\right)$.
Soit $M_1^{''} = M_1^{'}[v/u,]$ et $F = F_2 - F_1$, alors $makefree\left(F, M_1^{''}\right) = M_2^{'}$

Proposition 4.3 implique $M_1^{''} \approx_m^T /Id_{((A-F_2)\cup\{v\})}\ M_2^{'}$, on peut déduire donc que $M_1 \approx_m^T /f\ M_2$. □

Théorème 4.1 *Soit $\Sigma = (S, T, W)$ un système étiqueté, soit $stem_1$ (réspectivement $stem_2$) le système de transitions étiquetées maximales généré de Σ, en utilisant la sémantique de maximalité, (réspectivement la sémantique de maximalité avec agrégation de transitions) alors $stem_1$ et $stem_2$ sont maximalement bissimilaires pour l'ensemble des transitions T.*

4.6 Etude de cas :

4.6.1 Dîner des philosophes :

Afin d'illustrer l'intérêt de l'approche proposée, on étudie dans cette section un exemple classique de synchronisation des processus, à savoir le paradigme du dîner

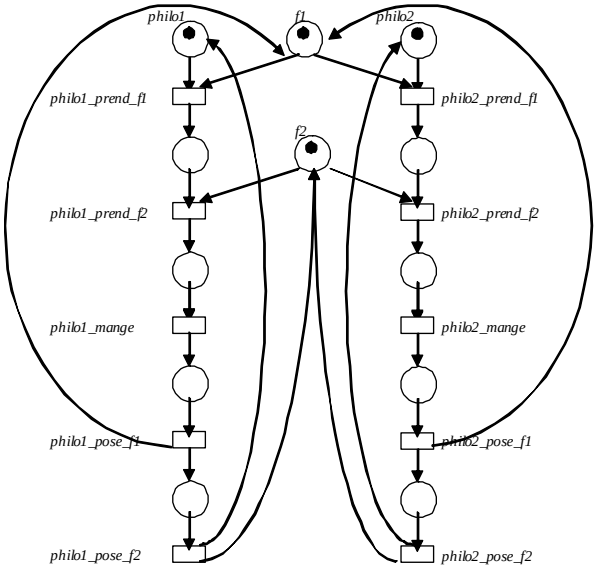


FIG. 4.23 – Réseau spécifiant le dîner de deux philosophes.

des philosophes [Dij71]. La spécification est donnée entre autres par un réseau et un système de transitions étiquetées maximales. L'accent sera mis sur la possibilité d'exprimer des comportements cycliques, dont le paradigme du dîner des philosophes constitue un bon exemple, à l'aide de systèmes de transitions étiquetées maximales, et d'autre part sur l'expression des propriétés de sûreté et de vivacité attendues de manière naturelle.

Spécification :

Considérons deux philosophes assis sur une table et partageant deux fourchettes `f1` et `f2`. Un philosophe ne peut manger que s'il a la possession des deux fourchettes et exécute dans l'ordre les actions $\texttt{philo}_i\texttt{\_prend\_f1}$, $\texttt{philo}_i\texttt{\_prend\_f2}$, $\texttt{philo}_i\texttt{\_mange}$, $\texttt{philo}_i\texttt{\_pose\_f1}$ et $\texttt{philo}_i\texttt{\_pose\_f2}$. La spécification de cet exemple sera ainsi donnée en terme des figures 4.23 et 4.24 illustrant respectivement le réseau de Petri et le système de transitions étiquetées maximales correspondant.

Vérification :

Les propriétés à vérifier sont exprimées dans la logique temporelle CTL [CES86, Eme90] de manière naturelle en raisonnant directement sur les actions. Ceci est rendu possible en considérant chaque action comme une proposition logique en partant de l'idée que les états des systèmes de transitions étiquetées maximales englobent l'information sur l'exécution potentielle des actions. De nouvelles propriétés sont ainsi exprimées, portant sur la compatibilité ou l'incompatibilité de plusieurs actions

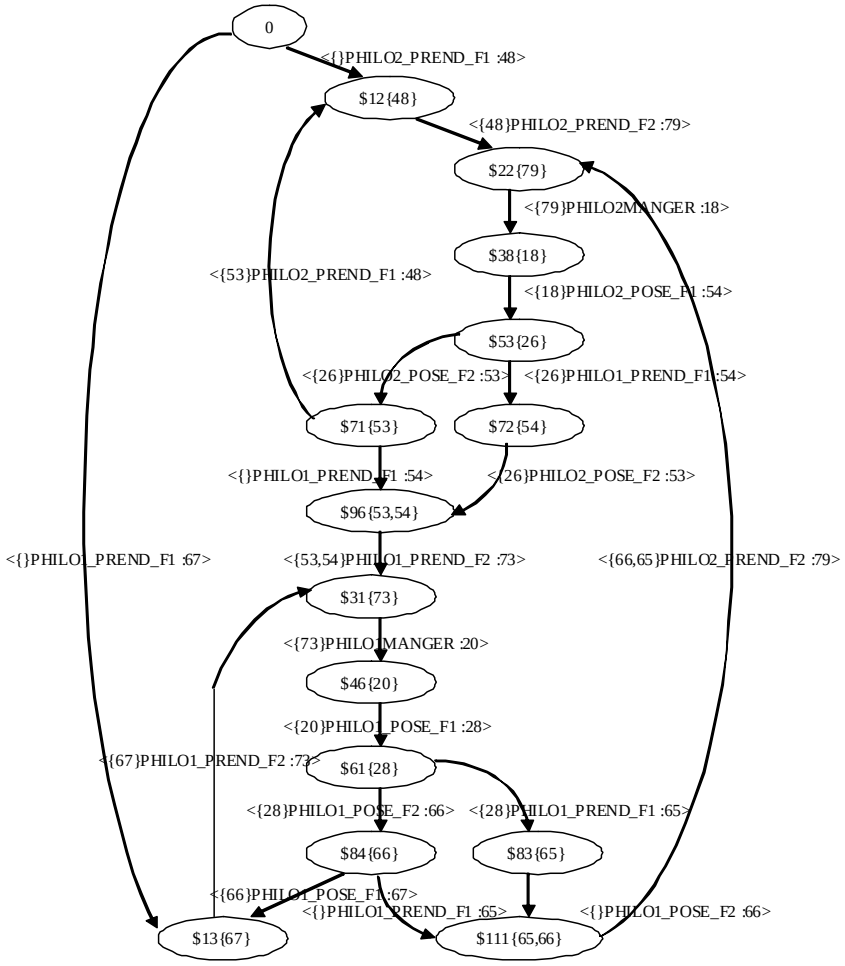


FIG. 4.24 – Système de transitions étiquetées maximales correspondant au dîner de philosophes.

et le degré d'auto-concurrence [SB05]. Toutes les propriétés qui seront montrées dans cette section ont été vérifiées par l'outil FOCOVEqui supportent la spécification et la vérification des structures des systèmes de transitions étiquetées maximales utilisant la logique temporelle CTL.

Exclusion mutuelle :

L'exclusion mutuelle peut concerner l'utilisation de chaque fourchette et exprimée ainsi : `A G (not(philo1_prend_f1 and philo2_prend_f1) and not(philo1_prend_f2 and philo2_prend_f2))` ou bien le fait de ne pas avoir à un instant donné la possibilité de l'exécution en concurrence des deux actions `philo1_mange` et `philo2_mange`. Ce dernier cas peut être exprimé par la formule `A G not(philo1_mange and philo2_mange)`.

Absence d'interblocage :

Dans notre exemple, la situation d'interblocage survient quand chaque philosophe a la possession d'une fourchette et requiert une autre fourchette déjà prise par l'autre philosophe. Cette situation mène à une attente circulaire. L'absence d'interblocage signifie que le système peut toujours évoluer dans le future dans chaque état. Cette propriété est exprimée par `A G (E X true)`.

Absence de famine :

L'absence de famine est exprimée par le fait qu'à chaque instant où un philosophe veut manger, il pourra le faire dans le futur. Cette propriété peut être exprimée pour un philosophe i par `A G (philo`$_i$`_prend_f1` $\Rightarrow$ `A F philo`$_i$`_mange)`.

4.6.2 Agence de réservation de billets d'avion :

Maintenant pour montrer l'intérêt de l'approche de réduction, on étudie dans cette section un exemple simple : le système de réservation de billet d'avion. A travers cet exemple nous montrons le taux de réduction considérable, ainsi que la préservation des propriétés de système à vérifier.

Spécification :

On considère le système de l'agence de réservation de billets d'avion. Pour acheter un billet, on passe généralement par deux guichets. Un premier guichet de type R (Réservation) permet de réserver une place dans un vol donnée et d'établir le billet d'avion. Un second guichet de type C (Caisse) permet d'encaisser l'argent et de remettre le billet au client. Sachant que cette agence dispose d'une salle d'attente, de trois guichets de type R et de deux guichets de type C. En arrivant, les clients passent d'abord en salle d'attente. Dès qu'un guichet de type R est libre, un client peut se présenter au guichet et procéder à la réservation. Une fois cette opération terminée, il attend qu'un guichet caisse soit libre pour procéder au paiement et au retrait de son billet.

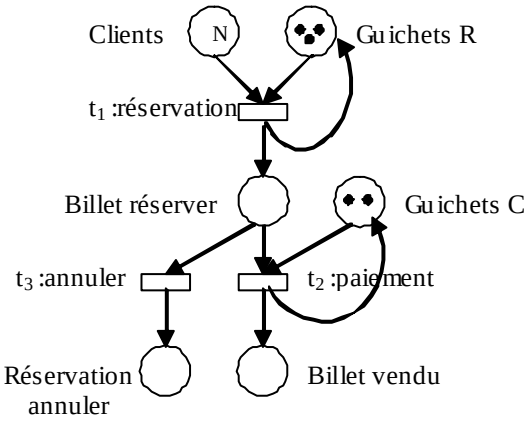


FIG. 4.25 – Système de réservation de billets d'avion.

La modélisation de ce système est donnée par le réseau de Petri de la figure 4.25.

Résultats de génération :

Afin de montrer le taux de réduction, nous appliquons d'abord la méthode de génération de systèmes de transitions étiquetées maximale présenté dans la section 4.3, ensuite nous appliquons l'approche de génération avec réduction. Ceci est fait en variant à chaque fois le nombre de clients. Les résultats obtenus sont résumés dans le tableau 2. A partir de ce dernier on observe que le nombre d'état diminue d'environ 30% (voir la figure 4.26), quand au nombre de transitions il diminue d'environ 50% (voir la figure 4.27).

	Avant réduction		Après réduction		Taux de réduction	
Nbre client	état	transition	état	transition	état	transition
1	4	3	4	3	0%	0%
2	17	28	13	18	24%	36%
3	62	153	40	81	35%	47%
4	212	665	129	326	39%	51%
5	696	2572	423	1233	39%	52%
6	2202	9100	1376	4436	38%	51%
7	6658	29686	4303	14926	35%	50%
8	19135	89929	12801	47000	33%	48%

TAB 4.1- Résultats de réduction.

4.7 Conclusion

Dans ce chapitre, nous avons proposé une méthode opérationnelle de génération de systèmes de transitions étiquetées maximales associés aux réseaux de Petri. De

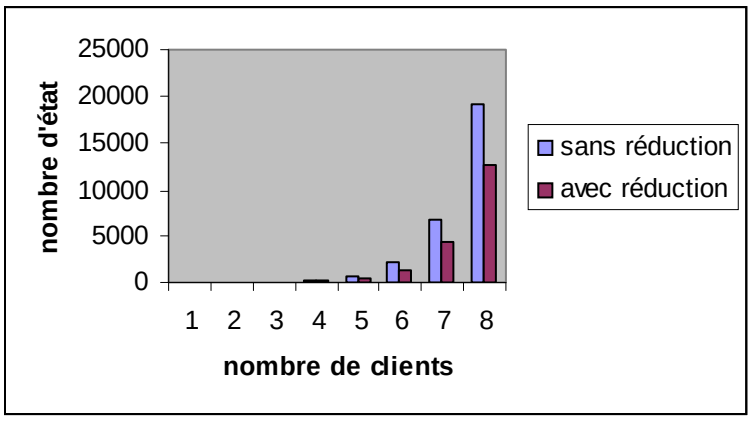


FIG. 4.26 – Taux de réduction du nombre d'états.

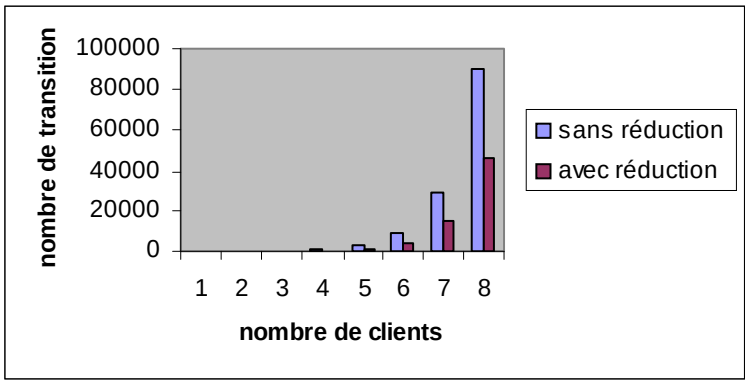


FIG. 4.27 – Taux de réduction du nombre de transitions.

ce fait, les propriétés relatives au bon fonctionnement d'un systèmes spécifié par un réseau de Petri peuvent être vérifiées sur le système de transitions étiquetées maximales qui lui correspond. D'autre part, la structure des systèmes de transitions étiqueteés maximales intègre en elle l'information sur l'exécution parallèle des actions. Cette structure nous permet d'exprimer plus facilement les propriétés relatives à l'exécution parallèle des actions sans avoir à recourir à l'éclatement des actions en début et fin. Il est à noter que l'éclatement des actions en début et fin pour capturer l'exécution parallèle des actions contribue directement à l'explosion combinatoire du graphe d'états.

Dont le but de réduire la taille des systèmes de transitions étiquetées maximales, nous avons proposé une nouvelle méthode de génération basée sur l'agrégation des transitions redondantes, et nous avons montré que les STEMs obtenus sont équivalents à celles générées sans agrégation des transitions, modulo la relation de bissimulation de maximalité. Il est à noter que les propriétés du système à vérifier et le degré de parallèlisme sont préservés par cette agrégation de transitions.

A la fin de ce chapitre nous avons étudié deux exemples : l'exemple du dîner des philosophes et l'exemple de l'agence de réservation de billets d'avion. Le premier exemple a montré l'aptitude de l'approche de maximalité proposée pour l'interprétation des systèmes à comportement cyclique et la vérification des propriétés de bon fonctionnement. Quand au deuxième exemple il a montré que l'approche d'agrégation des transitions proposée réduit significativement la taille des sytèmes de transitions étiquetées maximales ce qui permet d'atténuer, du moins partiellement, les effets du problème de l'explosion combinatoire du graphe d'états.

Chapitre 5

Implémentation :

5.1 Problématique :

Dans le présent travail nous avons enrichi l'environnement de vérification formelle FOCOVE par l'outil « MSOS-PN » pour « Maximality-based Structured Operational Semantics for Petri Net ». Dans ce chapitre, nous présentons la conception globale et les techniques adoptées pour implémenter cet outil. L'outil «MSOS-PN » est un outil paramétré qui permet de traduire un réseau de Petri en un système de transitions étiquetées maximales, ce qui permet sa vérification. Le mot paramétré signifie que l'outil prend comme paramètre une variable booléenne selon laquelle, il génère soit un système de transitions étiquetées maximales, soit un système de transitions étiquetées maximales t-réduit, ce qui contribue à la maîtrise du problème de l'explosion combinatoire du graphe d'états. Il est à noter que la méthode de réduction est faite à la volée, dont le principe consiste à détecter et supprimer les états équivalents modulo la relation de bissimulation de maximalité pour l'ensemble de transitions d'un réseau de Petri. L'outil «MSOS-PN» a comme entrée un réseau de Petri généré à l'aide de l'outil Tina (Time Petri Net Analyser). Cette spécification est compilée pour donner un code intermédiaire à partir du quel un STEM est généré. Ce dernier est stocké dans un fichier pour pouvoir ensuite être restauré, comme il peut être visualisé graphiquement en utilisant l'outil STEM Viewer ou l'outil Dot.

Notant que ocaml est le langage de programmation choisi, pour l'implémentation du noyau de cet outil, ce choix n'est pas fait aléatoirement et sera justifié par la suite.

5.2 L'environnement FOCOVE :

FOCOVE [1]est une boîte à outils pour la compilation, la vérification et la validation de programme LOTOS. La conception de l'architecture globale a été élaborée par Dr Djamel-Eddine Saidouni du département d'informatique de l'université Mentouri de constantine algérie en l'an 2000.

[1] `http ://www.focove.new.fr`.

FOCOVE intègre un ensemble cohérent d'outils permettant de traiter des spécifications comportementales et logiques, en utilisant les méthodes basées sur les modèles. L'environnement comprend divers outils, nous pouvons les classer en deux classes :

- **Les compilateurs :** utilisés en amont, leur rôle est de traduire le programme à vérifier vers un modèle sémantique du parallélisme. FOCOVE contient cinq compilateurs :

- ICC [BB01] (Interleaving CCS Compiler) : En entrée, il prend le programme CCS à vérifier et il nous fournit comme sortie un STE.

- ILC [**?**] (Interleaving LOTOS Compiler) : Nous permet d'obtenir un STE à partir d'une spécification LOTOS.

- LOTOSTEM [**?**] : au niveau de ce compilateur, nous avons adopté le modèle des arbres maximaux, ces arbres sont dérivés à partir d'une spécification LOTOS.

- LOTOSTEMv2.0 : cherche à générer à la volé un STEM réduit dont la vérification consiste à détecter et supprimer les états alpha-équivalents.

- LOTOSGPM : est un outil utilisant l'approche d'ordre partiel, il fournit à la volé un graphe qui couvre le STEM initial modulo l'équivalence de trace de Mazurckiewicz.

- **Les vérificateurs :** utilisés en aval, ils effectuent des vérifications sur les graphes générés par les compilateurs. FOCOVE couvre les quatre types de vérifications :

- La vérification comportementale : BSTE [BB01] et DBRE [Bel02] sont deux outils qui permettent respectivement la réduction et la comparaison des STEs par rapport à des relations d'équivalence comportementales à savoir la bissimulation forte et faible.

- La vérification logique : LotoStem contient un model-checker qui permet de vérifier des propriétés exprimées en logique temporelle arborescente (CTL) sur le STEM généré.

- La vérification symbolique : MSMC [DG02](Maximality based Symbolic Model Checking), il prend comme arguments une spécification de système à vérifier modélisée par un système de transitions étiquetées maximales, et une spécification des propriétés à vérifier exprimées en logique temporelle arborescente. L'implémentation de cet outil est basée sur l'utilisation des diagrammes de décision binaire (BDDs) qui permettent d'éviter la construction explicite du graphe d'états.

- La vérification par test : TEST [MF76] est un outil de mise en oeuvre de l'approche test, apportant une certitude mathématique par génération de tests de conformité sous forme d'une structure appelée "graphe de refus", et par extension et adaptation du calcul de bissimulation pour la vérification des relations de conformité d'une implémentation par rapport à sa spécification.

La boîte à outils FOCOVE fournit aussi deux outils permettant la visualisation graphique du STE (STEViewer [BBB01]) et du STEM (STEMViewer [BDS02]).

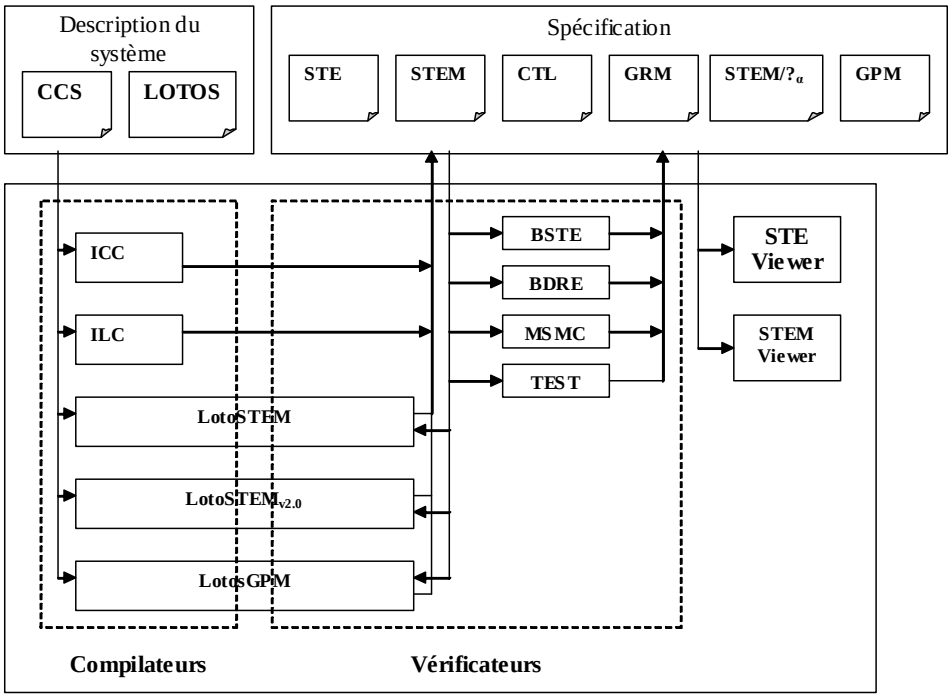


FIG. 5.1 – Architecture globale de l'environemment FOCOVE.

5.3 Outils utilisés :

5.3.1 Outil Tina :

Cette section présente un environnement de vérification formelle développé au LAAS/CNRS, connu sous le nom de Tina (TIme Petri Net Analyser). Tina est disponible à l'adresse "http ://www.laas.fr/tina". Dans notre travail nous avons utilisé la version 2.8.4.

Tina est un outil logiciel permettant l'édition et l'analyse de réseaux de Petri et réseaux de Petri Temporels [MF76]. Il offre les fonctions classiques d'édition et d'analyse énumérative (les graphes de marquages, les arbres de couverture) ou structurelle (semi-flots).

Tina n'est pas un model-checker au sens où il ne permet pas de vérifier la satisfaction de propriétés exprimées dans des logiques, sauf les propriétés générales d'accessibilité (le blocage et la vivacité).

Tina est une boîte à outils qui contient trois logiciels dont l'utilisation peut être combinée ou indépendante :

- Editeur graphique : il fournit des fonctions de dessin de réseaux de Petri et d'automates.
- Outil de construction de comportement et d'analyse structurelle.

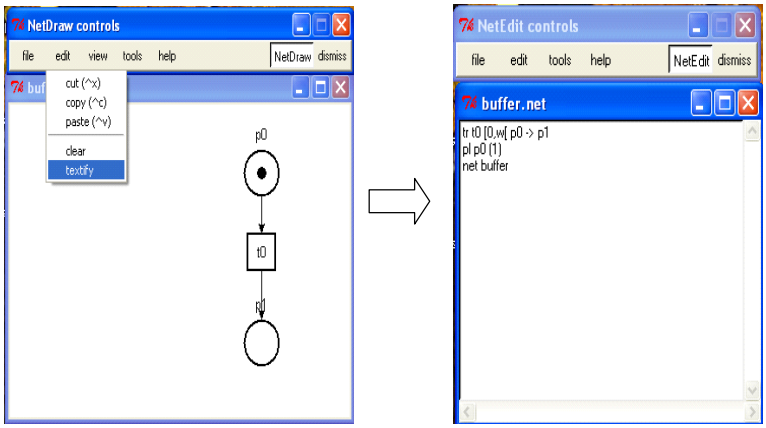


FIG. 5.2 – L'outil tina.

- Outil de simulation de réseaux de Petri et réseaux de Petri Temporels, il est très utile pédagogiquement.

Cette variété de construction, permet à Tina de se positionner dans l'enseignement et également dans l'industrie, à titre d'exemple, Tina est figuré parmi les environnement de vérification retenus dans le cadre du projet RNTL COTRE (Composants Temps Réels http ://www.laas.fr/cotre).

Au sein de notre travail nous avons utilisé l'éditeur graphique de Tina, nommé « NetDraw », pour dessiner les réseaux de Petri. La commande Edit -> Textify permet d'obtenir le fichier « net » de réseaux (voir la figure 5.2). Ce dernier est l'entrée de notre outil « MSOS-PN » .

5.3.2 L'outil Dot :

L'outil dot est un outil permettant de dessiner des graphes orientés. Il lit les informations de graphe à dessiner à partir d'un fichier texte et il donne le dessin correspondant dans un fichier de format graphique tel que gif, JPEG...etc.

Dot accepte le fichier texte en entrée dans le langage DOT. Ce langage décrit trois sortes d'objets : le graphe, les nœuds et les arcs.

Exemple :

On va prendre en entrée la spécification du stem correspondant à l'exemple de deux lecteurs et deux rédacteurs :

digraph G
{edge [color = blue, fontname = Arial, fontsize = 8, fontcolor = blue] ;
node [fontname = Arial, fontsize = 8, fontcolor = red]
1[label = 1]
1->2[label="{}*lire :0"]
1->3[label="{}*ecrire :0"]
2[label = 2]

```
2->4[label="{0}*lire :0"]
2->5[label="{}*lire :1"]
2->6[label="{0}*ecrire :0"]
3[label = 3]
3->7[label="{0}*lire :0"]
3->8[label="{0}*ecrire :0"]
4[label = 4]
4->9[label="{0}*ecrire :0"]
5[label = 5]
5->9[label="{01}*ecrire :0"]
6[label = 6]
6->10[label="{0}*lire :0"]
6->11[label="{0}*ecrire :0"]
7[label = 7]
7->10[label="{0}*lire :0"]
7->12[label="{}*lire :1"]
7->11[label="{0}*ecrire :0"]
8[label = 8]
8->13[label="{0}*lire :0"]
9[label = 9]
9->14[label="{0}*ecrire :0"]
10[label = 10]
10->14[label="{0}*ecrire :0"]
11[label = 11]
11->15[label="{0}*lire :0"]
12[label = 12]
12->14[label="{01}*ecrire :0"]
13[label = 13]
13->15[label="{0}*lire :0"]
13->16[label="{}*lire :1"]
14[label = 14]
15[label = 15]
16[label = 16]
}
```

La sortie du graphe en format JPEG est donnée par la figure 5.3.

5.3.3 Description du langage objective caml :

Ocaml est une implémentation du langage ML, basé sur la syntaxe du caml light, avec un système d'objets basé sur les classes, et un système de modules. Il possède les caractéristiques suivantes :

- Objective CAML est un langage fonctionnel :

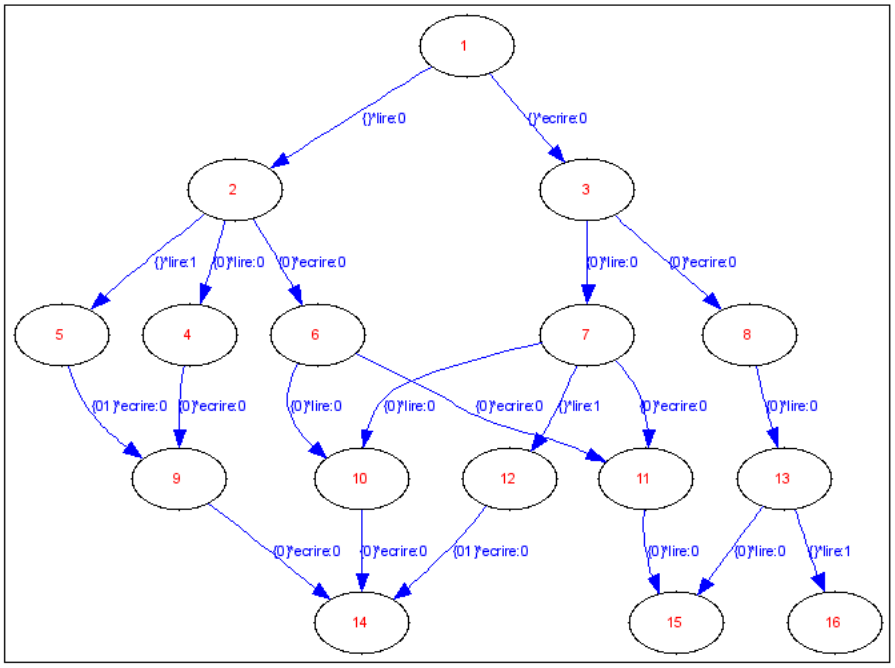


FIG. 5.3 – Petit exemple.

Il manipule les fonctions comme étant des valeurs du langage. Celles-ci peuvent être utilisées en tant que paramètres d'autres fonctions ou être retournées comme résultat d'un appel de fonction.

- Objective CAML est à typage statique :

La vérification de la compatibilité entre les types des paramètres formels et des paramètres d'appel est effectuée au moment de la compilation du programme. Dès lors, il n'est pas nécessaire de faire ces vérifications durant l'exécution du programme, ce qui accroît son efficacité. En outre, la vérification de type permet d'éliminer la plupart des erreurs introduites par maladresse, ce qui contribue à la sûreté de l'exécution.

- Objective CAML est polymorphe paramétrique :

Une fonction qui n'explore pas la totalité de la structure d'un de ses arguments accepte que celui-ci ait un type non entièrement déterminé. Ce paramètre est alors dit polymorphe. Cette particularité permet de développer un code générique utilisable pour des structures de données différentes tant que la représentation exacte de cette structure n'a pas besoin d'être connue par le code en question.

- Objective CAML possède l'inférence de types :

Le programmeur n'a besoin de donner aucune information de type à l'intérieur de son programme. Le langage se charge seul de déduire du code le type le plus général des expressions et des déclarations qui y figurent. Cette inférence est effectuée conjointement à la vérification, lors de la compilation du programme.

- Objective CAML est muni d'un mécanisme d'exceptions :

Il est possible de rompre l'exécution normale d'un programme à un endroit et de la reprendre à un autre endroit du programme prévu à cet effet. Ce mécanisme permet de gérer les situations exceptionnelles.

- Objective CAML possède des traits impératifs :

Les entrées-sorties, les modifications physiques de valeurs et les structures de contrôle itératives sont possibles sans avoir recours aux traits de la programmation fonctionnelle. Le mélange des deux styles est bien entendu accepté et offre une grande souplesse de développement ainsi que la possibilité de définir des structures de données nouvelles.

- Objective CAML dispose de nombreuses bibliothèques :

Structures de données classiques, entrées-sorties, interfaçage avec les ressources du système, analyses lexicale et syntaxique... etc.

Tous cela nous a conduit à juger ocaml comme le langage idéal pour nos besoins.

5.4 Développement de l'outil « MSOS-PN » :

5.4.1 Conception globale :

Dans cette section notre objectif principal est le développement d'un outil permettant, selon le choix, la traduction d'un réseau de Petri soit en un système de transitions étiquetées maximales, soit en un système de transitions étiquetées maximales t-réduit. Cet outil prend comme entrée un fichier Net généré par l'outil Tina, et produit en sortie, soit un fichier dot et un fichier STEM, soit un fichier dot et un fichier STEM t-réduit. Ce qui est montré par la figure 5.4.

Afin de maîtriser la complexité de développement de cet outil, nous avons adopter une technique de conception hiérarchique de cet outil selon les fonctionnalités (voir la figure 5.5).

Comme première étape, nous avons décomposé l'outil Pe-master en trois modules :

- Module « compil » : il se charge de l'analyse lexicale, donc il découpe le fichier Net en des jetons lexicales.
- Module « noyau » : il se préoccupe de la tâche principale pour la génération des fichier STEM et DOT.

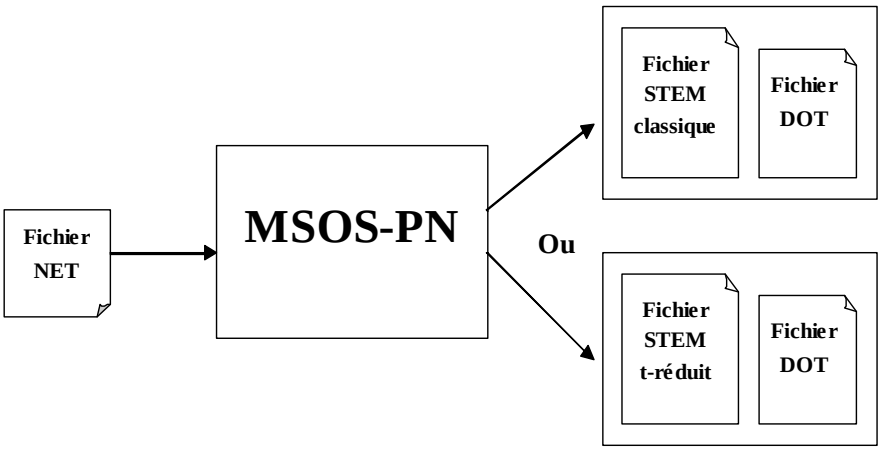


FIG. 5.4 – Fonction globale de l'outil MSOS-PN.

- Module « main » : il assure l'interfaçage entre les deux modules précédents car la sortie du module « compil » est l'entrée du module « noyau ».

Cependant la tâche effectuée par le module « noyau », peut être décomposée en quatre procédures :

- Procédure « analyse_syn » : effectue l'analyse syntaxique, ainsi que le traitement et le chargement de données.
- Procédure « genere » : prend en entrée les données chargées par « analyse_syn » et produit en sortie un tableau de configurations.
- Procédure « genere_dot » : génère un fichier DOT à partir d'un tableau des configurations.
- Procédure « genere_stem » : génère un fichier STEM à partir d'un tableau des configurations.

5.4.2 Conception détaillée :

Dans cette section nous présentons les structures de données, les techniques et les algorithmes utilisés.

Structure de donnée :

En général durant la programmation, il est nécessaire d'identifier et de spécifier des données assez tôt dans le processus de développement. Une fois la donnée complexe identifiée, on s'intéresse à sa spécification. Le but de cette spécification est de définir l'interface d'utilisation de cette donnée, c'est-à-dire toutes les opérations permises sur ce type, le type des arguments qu'elle prend et le type des résultats qu'elles retourne.

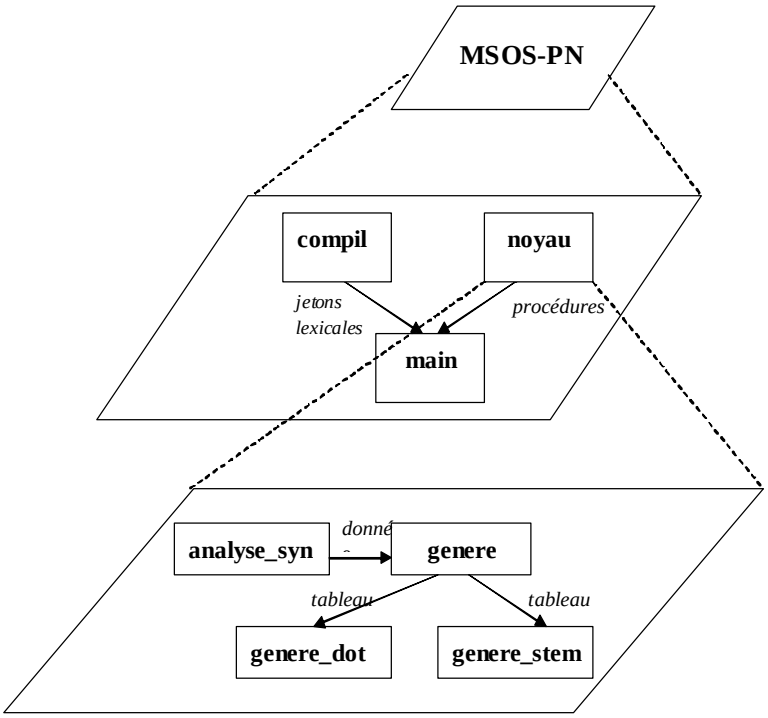


FIG. 5.5 – Conception hiérarchique de l'outil MSOS-PN.

Type ensemble
utilise : $entier, booléen$
opérations :
$créer\_ensemble : \quad -> ensemble$
$ajouter : ensemble \times entier -> ensemble$
$vide : ensemble -> booléen$
$member : entier \times ensemble -> booléen$
$union : ensemble \times ensemble -> ensemble$
$intersection : ensemble \times ensemble -> ensemble$
$inclus : ensemble \times ensemble -> booléen$
$difference : ensemble \times ensemble -> ensemble$
Type multi-ensemble
utilise : $ensemble, booléen$
opérations :
$créer\_multi - ensemble : \quad -> multi - ensemble$
$vide\_member : multi - ensemble -> booléen$
$diff\_dis : ensemble \times ensemble -> multi - ensemble$
$partition : ensemble -> multi - ensemble$
$conflit : ensemble -> multi - ensemble$
$conflit\_red : ensemble -> multi - ensemble$
Type item

utilise : $chaîne\ de\ caractère, entier$
opérations :
$créer\_item : \quad chaîne\ de\ caractère \times entier -> item$
$lire\_nomplace : item -> chaîne\ de\ caractère$
$lire\_jeton : item -> entier$
Type condition
utilise : $item, condition$
opérations :
$créer\_condition : \quad -> condition$
$ajouter\_item : condition \times item -> condition$

Type transition
utilise : $chaîne\ de\ caractère, condition$
opérations :
$créer\_transition : chaîne\ de\ caractère \times condition \times condition -> tranition$
$lire\_nom\_transition : transition -> chaîne\ de\ caractère$
$lire\_pre : transition -> condition$
$lire\_post : transition -> condition$

Type jeton
utilise : $chaîne\ de\ caractère, entier$
opérations :
$créer\_jeton : entier \times chaîne\ de\ caractère \times entier -> jeton$
$lire\_nombre : jeton -> entier$
$lire\_nomjeton : jeton -> chaîne\ de\ caractère$
$lire\_identificateur : jeton -> entier$

Type bound
utilise : $entier, booléen, jeton, ensemble$
opérations :
$créer\_bound : \quad -> bound$
$ajouter\_jeton : bound \times jeton -> bound$
$vide : bound -> booléen$
$lire\_evt\_bound : bound -> ensemble$
$nettoyer\_bound : bound \times ensemble -> bound$

Type place
utilise : $entier, jeton, bound, ensemble$
opérations :
$créer\_place : \quad -> place$
$set\_free : place \times entier -> place$
$set\_bound : place \times jeton -> place$
$lire\_free : place -> entier$
$lire\_bound : place -> bound$

$lire\_evt\_place : place- > ensemble$
$nettoyer\_place : place \times ensemble- > place$

Type configuration
utilise : $place, transition, ensemble, booléen$
opérations :
$créer\_configuration : \ - > configuration$
$ajouter\_place : configuration \times place- > configuration$
$init : configuration \times condition- > configuration$
$lire\_evt\_configuration : configuration- > ensemble$
$nettoyer\_configuration : configuration \times ensemble- > configuration$
$enabled : configuration \times transition- > booléen$
$consomme : configuration \times tranition- > configuration$
$franchissement : configuration \times transition- > configuration$

Type edge
utilise : $ensemble, chaîne\ de\ caractère, entier$
opérations :
$créer\_edge : ensemble \times chaîne\ de\ caractère \times entier- > edge$
$lire\_cause : edge- > ensemble$
$lire\_action : edge- > chaîne\ de\ caractère$
$lire\_ident : edge- > entier$

Téchniques utilisées pour paramétrer l'outil :

Le type abstraît de donnée *ensemble* est utilisé pour gérer les noms des évènements. Cependant pour paramétrer l'outil nous avons développé un type abstraît de donnée $multi - ensemble$ et nous avons enrichi ce dernier par plusieurs opérations, parmi ces opérations on a : $conflit$ et $conflit\_red$. La première est utilisée pour calculer l'ensemble Min dans le cas de génération d'un STEM. Quand à la deuxième, elle est utilisée pour calculer l'ensemble Min dans le cas de génération d'un STEM t-réduit.

En premier lieu, l'ensemble des évènements des configurations est lu, par la suite le calcul de la partition de cet ensemble est effectué, on obtient comme résultat un multi-ensemble. Pour chaque ensemble de ce dernier, on vérifie s'il satisfait la précondition de franchissement de transition et on calcule l'ensemble de conflit pour le multi-ensemble résultant.

Algorithmes :

Algorithme 5.1 *: chargement de donnée*
Variables :
liste_ tokens : liste de jetons lexicales ;
T : liste de transitions ;

nomtr : chaîne de caractère ;
Item : item ;
pre_ens, post_ens, Initialisation : condition ;
Début :
liste_tokens <- jetons lexicales / les jetons lexicales sont produit par le module compil */*
Tantque *liste_tokens non vide* ***Faire***
sélectionner et supprimer tête liste_tokens ;
Si *tête liste_tokens ="tr"* ***alors***
sélectionner et supprimer tête liste_tokens ;
nomtr <- tête liste_tokens ;
pre_ens <- créer_condition() ;
sélectionner et supprimer tête liste_tokens ;
Tantque *tête liste_tokens <> "->"* ***Faire***
Item <- créer_item(traiter(tête liste_tokens)) ;
pre_ens <- ajouter_item(pre_ens,Item) ;
sélectionner et supprimer tête liste_tokens ;
Fin Tanque
post_ens <- créer_condition() ;
Tant que *((liste_tokens non vide) et ((tête liste_tokens<>"tr") et (tête liste_tokens<>"pl")))* ***Faire***
Item <- créer_item(traiter(tête liste_tokens)) ;
post_ens <- ajouter_item(post_ens,Item) ;
sélectionner et supprimer tête liste_tokens ;
Fin Tantque
T <-ajouter(T,créer_transition(nomtr,pre_ens,post_ens)) ;
Sinon
Si *tête liste_tokens="pl"* ***alors***
sélectionner et supprimer tête liste_tokens ;
Tant que *(liste_tokens non vide)* ***Faire***
Item <- créer_item(traiter(tête liste_tokens)) ;
Initialiation <- ajouter_item(Initialisation,Item) ;
sélectionner et supprimer tête liste_tokens ;
Fin Tantque
Fin si
Fin si
Fin Tantque
Fin Algo

Algorithme 5.2 *genere*
Variables :
initial,conf, conf_new : configuration ;
Initialisation : condition ;

t : transition ;
tableau : liste de configuration ;
min : multi-ensemble ;
elt : ensemble ;
STEM_ t_ réduit :booléen ;
Début :
initial <- créer_ configuration() ;
initial <- init(initial,Initialiation) ;
Tantque ∃ *configuration non traiter* ***Faire***
sélectionner une configuration ***conf*** *de* ***tableau*** *;*
Pour chaque *transition* ***t Faire***
Si *enabled(conf,t)* ***Faire***
Si *STEM_ t_ réduit* ***Faire***
*Min <- conflit_ red(**filtrer**(partition(lire_ evt_ configuration(conf)))) ;*
Sinon */* STEM_ classique*/*
*Min <- conflit(**filtrer**(partition(lire_ evt_ configuration(conf)))) ;*
FinSi
Pour chaque *elément* ***elt*** *de Min* ***Faire***
edge_ new <-créer_ edge(elt,lire_ nom_ transition(t),get($\mathcal{M}$*)) ;*
conf_ new <- franchissement(consomme((nettoyer_ configuration(conf,elt)),t)) ;
ajouter conf_ new au tableau ;
ajouter edge_ new de conf à conf_ new ;
Fin Pour chaque
FinSi
Fin Pour chaque
Fin Tantque
Fin Algo

5.4.3 Codage :

Dans cette section, nous donnons quelques lignes de code en ocaml, ainsi que quelques statistiques sur le code source. A titre d'exemple nous présentons les déclarations des types et l'implémentation du type abtraît de donnée "ensemble".

Déclarations :

type item= {nomplace :string ;nbre :int} and condition= item list and

transition={name :string ;pre :condition ;post :condition} and

jeton={nb :int ;nomjeton :string ;identificateur :int} and bound= jeton list and

place={fr :int ;lt : bound} and configuration=place list and edge={cause :int list ;act :string ;ident :int}

implémentation de TAD ensemble :

let créer_ensemble()=[]

let vide(e)= if e=[] then true else false

let rec ajouter(ens,e)=match ens with

```
[]->[e]
|t : :q->t : :ajouter(q,e)
let rec member(e,ens)= match ens with
[]->false
|t : :q-> if t=e then true else member(e,q)
let rec union(ens1,ens2) = match ens1, ens2 with
[],ens2->ens2
|ens1,[]->ens1
|t : :q,ens2->if member(t,ens2) then union(q,ens2) else t : :union(q,ens2)
let rec intersection(ens1,ens2)= match ens1, ens2 with
[],ens2->[]
|ens1,[]->[]
|ens1,t : :q->if member(t,ens1) then t : :intersection(ens1,q) else intersection(ens1,q)
let rec inclus(ens1,ens2)= match ens1, ens2 with
[],ens2->false
|ens1,[]->true
|ens1,t : :q->if member(t,ens1) then inclus(ens1,q) else false
let rec difference= function
([],_)->[]
|(t : :q,ens2)->if member(t,ens2) then difference(q,ens2) else t : :difference(q,ens2)
```

Quelques statistiques sur le code source :

Nom module	Nombre de lignes
Compil	30
analyse_syn	44
genere	64
genere_dot	25
genere_stem	22
implémentation des TADs	233
main	6
Total	424

TAB 5.2- Quelques statistiques sur le code source.

5.5 Conclusion :

Dans ce chapitre, nous avons concrétisé les études théoriques du chapitre 4, à travers le développement de l'outil «MSOS-PN». Ce dernier a été intégré à l'environnement de vérification formelle FOCOVE, il prend en entrée une spécification d'un réseau de Petri généré via l'outil Tina et produit en sortie, selon le choix, soit un STEM classique, soit un STEM-t-réduit. Le STEM généré peut être visualisé en utilisant l'outil dotty ou l'outil STEM-Viewer.

La première partie de ce chapitre a présenté l'architecture globale de l'environnement de vérification formelle FOCOVE. Dans la deuxième partie nous avons

présenté les différents outils nécessaires au développement du logiciel. Par ailleurs, la dernière partie de ce chapitre a été consacrée à la présentation des différentes étapes de processus de développement : afin de maîtriser la complexité du logiciel nous avons adopté une technique de conception hiérarchique dont la décomposition a été faite selon les fonctionnalités. Par la suite, nous avons donné la conception détaillée de chaque composant et quelques lignes du code en objective caml. Il est à noter que nous avons utilisé ce langage vu son efficacité dans le développement de logiciel du genre compilateur.

Chapitre 6

Conclusion et Perspectives

Ce mémoire se situe dans le cadre de la vérification comportementale des réseaux de Petri. L'une des approches de la vérification des réseaux de Petri consiste à générer son graphe de marquage, ce dernier peut être vu comme étant un système de transitions étiquetées. Bien que la sémantique d'entrelacement soit souvent considérée comme étant la plus simple pour exprimer le parallèlisme, elle ne permet pas de représenter correctement le comportement des systèmes concurrents, dès que les actions ne sont plus atomiques. La non atomicité des actions implique l'utilisation des sémantiques, dites de vrai parallèlisme, tel que la sémantique de maximalité.

Nous avons proposé une méthode opérationnelle de génération de systèmes de transitions étiquetées maximales pour les réseaux de Petri. Donc la vérification des propriétés de bon fonctionnement d'un système revient en la vérification de ces propriétés sur le système de transitions étiquetées maximales généré. Il est à noter que la méthode proposée est valable pour tout réseau de Petri borné.

D'autre part, ce mémoire a abordé un problème crucial, à savoir le problème de l'explosion combinatoire du graphe d'état. Pour réduire les effets de ce problème nous avons proposé une méthode de réduction à la volée des systèmes de transitions étiquetées maximales. Cette méthode consiste en l'agrégation des transitions redondantes modulo la relation de bissimulation de maximalité définie sur l'ensemble des transitions du réseau de Petri. L'intérêt de l'approche proposée réside dont le fait qu'elle réduit significativement la taille du système de transitions étiquetées maximales tout en préservant les propriétés du système à vérifier.

Afin de concrétiser les études théoriques, nous avons développé l'outil MSOS-PN (Maximality-based Structured Operational Sémantics for Petri Net), en utilisant le langage objective caml. Cet outil permet de générer, selon le choix, soit un système de transitions étiquetées maximales, soit un système de transitions étiquetées maaximales t-réduit. Par ailleurs nous avons intégré cet outil dans l'environnement de vérification formelle FoCoVE.

Perspectives

Comme perpectives de notre travail nous proposons le développemnt des points suivants :

- Proposition d'un algorithme de réduction à la volée des systèmes de transitions étiquetées maximales associés aux réseaux de Petri, modulo la α-équivalence.
- Propotion d'une méthode de génération des systèmes des transitions étiquetées maximales pour les réseaux de Petri non bornés.
- Extension de l'approche proposée aux réseaux de Petri temporisés par la considération du modèle sémantique des DATAs (Durational Action Timed Automata).

Bibliographie

[AH91] L. Aceto and M. Hennessy. Adding action refinement to finite process algebra. In J. L. Albert, B. Monien, and M. R. Artalejo, editors, "ICALP'91", volume 510 of "LNCS", pages 506–519. Springer-Verlag (1991).

[Arn92] A. Arnold. "Systèmes de transitions finis et sémantique des processus communicants". Masson, Paris (1992).

[BB87] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems* **14**, 25–59 (1987).

[BB01] A. Benamira and A. Bouziane. "Conception et Implémentation D'un Environement de Vérification Pour Basic CCS: Approche d'Entrelacement." Université Mentouri- Constantine. (2001.) mémoire d'ingénieur.

[BBB01] S. Boulkera., T. Bourekab, and C. Berdoudi. "Conception et Implémentation D'un Environement de Vérification Pour Basic LOTOS: Approche d'Entrelacement." Université Mentouri Constantine. (2001). mémoire d'ingénieur.

[BC88] G. Boudol and I. Castellani. Concurrency and atomicity. *TCS* **59**, 1–60 (1988).

[BD87] E. Best and R. Devillers. Sequential and concurrent behaviour in petri net theory. *Theoretical Computer Science* **55**, 87–136 (1987).

[BDKP91] E. Best, R. Devillers, A. Kiehn, and L. Pomello. Concurrent bisimulations in Petri nets. *Acta Informatica* **28**, 231–264 (1991).

[BDS02] N. Belala, A. Delimi, and M. N. Seghir. "Développement D'un Environement de Vérification Formel Basé sur la Sémantique de Maximalité: Approche Logique." Université de Constantine. (2002). mémoire d'ingénieur.

[Bel02] M. Belguidoum. "Conception et Implémentation D'un Outil Pour la Vérification Formelle de Relation de Bissimulation Entrelacée." Université de constantine. (2002). mémoire d'ingénieur.

[BK85] J. A. Bergstra and J. W. Klop. Algebra of communicating processes with abstraction. *TCS* **37**, 77–121 (1985).

[BS06] A. Benamira and D. E. Saidouni. Consideration of the covering steps in the maximality-based labled transition systems. (Dec 2006).

[CC91] J. P. Courtiat and R. J. Coelho da Costa. A LOTOS based calculus with true concurrency semantics. In G. A. Rose and K. R. Parker, editors, "Formal Description Techniques (FORTE'91)". North-Holland (1991).

[CC92] R. J. Coelho da Costa and J. P. Courtiat. Using Petri nets as a model for Petri nets. In "Third IEEE Worshop on Future Trends of Distributed Computing Systems". IEEE Computer Society Press (1992).

[CC93] R. J. Coelho da Costa and J. P. Courtiat. A true concurrency semantics for LOTOS. In "Formal Description Techniques (FORTE'92)". North-Holland (1993).

[CCI88] CCITT88. "SDL. Recommendation Z.100-Z.104." CCITT (1988).

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems* **8**(2), 244–263 (April 1986).

[CH93] R. Cleveland and M. Hennessy. Testing equivalence as a bisimulation equivalence. *Formal Aspects of Computing* **5**, 1–20 (1993).

[Coe93] R. J. Coelho da Costa. "Systèmes de Transitions Etiquetés Causaux: Une Nouvelle Approche pour la Description du Comportement Evénementiel de Systèmes Concurrents". PhD thesis, LAAS-CNRS, 7 av. du Colonel Roche, 31077 Toulouse Cdex France (1993).

[CS92] J. P. Courtiat and D. E. Saïdouni. Action refinement in LOTOS (extended version). Research Report 92477, LAAS-CNRS, 7 av. du Colonel Roche, 31077 Toulouse Cedex France (December 1992).

[CS94] J. P. Courtiat and D. E. Saidouni. Protocol specification, testing and verification. pages 341–354, North-Holland. (1994).

[CS95] J. P. Courtiat and D. E. Saïdouni. Relating maximality-based semantics to action refinement in process algebras. In D. Hogrefe and S. Leue, editors, "IFIP TC6/WG6.1, 7th Int. Conf. on Formal Description Techniques (FORTE'94)", pages 293–308. Chapman & Hall (1995).

[DD89] P. Darondeau and P. Degano. Causal trees. In "ICALP'89", volume 372 of "LNCS", pages 234–248. Springer-Verlag (1989).

[DD91] P. Darondeau and P. Degano. About semantic action refinement. *Fundamenta Informaticae* **14**, 221–234 (1991).

[DD93] P. Darondeau and P. Degano. Refinement of actions in event structures and causal trees. *TCS* **118**, 21–48 (1993).

[Dev92a] R. Devillers. Maximality preservation and the ST-idea for action refinement. In G. Rozenberg, editor, "Advances in Petri Nets", volume 609 of "LNCS", pages 108–151. Springer-Verlag (1992).

[Dev92b] R. Devillers. Maximality preserving bisimulation. *TCS* **102**(1), 165–184 (August 1992).

[Dev93] R. Devillers. Construction of S-invariants and S-components for refined Petri boxes. In M. A. Marsan, editor, "ATPN'93", volume 691 of "LNCS", pages 242–261. Springer-Verlag (1993).

[DG91] P. Degano and R. Gorrieri. Atomic refinement in process description languages. In A. Tarlecki, editor, "Mathematical Foundations of Computer Science", volume 520 of "LNCS", pages 121–130. Springer-Verlag (1991).

[DG02] L. Deradj and A. Ghenai. "Conception et Implémentation D'un Outil Pour la Vérification Formelle Selon L'approche Test Basée sur la Sémantique D'entrelacement." Université de constantine. (2002.) mémoire dingénieur.

[Dij71] E. W. Dijkstra. Hierarchical ordering of sequential processes. *Acta Informatica* **1**(2), 115–138 (1971).

[DLW94] A. Danthine, G. Leduc, and P. Wolpe, editors. "Action Refinement in LOTOS". North-Holland (1994).

[Eme90] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, "Handbook of Theoretical Computer Science : Formal Models and Semantics, Ch. 16", volume B, pages 995–1072. Elsevier (1990).

[FM] J. C. Fernandez and L. Mounier. A tool set for deciding behavioral equivalences. pages 23–42.

[Hoa85] C. A. R. Hoare. "Communicating Sequential Processes". Prentice-Hall (1985).

[ISO88] ISO/IEC. "LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, International Standard 8807". International Organisation of Standardization – Information Processing Systems – Open Systems Interconnection, Genève (September 1988).

[JM81] R. Johnsonbaugh and T. Murata. Additional methods for reduction an expansion of marked graphs. *IEEE* **CAS-28**(10), 1009–1014 (Oct 1981).

[JPZ91] W. Janssen, M. Poel, and J. Zwiers. Action systems and action refinement in the development of parallel systems. In J. C. M. Baeten and J. F. Groote, editors, "CONCUR'91", volume 527 of "LNCS", pages 298–316. Springer-Verlag (1991).

[MF76] P. M. Merlin and D. J. Farber. Recoverability of communication protocols: Implications of theoretical study. (Sept 1976).

[Mil80] R. Milner. "Communication and Concurrency", volume 92 of "LNCS". Springer-Verlag (1980).

[Mil83] R. Milner. Calculus for synchrony and asynchrony. *TCS* **25**, 267–310 (1983).

[Mil89] R. Milner. "Communication and Concurrency". Prentice Hall (1989).

[Mur89] T. Murata. Petri nets: properties, analysis and applications. In "proc. IEEE", volume 77. IEEE Press (1989).

[Pet78] C. A. Petri. Concurrency as a basis of systems thinking. *Interner Berchit ISF-78-06* (Sept 1978).

[Rei85] W. Reisig. "Petri nets", volume 4 of "EATCS Monographs on Theoretical Computer Science". Springer-Verlag, Berlin, Heidelberg, New York, Tokyo (1985).

[Saï96] D. E. Saïdouni. "Sémantique de Maximalité: Application au Raffinement d'Actions en LOTOS". PhD thesis, LAAS-CNRS, 7 av. du Colonel Roche, 31077 Toulouse Cedex France (1996).

[Sai05] D. E. Saidouni. "Modèles Du Parallélisme". (2005).

[SB05] D. E. Saïdouni and N. Belala. Using maximality-based labeled transition system model for concurrency logic verification. *The International Arab Journal of Information Technology (IAJIT)* **2**(3), 199–205 (July 2005). ISSN: 1683-3198.

[SB06] G. Scorletti and G. Binet. "Réseaux de Petri". Cours EL401T2. Université de CAEN/BASSE- NORMANDIE. U.F.R : SCIENCE CAEN. (2006).

[SBB08a] D. E. Saidouni, N. Belala, and M. Bouneb. Aggregation of transitions in marking graph generation based on maximality semantics for petri net. (2008). Verification and Evalution of Computer and Communication Systems (VECOS'2008).

[SBB08b] D. E. Saidouni, N. Belala, and M. Bouneb. Réseau de petri et sémantique de maximalité. Séminaire National d'Informatique Biskra (SNIB'2008). (2008).

[SBB08c] D. E. Saidouni, N. Belala, and M. Bouneb. Using maximality-based labeled transition system as model for petri nets. (2008). The international Arab Conference on Information Technology (ACIT'2008).

[SC] D. E. Saidouni and J. P. Courtiat. Sémantique de maximalité.

[SC94] D. E. Saïdouni and J. P. Courtiat. Syntactic action refinement in presence of multiway synchronization. In D.J. Andrews, J.F. Groote, and C.A. Middelburg, editors, "Workshop on Semantics of Specification Languages (SoSL'93)", Workshops in Computing, pages 289–303, Utrecht (1994). Springer-Verlag.

[SC03] D. E. Saïdouni and J. P. Courtiat. Prise en compte des durées d'action dans les algèbres de processus par l'utilisation de la sémantique de maximalité. In "Ingénierie Des Protocoles (CFIP'2003)". Hermes, France (2003).

[SIL85] "Petri Nets in Automation and Computer Engineering.", Madrid Spain (1985). English translation published by Kluwer Academic 1990.

[SL03] D. E. Saïdouni and Ouassila Labbani. Maximality-based symbolic model checking. In "ACS/IEEE International Conference on Computer Systems and Applications", Tunisia (July 2003).

[van90a] R. J. van Glabbeek. "Comparative Concurrency Semantics and Refinement of Actions". PhD thesis, Vrije Universiteit Te Amsterdam (1990).

[van90b] R. J. van Glabbeek. The refinement theorem for ST-bisimulation semantics. In "IFIP Working Conference on Programming Concepts and Methods". North-Holland (1990).

[VNCG92] Vidal, Naquet, Choquet, and Geniet. "Réseaux de Petri et Systèmes Parallèles." (1992).

[Vog91] W. Vogler. Bisimulation and action refinement. In "STACS'91", volume 480 of "LNCS", pages 309–321. Springer-Verlag (1991).

[Vog93] W. Vogler. Bisimulation and action refinement. *TCS* **114**, 173–200 (1993).

[vV87] R. J. van Glabbeek and F. W. Vaandrager. Petri net models for algebraic theories of concurrency. In "PARLE'87, Vol. II (Parallel Languages)", volume 259 of "LNCS", pages 224–242. Springer-Verlag (1987).