Réseaux d'Automates Stochastiques : Génération de l'espace d'états atteignables et Multiplication vecteur-descripteur pour une sémantique en temps discret

Afonso Henrique Correa de Sales

INSTITUT POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque
|_|_|_|_|_|_|_|_|_|_|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'Institut Polytechnique de Grenoble

Spécialité : **"Informatique"**

préparée au Laboratoire d'Informatique de Grenoble dans le cadre de
l'Ecole Doctorale "Mathématiques, Sciences et Technologies de l'Information, Informatique"

présentée et soutenue publiquement

par

Afonso Henrique CORRÊA DE SALES

le 10 Septembre 2009

Réseaux d'Automates Stochastiques : Génération de l'espace d'états atteignables et Multiplication vecteur-descripteur pour une sémantique en temps discret

---

Directeur de thèse : Brigitte Plateau

JURY

ROLAND GROZ	Grenoble INP	Président
JEAN-MICHEL FOURNEAU	Université de Versailles Saint Quentin	Rapporteur
WILLIAM J. STEWART	North Carolina State University	Rapporteur
BRUNO GAUJAL	INRIA	Invité
BRIGITTE PLATEAU	Grenoble INP	Directeur de thèse

À Carol, mon amour éternel.

Nous sommes le reflet de nos actions, de nos pensées et de nos intentions passées.

Remerciements

La thèse est sûrement la tâche la plus difficile que j'ai déjà réalisée. C'est une tâche qui n'est pas réalisable sans le soutien des amis et de la famille. C'est la fin d'une étape qui a commencé il y a très longtemps. En fait, ce n'est pas vraiment la fin... mais le début d'une prochaine étape.

J'aimerais tout d'abord remercier les membres de mon jury, Roland Groz, Jean-Michel Fourneau, Billy Stewart et Bruno Gaujal pour avoir accepté de s'intéresser à mon travail, je les remercie aussi pour leurs remarques, commentaires et conseils.

Un grand merci à Brigitte Plateau d'avoir accepté de m'encadrer pendant ces quatre années de thèse. Merci pour tous ses conseils et ses enseignements remarquables pour ma vie professionnelle et personnelle.

Je ne pourrais jamais oublier de remercier Paulo Fernandes. Une grande partie de mon projet de vie a été réalisée avec son soutien et ses conseils.

J'aimerais aussi remercier tous les amis qui m'ont permis un séjour exceptionnel à Grenoble. Je tiens surtout à citer mes amis César et Beatriz Gonçalves, Mateus et Janaina Cardoso, Pedro Velho, Patricia Scheeren, Lucas Schnorr, Fabiane Basso, Bringel Filho, Suely Ribeiro Gonçalves, Everton et Ana Paula Hermann, Rodrigo et Christiane Ribeiro, Daniel Cordeiro, Kelly Braghetto, Marcio Castro, Bruno Donassolo, Marcia Cera, Hyane Trigueiro, Thais Webber, Ricardo Czekster, Edson Moreno, Marcia Pasin, Tiago Camargo, Michelle Leonhardt, Allen Medeiros, Max et Geni Reynier. Et j'ai certainement oublié de citer certains d'entre vous qui ont fait partie de ma vie pendant cette fantastique période. Merci à tous.

Merci à tous les membres du laboratoire LIG qui m'ont accueilli et m'ont aidé pendant ma thèse.

Je ne pourrais pas oublier de citer mes amis et collègues de bureau Ihab Sbeity et, spécialement, Daouda Traoré. Ils ont toujours été disponibles pour m'aider, surtout, pour les questions de la langue française.

Je tiens également à remercier Leonardo Brenner et Marina Savoldi, ça a été un grand plaisir de passer ces quatre années avec vous à Grenoble.

J'aimerais remercier tous les amis qui sont restés au Brésil et m'ont soutenu moralement pendant ces années de thèse.

J'aimerais aussi remercier toute ma famille pour leur soutien inconditionnel. Je remercie ma sœur et mon beau-frère, mon père et ma mère d'avoir compris mon absence pendant ces années de thèse.

Finalement, je tiens à remercier plus spécialement la personne la plus importante de ma vie, Carol. Plus que mon épouse, mon amie et ma copine pour tous les moments. Son amour et son soutien m'ont permis de terminer cette thèse. Elle sait bien que cette thèse est la sienne aussi. Merci pour tout, mon amour !

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Ces dernières années, la complexité des systèmes et réseaux informatiques ne fait qu'augmenter et de façon extrêmement rapide. Cette complexité est liée à la taille, la fiabilité, la performance et la capacité du système à s'adapter à différentes conditions d'exécution. La conception des systèmes informatiques ne peut pas être seulement fondée sur l'expérience (et parfois, l'intuition) du concepteur de ces systèmes. De plus en plus, des outils sophistiqués sont nécessaires pour la planification et gestion de ces systèmes.

Afin de prévoir les problèmes de performance des systèmes, avant même l'étape de prototypage et de test, l'utilisation des *méthodes de modélisation* et de *prédiction de performances* deviennent fondamentales. Nous allons nommer les étapes de modélisation et de prédiction de performances par le terme générique d'*évaluation de performance*.

Tout au long du *cycle de vie* du système (*i.e.*, de la spécification à l'exploitation), l'évaluation de performance est essentielle pour comprendre le comportement dynamique du système et répondre à des questions de coût et de performance.

Pour chaque étape du cycle de vie, l'utilisation de certaines méthodes sont plus efficaces que d'autres : les formalismes de modélisation et les méthodes de simulation [120] sont utilisés au début du cycle de vie du système (*i.e.*, dans la phase de spécification du système) ; l'observation (monitoring) n'est appliquée qu'à la fin du cycle de vie (*i.e.*, au moment où le système est déjà implanté) et donc les modifications nécessaires au système peuvent, parfois, amener à des changements coûteux.

Nous nous sommes intéressés, dans cette thèse, à l'évaluation de performance pour la première phase du cycle de vie du système (i.e., l'abstraction du système par un modèle). Nous nous intéressons plus particulièrement à la conception des systèmes informatiques parallèles et distribués.

Des systèmes informatiques parallèles et distribués sont naturellement grands et trop complexes pour être traités par des méthodes traditionnelles[1] d'évaluation de performance. La modélisation d'un système complexe en utilisant les chaînes de Markov [126] peut parfois rendre impraticable le calcul de la solution du modèle à cause de la complexité de sa description (*e.g.*, un système avec des centaines de millions d'états). L'utilisation d'un formalisme de haut-niveau devient alors impératif pour que nous puissions appréhender toutes les caractéristiques importantes du système étudié.

[1] Une reprise des méthodes traditionnelles d'évaluation de performance connues dans la littérature sera présentée dans la section suivante.

Les systèmes parallèles et distribués sont fréquemment composés par des modules et des interactions entre ces modules sont assez rares. Ces interactions se font par des mécanismes comme la synchronisation. Plateau a proposé le formalisme de modélisation des *Réseaux d'Automates Stochastiques* (SAN - *Stochastic Automata Networks*) [110, 4] qui bien décrit ce type de comportement des systèmes parallèles et distribués. L'idée principale du formalisme SAN est de modéliser un système en plusieurs sous-systèmes, *i.e.*, un système composé de *modules*. Le formalisme SAN représente de plus la matrice de transition de la chaîne de Markov équivalente par une formule tensorielle [42] (appelée *descripteur*) qui réduit énormément les besoins de stockage de cette matrice.

Les travaux de cette thèse s'adressent à l'évaluation de performance de systèmes complexes représentés par des modèles compositionnels avec de très grands espaces d'états qui sont décrits par des formalismes de haut-niveau.

Lorsque nous parlons des systèmes modélisés par des formalismes de haut-niveau, le principal problème qui surgit est l'*explosion combinatoire de l'espace d'états* du modèle. La génération de l'espace d'états est le principal défi pour la grande majorité des outils de vérification formelle (par exemple, le vérificateur de modèles [37]). Des modèles décrits par des formalismes de haut-niveau permettent automatiquement la vérification et l'évaluation de performance de systèmes dont l'analyse, autrement, serait impossible.

Certains types d'analyse sont basés sur la vérification *systématique* des états d'un modèle décrit par un formalisme de haut-niveau [28, 84, 33, 73]. Toutefois, des techniques performantes pour la génération symbolique de l'espace d'états, qui n'exigent pas la vérification *explicite* de chaque état du modèle, peuvent être employées pour les modèles compositionnels [36, 25, 105, 102, 103, 94, 104]. Ces techniques permettent de générer l'espace d'états de modèles compositionnels avec des millions ou même billions d'états.

L'objectif de cette thèse consiste à proposer des méthodes pour la génération de l'espace d'états atteignables de modèles compositionnels à temps continu qui utilisent des taux fonctionnels.

Dans les systèmes modélisés sur une *échelle de temps continue*, un *seul* événement peut avoir lieu à chaque instant de temps car la loi d'occurrence d'un événement est un variable aléatoire indépendante suivant une distribution exponentielle. Dans les systèmes modélisés sur une *échelle de temps discrète*, à chaque intervalle de temps, *plusieurs* événements peuvent avoir lieu dans le même instant : la loi d'occurrence d'un événement est une variable aléatoire indépendante suivant une distribution géométrique. La possibilité d'avoir des événements simultanés rend la modélisation de systèmes en temps discret plus complexe que la modélisation de systèmes en temps continu.

Un deuxième objectif de cette thèse est de proposer, à partir d'une algèbre sur les événements et d'une algèbre sur des matrices de transition "locale", une méthode numérique pour la résolution d'un modèle à temps discret dont la matrice de transition de la chaîne de Markov est représentée sous un format tensoriel.

Nous présentons, dans la section suivante, une synthèse des méthodes d'évaluation de performance connues dans la littérature. Ensuite, nous présentons les objectifs de cette thèse en soulignant ses apports par rapport aux travaux de la littérature.

1.1 Méthodes d'évaluation de performances

Nous pouvons séparer l'évaluation de performances en deux étapes distinctes : la *modélisation* du système et la *résolution* du modèle.

L'étape de modélisation du système est basée sur le développement d'une description formelle du système et l'étape de résolution du modèle consiste calculer à partir de cette description pour l'obtention des prédictions de performances du système.

Dans la section suivante, nous allons présenter les formalismes de modélisation et, dans la section 1.1.2, nous allons présenter les méthodes de résolution.

1.1.1 Formalismes de modélisation

Les recherches destinées à la modélisation et prédiction de performances ont toujours été très liées. Les travaux les plus anciens viennent de l'analyse des processus stochastiques [128] et plus généralement des chaînes de Markov [52, 126]. Le formalisme des chaînes de Markov est un des formalismes de modélisation le plus ancien et aussi un des plus employés dans l'évaluation de performance.

De manière à répondre à la complexité des nouvelles générations de systèmes, plusieurs techniques de modélisation ont été développées. Ces techniques sont fondées sur le même principe, consistant à définir de façon successive :

- les *états*[2] du système ;
- les *transitions* entre les états, *i.e.*, la dynamique du système ;
- la *temporisation* des transitions.

En se basant sur ce principe, les modèles permettent d'analyser le comportement dynamique d'un système en déterminant l'ensemble de toutes les transitions possibles entre les différents états de ce système. Ces systèmes modélisés s'exécutent dans un environnement aléatoire. Le formalisme des chaînes de Markov, que ce soit à temps continu ou à temps discret, facilite l'analyse des performances des systèmes dynamiques dans de nombreux domaines d'application [67, 89, 126, 51, 61, 101]. Notamment, il est aussi bien adapté à l'étude des systèmes parallèles que distribués [125, 130, 3, 65].

Néanmoins, lorsque le nombre d'états d'un modèle est important (de l'ordre de millions d'états), il est pratiquement impossible d'envisager une modélisation directe sous la forme *état-transition*. La taille de l'espace d'états d'un modèle est parfois si grande qu'il est impossible d'obtenir de façon exacte des indices de performances de ce modèle. D'ailleurs, la représentation de la matrice de transition de la chaîne de Markov de ce modèle devient aussi impossible.

Afin de répondre à cette limitation des chaînes de Markov, plusieurs techniques et formalismes de modélisation de haut-niveau ont été développés au cours des dernières années. La forme structurée de ces formalismes de haut-niveau permet de prendre en compte l'aspect compositionnel et hiérarchique des systèmes.

[2] Dans cette thèse, on se restreint à des systèmes avec un nombre fini d'états.

Vers la fin des années 50, Jackson a proposé une nouvelle approche pour les *Réseaux de Files d'Attente* [77, 78]. Ce formalisme est devenu très populaire à la fin des années 80, notamment avec les travaux de Little [88], Baskett, Chandy, Muntz et Palacios [6], et Reiser et Lavenberg [116].

Le domaine d'application des réseaux de files d'attente classiques est toutefois assez limité. Pour traiter de problèmes plus complexes (*e.g.*, des systèmes avec synchronisation), des approches alternatives ont souvent abouti à des extensions du formalisme des réseaux de files d'attente, comme c'est le cas des travaux sur :

- les *méthodes d'approximation* de Chandy et Sauer [26] et de Courtois [38] ;
- les *réseaux à capacité limitée* de Dallery (*fork-join*) [40, 41] ;
- les *réseaux avec clients négatifs* (réseaux généralisés) [66] ;
- les *réseaux avec contrôle de décision* [13, 80] ;
- les *réseaux hiérarchiques* [20].

Ces extensions des réseaux de files d'attente ont fourni des outils puissants pour la modélisation de systèmes d'attente. Néanmoins, le domaine d'application de ces extensions continue à être relativement restreint à des systèmes d'attente.

D'autres approches ont abandonné le formalisme de réseaux de files d'attente. Ainsi, le formalisme de *Réseaux de Petri* [108, 109, 107, 14, 117] est devenu populaire au début des années 80 et d'autres approches ont adopté les réseaux de Petri comme formalisme de base en ajoutant des extensions qui vont de simples temporisations constantes [118, 127] jusqu'à des mécanismes beaucoup plus sophistiqués, comme par exemple :

- les *réseaux de Petri stochastiques* [58] ;
- les *réseaux de Petri colorés* [71, 72, 79] ;
- les *réseaux de Petri stochastiques généralisés* [2, 1] ;
- les *réseaux de Petri stochastiques généralisés superposés* [48, 49].

Le formalisme des *Algèbres de Processus Stochastiques* permet de modéliser des délais dans les algèbres de processus, et il est donc particulièrement bien adapté à l'évaluation des performances. Plusieurs variantes existent aussi pour les algèbres de processus stochastiques tels que PEPA [74, 75], EMPA [11] et TIPP [69].

Enfin, le formalisme des *Réseaux d'Automates Stochastiques* (SAN - *Stochastique Automata Networks*) [110, 4, 54, 8, 124, 15] permet à la fois de structurer l'approche par composants (automates) et de modéliser les délais.

La plupart des travaux en évaluation de performance utilisent des chaînes de Markov à temps continu et notamment les formalismes de haut-niveau cités précédemment en font partie. Les hypothèses probabilistes de systèmes modélisés à temps continu (durée ayant une distribution exponentielle, donc non

bornée et sans mémoire, et *aucune simultanéité d'événements*) rendent relativement aisée la génération de la matrice de transition de la chaîne de Markov sous-jacente.

Il n'en est pas moins vrai que l'intérêt d'étudier un système en temps discret est réel : lorsque la réalité est perçue lors d'intervalles de temps discrétisé, les modèles en temps discret permettent de modéliser des durées fixes et bornées et enfin la *simultanéité d'événements* est possible. Différemment des modèles en temps continu, la grande difficulté pour les modèles en temps discret est le calcul effectif de la formulation matricielle de la chaîne de Markov sous-jacente. Cette difficulté vient essentiellement de la combinatoire générée par la possibilité d'avoir des événements simultanés. Cette combinatoire est particulièrement importante dans le cas de modèles à base de composants (*e.g.*, les modèles décrits par des formalismes structurés), dédiés à l'étude des systèmes parallèles et distribués.

Nous nous intéressons, dans le cadre de cette thèse, plus spécifiquement au formalisme des ***Réseaux d'Automates Stochastiques*** *(SAN).*

L'utilisation du formalisme SAN est devenue populaire dans le domaine de la modélisation stochastique de systèmes informatiques parallèles et distribués telles que *processeurs concurrents*, la *mémoire partagée*, la *communication inter-processus*, les *protocoles de communication* et d'autres dans le domaine d'application des chaînes de Markov [96, 53, 5, 12, 50, 27, 45, 16].

1.1.2 Méthodes de résolution

Nous nous sommes intéressés à calculer des indices de performance du système modélisé dans le cadre Markovien, indépendamment du formalisme de modélisation utilisé. Dans ce but, plusieurs méthodes de résolution sont proposées dans la littérature. Certains méthodes sont plus spécifiques à un formalisme donné que d'autres, cependant des techniques peuvent être adaptées d'un formalisme à un autre.

Parmi ces méthodes de résolution, nous pouvons faire la distinction entre :

- les *méthodes analytiques* [6, 85, 116] ;
- les *méthodes numériques* [52, 122, 126] ;
- les *simulations* [119, 81, 134].

Les méthodes de simulations ne serons pas étudiées dans le cadre de cette thèse.

Les méthodes analytiques sont les méthodes qui donnent une solution sans passer par la résolution numérique du système linéaire $\pi Q = 0$, où π est le vecteur solution et Q est la matrice de transition de la chaîne de Markov (appelé aussi de *générateur infinitésimal*). Un des avantages de ces méthodes est d'éviter la résolution du système linéaire qui est généralement très grand.

Certains chaînes de Markov, de par leur structure, ont des solutions avec une *forme-produit*. Un cas simple de ce type de solution est le cas très particulier de chaîne de Markov : les *processus de naissance et mort* [128]. Il est possible de trouver des solutions à forme-produit pour des cas plus complexes que les processus de naissance et mort. Les cas les plus étudiés parmi les modèles Markoviens avec

solution à forme-produit sont probablement les *réseaux de files d'attente* [85, 46]. D'autres solutions à forme-produit sont connues en partant d'autres formalismes de modélisation, comme c'est le cas pour les *réseaux de Petri stochastiques* [58] et *réseaux de Petri stochastiques généralisés* [2, 1]. Pour quelques modèles de *réseaux d'automates stochastiques*, une solution à forme-produit a pu être trouvée [114, 62, 63, 60].

Les méthodes numériques peuvent être classées en deux groupes : les méthodes numériques *directes* et les méthodes numériques *itératives*.

La littérature décrivant les méthodes numériques de résolution de systèmes linéaires est très riche [86, 122, 52, 126]. De façon générale, les méthodes numériques de résolution de systèmes linéaires applicables aux chaînes de Markov sont des méthodes itératives. Les méthodes directes (*e.g.*, la méthode de Gauss [86, 52]) ne sont pas utilisables pour des modèles de très grande taille comme ceux qui nous préoccupent dans cette thèse.

Les méthodes itératives peuvent aller des méthodes simples (*e.g.*, la méthode de la puissance, Jacobi, Gauss-Siedel et sur-relaxation successive) jusqu'à des méthodes plus complexes (*e.g.*, les méthodes de projection : Arnoldi, GMRES, Lanczos, *etc*). Les descriptions de ces méthodes peuvent être trouvées en détail dans [126, 122].

D'autres méthodes numériques de résolution sont applicables aux modèles Markoviens, avec des propriétés structurelles du générateur Q :

- la résolution des modèles où le générateur est une *matrice géométrique* [99, 100] ;
- la résolution des modèles utilisant des *techniques de décomposition* [38, 115, 64, 98].

Il est aussi possible d'appliquer les méthodes numériques de résolution aux modèles Markoviens qui donnent une solution purement numérique de la résolution du système $\pi Q = 0$, telles que :

- la résolution des modèles où le générateur est stocké sous une *forme creuse* [52, 122, 126], *i.e.* seulement les éléments non-nuls et leur position sont stockés ;
- la résolution des modèles où le générateur est représenté par un *format tensoriel* [112, 48, 20, 55, 10], *i.e.*, le générateur est décrit par une formule contenant des opérateurs matriciels de l'algèbre tensorielle [42].

Des méthodes de résolution qui utilisent le générateur stocké sous une forme creuse représentent un gain significatif par rapport à l'utilisation de matrices stockées sous format plein. Les restrictions ne sont plus en fonction du carré de la taille du problème (*i.e.*, de l'ordre du générateur), mais en fonction du nombre d'éléments non-nuls du générateur.

Le stockage sous format tensoriel de la matrice de transition (appelé de *descripteur*) est de plus en plus utilisé [48, 20, 98, 10] à cause de sa faible exigence au niveau de la quantité de mémoire demandée (des matrices bien plus petites que la matrice de transition sont stockées). Si les opérations de base sont déjà très efficaces pour les générateurs sous format creux, le plus grand défi pour l'application des méthodes de résolution pour les descripteurs sous format tensoriel reste la faible efficacité de ces mêmes opérations de base. La multiplication d'un vecteur par le descripteur est l'opération fondamentale de toutes les méthodes itératives (simples ou sophistiquées).

Dans le cadre de cette thèse, nous nous sommes intéressés aux méthodes itératives, plus particulièrement à l'opération fondamentale des méthodes itératives : la multiplication d'un vecteur de probabilité par le descripteur de la chaîne de Markov.

1.2 Objectifs de cette thèse

Dans cette thèse, les travaux présentés se situent dans le contexte de la recherche de méthodes d'évaluation de performance liée au formalisme des Réseaux d'Automates Stochastiques.

Les objectifs de cette thèse s'articulent autour de deux axes de recherche : les *formalismes de modélisation* et les *méthodes numériques*. Plus précisément, nous abordons la génération de l'espace d'états atteignables d'un modèle décrit par le formalisme des réseaux d'automates stochastiques afin de calculer d'une façon performante l'espace d'états du système modélisé. Et de plus en matière de méthode, numérique, dans le contexte des réseaux d'automates stochastiques à temps discret dont le descripteur est représenté sous un format tensoriel, appelé *produit tensoriel complexe*, nous proposons une méthode de multiplication vecteur-matrice que prend en compte l'aspect compositionnel du modèle.

Les principales contributions de cette thèse sont :

- la définition d'une formule tensorielle qui représente les relations de transition entre les états d'un modèle décrit par les réseaux d'automates stochastiques à temps continu (cette formule est appelée de *descripteur d'atteignabilité* et sera présentée en détail dans le chapitre 4) ;

- la définition de méthodes de génération de l'espace d'états atteignables de modèles compositionnels à temps continu. L'avancée par rapport à l'état de l'art a été de proposer de méthodes qui prennent en compte des fonctions qui expriment des relations entre les composants des modèles. La contribution de ce travail a conduit à une publication dans la conférence internationale *QEST* (*Quantitative Evaluation of SysTems*) [123] ;

- la preuve de propriétés de l'Algèbre Tensorielle Complexe (une algèbre adaptée à la composition parallèle des réseaux d'automates stochastiques à temps discret pour la représentation du descripteur) ;

- la définition d'une méthode de multiplication d'un vecteur de probabilité par le descripteur d'un modèle exprimé sous forme d'un produit tensoriel complexe, sans jamais générer la matrice qui représente ce descripteur.

1.2.1 Plan de la thèse

Cette thèse s'articule autour de deux partie : *Réseaux d'Automates Stochastiques à temps continu* et *Réseaux d'Automates Stochastiques à temps discret*.

Nous dédions la première partie de cette thèse à la présentation et définition du formalisme des réseaux d'automates stochastiques à temps continu afin de proposer des méthodes pour la génération de l'espace d'états atteignables de modèles décrits par ce formalisme. Alors que, dans la deuxième partie

de cette thèse, nous présentons les définitions et notations du formalisme des réseaux d'automates stochastiques à temps discret, ainsi que la proposition d'une algèbre tensorielle adaptée à la composition parallèle de ce formalisme, afin de présenter une méthode de multiplication d'un vecteur par le descripteur d'un modèle décrit par les réseaux d'automates stochastiques à temps discret.

Réseaux d'Automates Stochastiques à temps continu

Le chapitre 2 traite du formalisme des réseaux d'automates stochastiques à temps continu. Nous commençons donc le chapitre présentant le formalisme de façon informelle afin de permettre la compréhension des modèles développés. Ensuite, une description formelle du formalisme est présentée pour éviter toute ambiguïté dans les modèles.

Nous présentons dans le chapitre 3 des exemples de modélisation de systèmes utilisant le formalisme des réseaux d'automates stochastiques à temps continu. Les exemples présentés dans ce chapitre ont pour but de présenter les différents types d'interaction entre automates d'un modèle. Ces exemples sont aussi utilisés pour tester l'efficacité des méthodes présentées dans le chapitre 5.

Le chapitre 4 présente une formule tensorielle qui exprime les relations de transition entre les états d'un modèle. La formule tensorielle qui permet la représentation de ces relations est dénommée *descripteur d'atteignabilité*. L'objectif du descripteur d'atteignabilité est de représenter les relations de transition entre les états globaux d'un modèle (et non à quels taux ils peuvent être atteignables), *i.e.*, le descripteur d'atteignabilité est un descripteur restreint aux relations d'atteignabilité, sans considérer les taux.

Nous présentons dans le chapitre 5 les méthodes de génération de l'espace d'états atteignables d'un modèle compositionnel. L'avancée par rapport d'autres méthodes de génération trouvées dans la littérature est de proposer des méthodes qui prennent en compte des fonctions qui expriment des relations entre les composants des modèles.

Dans le chapitre 6, nous présentons quelques mesures de performance obtenues avec les méthodes présentées dans le chapitre 5 pour la génération de l'espace d'états atteignables de modèles qui utilisent des taux fonctionnels. Les mesures présentées dans ce chapitre explicitent les différences entre les méthodes de génération proposées dans cette thèse.

Réseaux d'Automates Stochastiques à temps discret

Dans le chapitre 7, nous présentons le formalisme des réseaux d'automates stochastiques à temps discret. La définition formelle présentée dans ce chapitre nous permet de définir un modèle SAN pour une sémantique en temps discret et d'obtenir la chaîne de Markov associée à ce modèle.

Le chapitre 8 présente des exemples de modélisation de systèmes utilisant le formalisme des réseaux d'automates stochastiques à temps discret. Ces exemples ont pour but d'illustrer plusieurs façons de représenter des interactions dans les modèles et de présenter les spécificités du formalisme SAN à temps discret.

Dans le chapitre 9, nous présentons une nouvelle algèbre tensorielle (appelée *Algèbre Tensorielle Complexe*) adaptée à la composition parallèle des réseaux d'automates stochastiques à temps discret.

Nous démontrons des propriétés de cette algèbre qui servent de base aux méthodes itératives pour la résolution de la chaîne de Markov associée au modèle SAN à temps discret.

Le chapitre 10 présente une formule tensorielle (appelée *descripteur discret*) basée sur l'algèbre tensorielle complexe présentée dans le chapitre 9 qui a pour but la représentation compacte de la matrice de transition d'un modèle SAN à temps discret.

Enfin, dans le chapitre 11, nous présentons une méthode de multiplication d'un vecteur de probabilité par le descripteur discret d'un modèle SAN, sans jamais générer la matrice qui représente ce descripteur, *i.e.*, sans jamais générer la matrice de transition de la chaîne de Markov du modèle. Cette méthode vise à exploiter des propriétés de l'algèbre tensorielle complexe présentée dans le chapitre 9 de façon à ce que la multiplication par un opérateur sur l'espace produit du modèle soit remplacée par une suite d'opérations qui manipulent des données de la taille d'une composante du modèle (et pour toutes les composantes).

Conclusion

Dans la conclusion, nous présentons un bilan des travaux développés dans cette thèse, ainsi que les travaux en cours. Finalement, nous présentons des considérations sur les travaux futurs envisagés pour donner suite à cette thèse.

Annexe

Dans l'annexe intitulée *Algèbre Tensorielle*, nous présentons un résumé des notations et propriétés de l'*algèbre tensorielle classique* avec les opérateurs *somme tensorielle* et *produit tensoriel*. Nous présentons aussi les notations et propriétés de l'*algèbre tensorielle généralisée* avec les opérateurs *somme tensorielle généralisée* et *produit tensoriel généralisé* utilisés au long de cette thèse.

Première partie

Réseaux d'Automates Stochastiques à temps continu

Chapitre 2

Formalisme des Réseaux d'Automates Stochastiques (SAN) à temps continu

À cause de la nécessité de manipuler de grands volumes de données nécessaires pour modéliser des systèmes complexes, les recherches actuelles visent à trouver des solutions de plus en plus rapides et performantes. Dans ce contexte, le formalisme des Réseaux d'Automates Stochastiques (SAN - *Stochastic Automata Networks*) [110, 4, 54, 8, 124] propose des techniques qui permettent de traiter la modélisation de systèmes complexes.

Le formalisme SAN est basé sur le formalisme des Chaînes de Markov [126]. Néanmoins, la modélisation de systèmes complexes en utilisant le formalisme des Chaînes de Markov peut parfois rendre impraticable la réalisation du modèle à cause de la complexité de sa description, *e.g.*, un système avec des centaines de millions d'états.

Ainsi, le formalisme des Réseaux d'Automates Stochastiques, centre d'intérêt de ce chapitre[1], fournit une façon compacte et performante pour la description de systèmes complexes à grand espace d'états, et des méthodes pour optimiser des solutions stationnaires et transitoires. Ainsi, le formalisme SAN associe des techniques qui visent à éliminer (ou à réduire au maximum) la difficulté de modélisation en améliorant la vitesse d'obtention d'indices de performance, et en facilitant la modélisation des systèmes à grand espace d'états.

Un réseau d'automates stochastiques représente une chaîne de Markov. En conséquence, les propriétés des Chaînes de Markov sont aussi applicables à l'objet décrit par des Réseaux d'Automates Stochastiques. Le formalisme SAN permet la modélisation d'un système complexe par des petits sous-systèmes presque indépendants de façon à ce que soit possible le parallélisme (quand les automates n'interagissent pas) et le synchronisme (quand les automates interagissent).

D'abord, dans la section 2.1, le formalisme SAN est décrit d'une façon informelle pour présenter, sans préoccupation théorique, les idées de base de l'outil : les automates, les événements, les taux et probabilités (constants ou fonctionnels), *etc.* Ensuite, dans la section 2.2, une description formelle est donnée afin de lever les ambiguïtés et pour obtenir le descripteur Markovien du système modélisé.

[1] La présentation du formalisme SAN à temps continu donnée dans ce chapitre est commune dans cette thèse et dans [15].

2.1 Description informelle des SAN à temps continu

Le formalisme des Réseaux d'Automates Stochastiques a été proposé par Plateau [110]. L'idée principale du formalisme SAN est de modéliser un système en plusieurs sous-systèmes, *i.e.*, un système composé de modules. Cette modularité définie par le formalisme permet le stockage et la solution de *systèmes complexes* en faisant face au problème de l'*explosion combinatoire de l'espace d'états*. Le formalisme SAN est utilisé pour la modélisation de systèmes à *grand espace d'états*.

Chaque sous-système est représenté par un *automate stochastique*. Les transitions entre les états de chaque automate représentent les transitions d'un processus stochastique à échelle de temps continue ou discrète, grâce à des distributions exponentielles ou géométriques respectivement.

Il est intéressant de remarquer que tout modèle SAN peut être représenté par un seul automate stochastique qui a tous les états possibles du système modélisé. Cet automate correspond à la chaîne de Markov équivalente au modèle SAN (Section 2.1.6, page 20).

Les modèles SAN présentés dans ce chapitre peuvent être interprétés dans une échelle de temps discrète ou continue. Cependant, l'explication et les exemples présentés dans ce chapitre font référence à l'échelle de temps continue (*taux d'occurrence*), à la différence des modèles à échelle de temps discrète (*probabilité d'occurrence*). Un modèle SAN à échelle de temps continue gère une chaîne de Markov à échelle de temps continue (CTMC - *Continuous Time Markov Chain*), tandis qu'un modèle SAN à échelle de temps discrète gère une chaîne de Markov à échelle de temps discrète (DTMC - *Discrete Time Markov Chain*). Une description formelle du formalisme SAN à échelle de temps discrète, qui est le formalisme d'intérêt de la deuxième partie de cette thèse, sera présenté dans le chapitre 7. Dans cette thèse, on va adopter les notations suivantes pour la définition de modèles SAN. Pour le temps continu, nous allons présenter ce qui se trouve dans [54, 8, 124].

✍ **Soit**

$\mathcal{A}^{(i)}$	i-ème automate d'un modèle SAN, où $\mathcal{A}^{(1)}$ est le premier automate ;
$x^{(i)}$	x-ème état de l'automate $\mathcal{A}^{(i)}$, où $0^{(i)}$ est le premier état de l'automate $\mathcal{A}^{(i)}$;
e	l'identificateur d'un événement (local ou synchronisant)[2] ;
$e(\pi)$	l'identificateur d'un événement e avec une probabilité de routage π ;

Les probabilités de routage (π) sont utilisées quand un événement a plusieurs alternatives de transition. La probabilité peut être omise dans le cas où elle est égale à 1. De plus, la somme des probabilités pour les transitions du même événement partant du même état local doit être égale à 1.

Dans ce qui suit, on va introduire les notions d'automates et d'événements, qui sont les notions de base du formalisme SAN. Ensuite, on présente les concepts d'éléments fonctionnels, ainsi que le concept de fonction d'atteignabilité et de fonction indice de performance. Enfin, on montre comment faire la génération d'une chaîne de Markov à partir d'un modèle SAN.

[2]Dans le contexte de cette thèse, on utilise le caractère e comme identificateur d'un événement. Cependant n'importe quel caractère peut être utilisé pour définir l'identificateur d'un événement.

2.1.1 Automates stochastiques

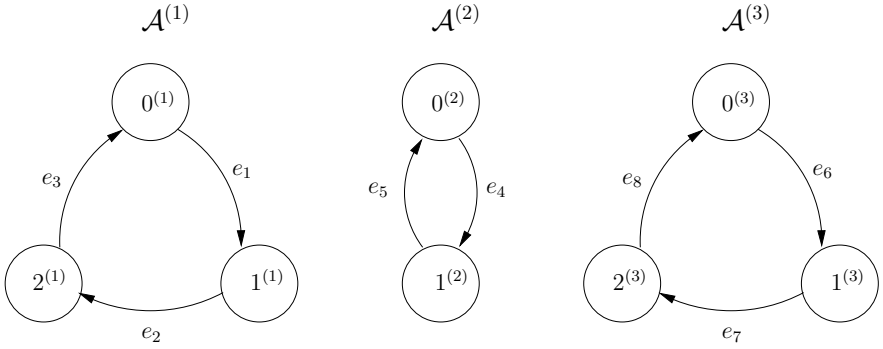
Un automate stochastique est un modèle mathématique d'un système qui possède des entrées et sorties discrètes. Le système peut se trouver dans un quelconque état ou configuration interne. L'état dans lequel le système se trouve résume les informations sur les entrées précédentes et indique ce qui est nécessaire pour déterminer le comportement du système pour les entrées suivantes [76].

Selon cette définition, on peut décrire un automate stochastique comme un *ensemble d'états* et un *ensemble de transitions* entre eux [113]. La désignation stochastique attribuée aux automates signifie que le traitement du temps se fait par des variables aléatoires, qui suivent une distribution exponentielle sur une échelle de temps continue ou une distribution géométrique sur une échelle de temps discrète.

Graphiquement, un réseau d'automates stochastiques peut être représenté par un ensemble de graphes orientés, où chaque graphe est associé à un automate. Les *noeuds* d'un graphe représentent les *états* d'un automate et les *arcs* entre les états représentent la *transition* d'un état à l'autre.

L'*état local* du système modélisé par un SAN est l'état individuel de chaque automate du modèle. L'*état global* d'un modèle est composé par l'ensemble des états locaux de chaque automate du système modélisé. Le changement de l'état global du système est le résultat du changement de l'état local d'un (au moins) automate du modèle.

Le changement d'un certain état local à l'autre est fait par l'occurrence des événements qui provoquent des transitions. Les transitions sont des primitives qui indiquent la possibilité de changement d'un état à l'autre. Chaque transition est associée à un ou plusieurs *événements*. Dans FIG. 2.1, on présente un modèle SAN avec trois automates complètement indépendants.



Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux
loc	e_1	α	loc	e_4	δ	loc	e_6	μ
loc	e_2	β	loc	e_5	λ	loc	e_7	σ
loc	e_3	γ						

FIG. 2.1 – Modèle SAN avec 3 automates indépendants

Dans ce premier exemple, les automates $\mathcal{A}^{(1)}$ et $\mathcal{A}^{(3)}$ du modèle possèdent trois états[3] chacun : les états $0^{(1)}$, $1^{(1)}$ et $2^{(1)}$ correspondent aux états du premier automate, les états $0^{(3)}$, $1^{(3)}$ et $2^{(3)}$ correspondent aux états du troisième automate et l'automate $\mathcal{A}^{(2)}$ possède seulement deux états $0^{(2)}$ et $1^{(2)}$.

[3]Indépendamment du choix des étiquettes des états locaux d'une automate, chaque état possède une représentation numérique récurrente de l'ordre de la modélisation, *i.e.*, un état possède une représentation numérique entre les valeurs $0, 1, 2, \ldots, n^{(i)} - 1$, où $n^{(i)}$ est le nombre d'états de l'automate $\mathcal{A}^{(i)}$.

Huit événements sont modélisés dans cet exemple : trois événements (e_1, e_2 et e_3) se produisent dans l'automate $\mathcal{A}^{(1)}$; deux événements (e_4 et e_5) se produisent dans l'automate $\mathcal{A}^{(2)}$; et les autres trois événements (e_6, e_7 et e_8) se produisent dans l'automate $\mathcal{A}^{(3)}$. Vu qu'il n'y a pas d'interaction entre les automates, la solution de ce type de modèle peut être obtenue de manière totalement indépendante pour chacun des automates, *i.e.*, chaque automate peut être évalué individuellement et pour le système global on a une forme produit.

2.1.2 Événements

L'événement est l'entité du modèle responsable de l'occurrence d'une transition qui modifie l'*état global* du modèle. Un ou plusieurs événements peuvent être associés à une transition. Une transition est provoquée par l'occurrence d'un des ses événements.

Chaque événement doit avoir un taux de franchissement et une probabilité de routage. Le taux de franchissement et la probabilité de routage peuvent avoir des valeurs constantes ou fonctionnelles (Section 2.1.3, page 18). Le non-déterminisme entre le tirage des différents événements associés à une même transition est traité par un comportement Markovien, *i.e.*, tous les événements habilités peuvent se produire et leurs taux de franchissement respectifs définissent la fréquence avec laquelle ils vont se produire.

Deux types d'événements peuvent être modélisés par le formalisme SAN : événements *locaux* ou *synchronisants*.

2.1.2.1 Événements locaux

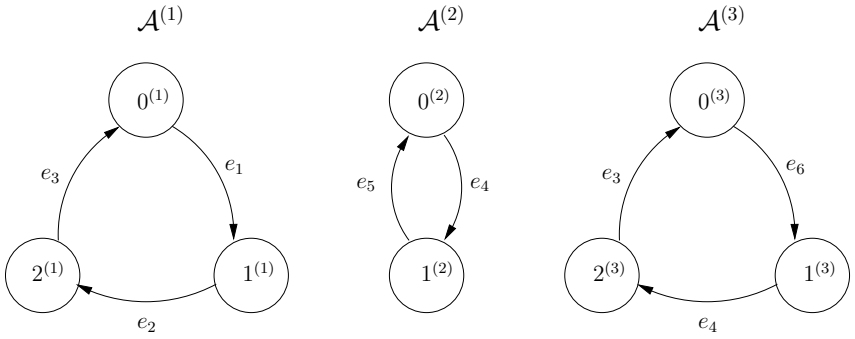
Les événements locaux sont utilisés dans les modèles SAN pour modifier l'état interne (local) d'un *seul automate*, sans que cette modification cause un changement d'état dans autre automate du modèle. Ce type d'événement est particulièrement intéressant, vu qu'il permet que plusieurs automates aient un comportement *parallèle*, travaillant de façon *indépendante* entre eux.

Notamment, on peut observer des exemples d'événements locaux dans FIG. 2.1, qui contient exclusivement par ce type d'événement.

2.1.2.2 Événements synchronisants

Si les événements locaux sont très simples à définir, les événements synchronisants, au contraire, sont plus complexes. Les événements synchronisants changent l'état local de *deux ou plus automates* simultanément, *i.e.*, l'occurrence d'un événement synchronisant dans un automate *oblige* l'occurrence de ce même événement dans les autres automates concernés. Il est possible de modéliser l'interaction entre des automates en utilisant des événement synchronisants. Cette interaction se fait sous la forme de synchronisme dans le tirage des transitions.

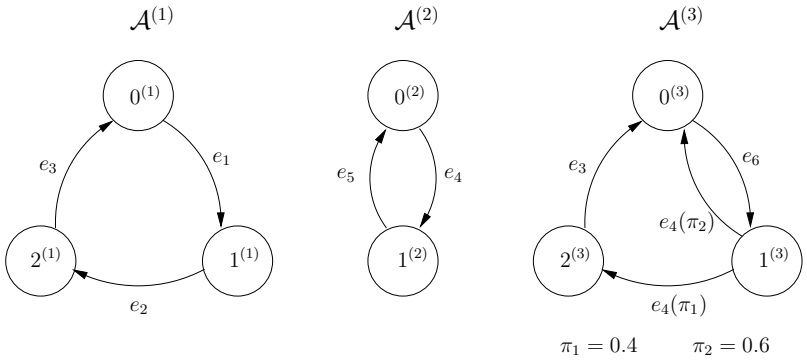
Un événement est classé comme *local* ou *synchronisant* selon l'occurrence de l'identificateur de l'événement e dans l'ensemble des événements d'un automate. Un événement est dit *local* si son identificateur est associé aux transitions d'un *seul automate*. Si l'identificateur d'un événement est associé aux transitions de *plusieurs automates*, cet événement est classé comme *synchronisant*.



Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux
loc	e_1	α	syn	e_4	δ	loc	e_6	μ
loc	e_2	β	loc	e_5	λ			
syn	e_3	γ						

FIG. 2.2 – Modèle SAN avec événements synchronisants

FIG. 2.2 est un modèle SAN modifié par rapport au modèle présenté dans FIG. 2.1. Dans ce nouveau modèle, les événements e_3 et e_4 ne sont plus locaux mais sont des événements synchronisants, car l'identificateur de ces deux événements apparaît dans deux automates. L'événement e_3 synchronise les automates $\mathcal{A}^{(1)}$ et $\mathcal{A}^{(3)}$ de telle façon à changer les états locaux de chaque automate simultanément, *i.e.*, l'occurrence de l'événement e_3 change l'état de l'automate $\mathcal{A}^{(1)}$ de $2^{(1)}$ vers $0^{(1)}$ en même temps qu'elle change aussi l'état de l'automate $\mathcal{A}^{(3)}$ de $2^{(3)}$ vers $0^{(3)}$. De façon analogue, l'occurrence de l'événement e_4 change l'état de l'automate $\mathcal{A}^{(2)}$ de $0^{(2)}$ vers $1^{(2)}$ en même temps qu'elle change l'état de l'automate $\mathcal{A}^{(3)}$ de $1^{(3)}$ vers $2^{(3)}$.



Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux
loc	e_1	α	syn	e_4	δ	loc	e_6	μ
loc	e_2	β	loc	e_5	λ			
syn	e_3	γ						

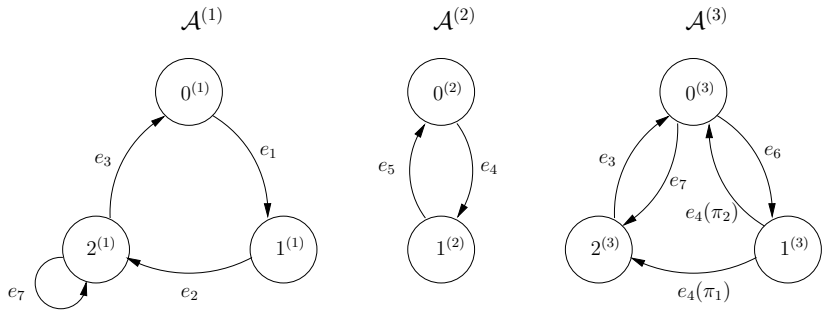
FIG. 2.3 – Modèle SAN avec probabilités de routage

Des événements synchronisants, ainsi que les événements locaux, peuvent avoir des probabilités de routage pour un même événement. Les probabilités de routage sont utilisées pour déterminer dans quelles proportions les transitions alternatives possibles se produisent. Dans FIG. 2.3, l'événement synchronisant e_4 peut déclencher deux transitions différentes. Ainsi, l'occurrence de l'événement dans l'automate $\mathcal{A}^{(3)}$ déclenche avec probabilité π_1 la transition de l'état $1^{(3)}$ vers l'état $2^{(3)}$, et avec probabilité π_2 la transition de l'état $1^{(3)}$ vers l'état $0^{(3)}$.

2.1.3 Taux et probabilités fonctionnels

Une possibilité d'exprimer une interaction entre des automates est l'utilisation d'événements synchronisants. Une autre façon est l'utilisation de fonctions pour définir des taux et/ou probabilités qui permet d'associer à un même événement différentes valeurs déterminant son occurrence en fonction de l'état local des autres automates du modèle. L'utilisation de taux et probabilités fonctionnels peut être employée autant que avec des événements locaux ou synchronisants.

Les taux et probabilités fonctionnels sont exprimés par des fonctions qui ont comme paramètres des états courants des automates du modèle. FIG. 2.4 présente une variation du modèle présenté dans FIG. 2.3, où l'événement e_1 possède un taux d'occurrence *fonctionnel* au lieu d'un taux d'occurrence *constant*, *i.e.*, l'événement e_1 se produit avec un taux exprimé par la fonction[4] f_1. Ici, $(st\ \mathcal{A}^{(3)} == 0)$ est la fonction caractéristique de l'ensemble d'états où l'automate $\mathcal{A}^{(3)}$ est dans l'état 0.



Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux
loc	e_1	f_1	syn	e_4	δ	loc	e_6	μ
loc	e_2	β	loc	e_5	λ	syn	e_7	σ
syn	e_3	γ						

FIG. 2.4 – Modèle SAN avec taux fonctionnel

Ainsi, la fonction f_1 détermine un taux d'occurrence variable pour l'événement e_1. Selon la fonction f_1, l'occurrence de l'événement e_1 se produira avec le taux suivant :

$$f_1 = \begin{cases} \alpha & \text{si l'automate } \mathcal{A}^{(3)} \text{ est dans l'état } 0^{(3)} \\ 0 & \text{si l'automate } \mathcal{A}^{(3)} \text{ est dans l'état } 1^{(3)} \\ 0 & \text{si l'automate } \mathcal{A}^{(3)} \text{ est dans l'état } 2^{(3)} \end{cases}$$

L'évaluation de f_1 donne le résultat en *faux* quand l'automate $\mathcal{A}^{(3)}$ est dans les états $1^{(3)}$ ou $2^{(3)}$, *i.e.*, la valeur résultant pour cette comparaison est *zéro*, qui annule le taux d'occurrence de l'événement e_1. Donc, le taux fonctionnel associé au tirage de l'événement e_1 est tel que, le taux est α si l'état de l'automate $\mathcal{A}^{(3)}$ est égal à $0^{(3)}$ et nul, ce qui empêchera que l'événement e_1 puisse être tiré, pour tous les autres cas.

Les probabilités de routage d'un événement peuvent aussi être exprimées par des fonctions. La définition de fonctions utilisées pour exprimer les probabilités fonctionnelles sont les mêmes que les fonctions utilisées pour exprimer les taux d'occurrence d'un événement.

[4]Dans cette thèse, les fonctions sont définies utilisant le notation adoptée par le logiciel PEPS [17].

Dans FIG. 2.4, il faut observer que les probabilités de routage π_1 et π_2 de l'événement e_4 sont aussi exprimées par une fonction. Comme on l'a dit précédemment, la somme des probabilités des transitions d'un événement à partir d'un même état doit être toujours égale à 1 (100%). Donc, de même si les probabilités de routage d'un événement sont exprimées par une fonction, cette caractéristique doit être respectée.

$$\pi_1 = \begin{cases} 1 & \text{si l'automate } \mathcal{A}^{(1)} \text{ est dans l'état } 1^{(1)} \\ 0 & \text{si l'automate } \mathcal{A}^{(1)} \text{ est dans l'état } 0^{(1)} \text{ ou } 2^{(1)} \end{cases}$$

$$\pi_2 = \begin{cases} 0 & \text{si l'automate } \mathcal{A}^{(1)} \text{ est dans l'état } 1^{(1)} \\ 1 & \text{si l'automate } \mathcal{A}^{(1)} \text{ est dans l'état } 0^{(1)} \text{ ou } 2^{(1)} \end{cases}$$

Dans cet exemple, les probabilités π_1 et π_2 s'excluent, *i.e.*, si le résultat de l'évaluation de la probabilité π_1 est égal à 1, le résultat de la probabilité π_2 sera égal à 0, et vice-versa. Cependant, des probabilités exprimées par des fonctions n'ont pas toujours l'obligation de l'exclusion.

2.1.4 Fonction d'atteignabilité

Il y a un autre utilisation des fonctions pour la modélisation dans le formalisme SAN : la *fonction d'atteignabilité*. Les expressions qui définissent la fonction d'atteignabilité sont décrites de la même façon que les fonctions pour les taux et probabilités fonctionnels. Cependant, ce type de fonction a un rôle différent dans le formalisme.

Comme la représentation d'un modèle SAN est faite de façon modulaire et l'automate global (équivalent à chaîne de Markov) est composé de la combinaison de tous les états des automates du modèle (Section 2.1.6, page 20), il faut déterminer une fonction sur le modèle global qui définit les *états atteignables* du modèle SAN.

La définition de quels états peuvent être *atteignables* ou *accessibles* dans un modèle SAN est donnée par la *fonction d'atteignabilité*. Jusqu'à présent, la fonction d'atteignabilité est définie par l'utilisateur. Il faut aussi remarquer que la fonction d'atteignabilité pourrait être calculée à partir d'un état initial et de transitions définies par le modèle. Ceci sera l'objet de notre contribution du chapitre 5.

Cette fonction utilise les règles adoptées pour la définition des taux et probabilités fonctionnels.

La notion de fonction d'atteignabilité est plus claire si on imagine, par exemple, un modèle de partage de ressources avec N clients distincts qui partagent R ressources communes identiques entre elles. Ce système peut être modélisé par le formalisme SAN en utilisant un automate avec deux états pour chaque client. L'état $0^{(i)}$ indique que la ressource n'est pas utilisée par le client i, tant que l'état $1^{(i)}$ indique que la ressource est utilisée par le client i. Il est facile d'imaginer que s'il y a plus de clients que de ressources ($N > R$), l'état global qui représente tous les clients utilisant une ressource ne pourra pas se produire. Les états qui possèdent telle caractéristique sont nommés de *états non-atteignables* et doivent être éliminés du modèle par la *fonction d'atteignabilité*. La probabilité du modèle se trouver dans quelconque de ces états est égale à *zéro*. La fonction d'atteignabilité correcte pour le modèle de partage de ressources décrit ci-dessus est[5] :

$$reachability = nb\,[\mathcal{A}^{(1)}..\mathcal{A}^{(N)}]\ 1 \leq R$$

[5]Notation utilisée par le logiciel PEPS [17].

Ainsi, la fonction d'atteignabilité représente un concept important dans la description de modèles SAN. Néanmoins, la complexité de certains modèles peut rendre la description de la fonction d'atteignabilité difficile à réaliser.

2.1.5 Fonction à intégrer

Une autre utilisation des fonctions dans la modélisation avec le formalisme SAN est l'utilisation de *fonctions à intégrer*. Une fonction à intégrer est utilisée pour l'obtention d'indice de performance ou de fiabilité moyens sur le modèle. Ces fonctions évaluent, par exemple, la probabilité du modèle SAN de se trouver dans un état donné. Calculer un indice de performance moyen revient à intégrer cette fonction sur le *vecteur de probabilité* stationnaire ou transitoire du modèle.

Par exemple, avec le modèle de partage de ressources (décrit dans la section 2.1.4), on peut définir la fonction u pour déterminer la probabilité de l'automate $\mathcal{A}^{(1)}$ de ne pas utiliser une ressource (quand l'automate $\mathcal{A}^{(1)}$ se trouve dans l'état $0^{(1)}$).

$$u = \Big(st(\mathcal{A}^{(1)}) == 0\Big)$$

Toutes les fonctions utilisées dans les modèles SAN sont décrites de la même façon, ce qui les différencie est l'utilisation de chaque fonction dans le modèle.

2.1.6 Construction de la chaîne de Markov équivalente

Bien qu'un modèle SAN soit représenté par un ensemble d'automates, il peut aussi être représenté par un *seul* automate qui contient tous les états globaux possibles du modèle. Cet automate est une représentation de la chaîne de Markov du système modélisé [55]. Une autre représentation est sa matrice de transition, ou bien le descripteur qui sera calculé ultérieurement (Section 2.1.7, page 21).

Dans cet automate global, il n'y a plus d'événements synchronisants, mais seulement des événements locaux. Les arcs de cet automate global sont étiquetés par les taux d'occurrence des événements et les probabilités de routage. Les durées de résidence dans chaque état sont des variables aléatoires de distribution exponentielle. Donc, à un instant de temps, le changement vers un état suivant ne dépend que de l'état courant et pas du temps écoulé dans cet état.

La représentation graphique de la chaîne de Markov équivalente est un graphe à *états-transitions* qui représente l'automate global du système. Les états de la chaîne de Markov sont formés à partir du produit cartésien des états locaux de chaque automate du modèle SAN. Dans FIG. 2.5, on présente la chaîne de Markov (CTMC) équivalente au modèle présenté dans FIG. 2.4, en supposant que l'état global initial est égal à 000.

Pour représenter clairement les états de la CTMC équivalente, on omet les indices de chaque automate, en supposant que le chiffre le plus à gauche représente l'état du premier automate et le chiffre plus à droite représente l'état du dernier automate, *i.e.*, l'état 201 est équivalent à l'état $2^{(1)}$ dans l'automate $\mathcal{A}^{(1)}$, $0^{(2)}$ dans l'automate $\mathcal{A}^{(2)}$ et $1^{(3)}$ dans l'automate $\mathcal{A}^{(3)}$.

Le formalisme SAN propose une vision modulaire du système, alors que la CTMC équivalente (ou l'automate global) représente une vision "centralisée" (mais potentiellement très grande) du même sys-

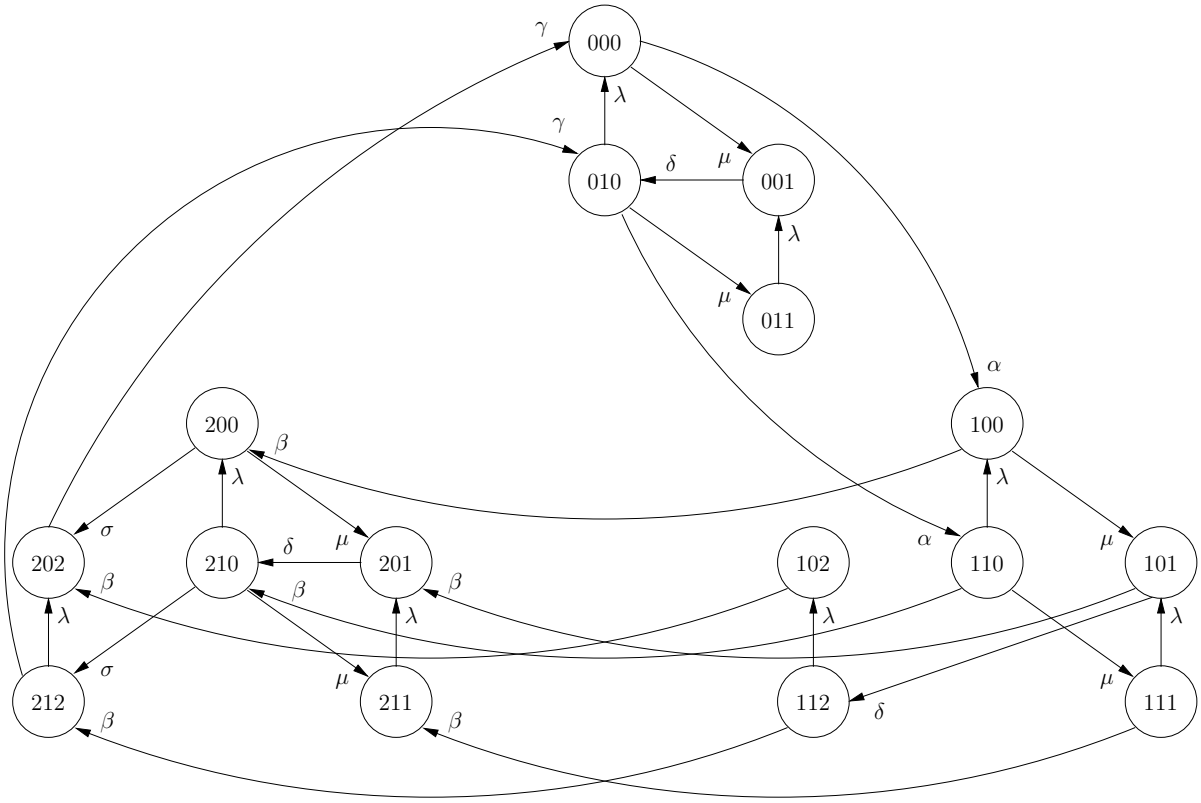


FIG. 2.5 – CTMC équivalente au modèle SAN dans FIG. 2.4

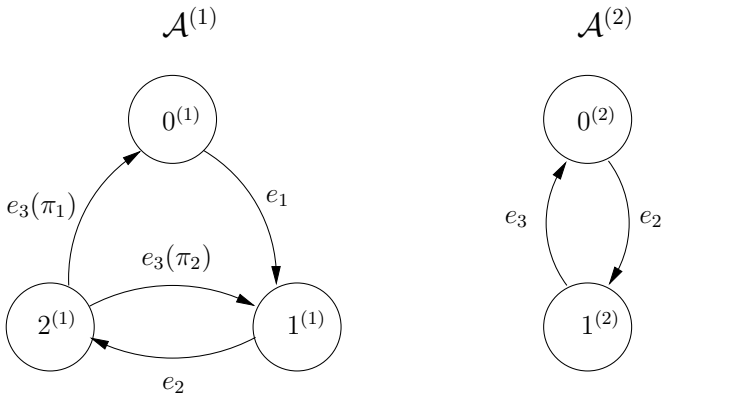
tème. Le générateur infinitésimal de cette chaîne de Markov est représenté directement par une matrice, dont la taille est exprimée par le nombre d'états atteignables du modèle. Comme on peut l'observer dans FIG. 2.4, il y a seulement 16 (sur 18) états atteignables dans le modèle. Les états 002 et 012 ne sont pas représentés, vu qu'ils ne sont jamais *atteignables* dans le modèle (en faisant des tirages successifs des transitions possibles du modèle, on n'atteint jamais ces deux états).

Vu que le formalisme SAN propose une représentation modulaire du système, la représentation du générateur infinitésimal de la chaîne de Markov équivalente au modèle SAN peut aussi être exprimée de façon modulaire. Conséquemment, au lieu de l'obtention d'une seule matrice de transition, on a un ensemble de *petites* matrices pour chaque automate de taille $(n^{(i)})^2$, où $n^{(i)}$ est le nombre d'états de l'automate $\mathcal{A}^{(i)}$. L'obtention du générateur infinitésimal à partir de cet ensemble de petites matrices est exprimée par un format tensoriel, qui est aussi nommé *descripteur Markovien*. Dans ce qui suit, on va démontrer comment on peut obtenir le descripteur Markovien à partir d'un modèle SAN.

2.1.7 Descripteur Markovien

En plus de faciliter la description des modèles, un des avantages du formalisme SAN est de générer automatiquement et d'une façon compacte le générateur infinitésimal Q de la chaîne de Markov équivalente du modèle. Cette façon compacte de représenter le générateur infinitésimal est une formule mathématique nommée *descripteur* Markovien [55, 111].

La formule du descripteur Markovien est basée sur l'algèbre tensorielle. Quand le modèle utilise des taux et/ou probabilités fonctionnels, il est impératif d'utiliser l'algèbre tensorielle généralisée pour la représentation du descripteur Markovien. Cependant, l'utilisation de l'algèbre tensorielle généralisée ne change rien par rapport à la structure de la formule du descripteur. Les propriétés de l'algèbre tensorielle classique et de l'algèbre tensorielle généralisée sont présentée dans Annexe A. D'autres formalismes, tels que les Réseaux de Petri Stochastiques Généralisés [49] et l'Algèbre de Processus [75] par exemple, utilisent des techniques similaires pour représenter le générateur infinitésimal d'une chaîne de Markov d'une façon compacte et performante.



Type	Even.	Taux
loc	e_1	f_1
syn	e_2	β
syn	e_3	μ

$$f_1 = (st\mathcal{A}^{(2)} == 0) \times \alpha$$
$$\pi_1 = 0.4$$
$$\pi_2 = 0.6$$

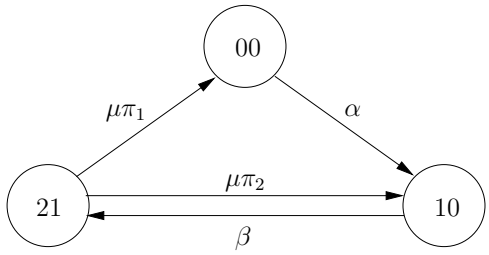
FIG. 2.6 – Modèle SAN

Pour simplifier la démonstration de la génération du descripteur Markovien d'un modèle SAN, on va utiliser l'exemple présenté dans FIG. 2.6, où l'état global initial est égal à 00. Ce petit exemple a six états potentiels, mais seulement trois des états sont atteignables : 00, 10 et 21.

La chaîne de Markov équivalente au modèle montrée dans FIG. 2.6 est présentée dans FIG. 2.7. Il y a seulement trois états atteignables, et le changement de l'état global 00 vers l'état 10 est exprimé par le taux fonctionnel f_1.

À partir de ce petit exemple, on montre comme s'obtient le descripteur Markovien d'un modèle SAN. La formule mathématique qui représente le descripteur est décomposée en deux parties : *locale* et *synchronisante*. La partie locale représente les transitions des événements locaux des automates du modèle, tandis que la partie synchronisante représente les transitions de chaque événement synchronisant.

Par rapport à partie locale, à chaque automate est associé une matrice nommée $Q_l^{(i)}$, qui regroupe toutes les transitions des événements locaux de l'automate $\mathcal{A}^{(i)}$. La partie locale du descripteur est représentée par la matrice Q_l qui est définie comme la somme tensorielle des matrices locales de chaque automate. Pour l'exemple présenté dans FIG. 2.6, la partie locale du modèle est représenté par :



	00	01	10	11	20	21
00	-α	0	α	0	0	0
01	0	0	0	0	0	0
10	0	0	-β	0	0	β
11	0	0	0	0	0	0
20	0	0	0	0	0	0
21	$\mu\pi_1$	0	$\mu\pi_2$	0	0	-μ

FIG. 2.7 – CTMC équivalente au modèle FIG. 2.6

$$Q_l = Q_l^{(1)} \underset{g}{\oplus} Q_l^{(2)} = \begin{pmatrix} -f_1 & f_1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \bigoplus_{g} \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Chaque matrice $Q_l^{(i)}$ a la taille n_i, où n_i est le nombre d'états de l'automate $\mathcal{A}^{(i)}$. Il est intéressant de remarquer que la matrice $Q_l^{(2)}$ est une matrice nulle, *i.e.*, tous les éléments sont égaux à zéro, car l'automate $\mathcal{A}^{(2)}$ (FIG. 2.6) ne possède pas d'événements locaux.

Les matrices locales sont formées à partir des transitions locales de chaque automate. Ainsi, l'élément sur la ligne i et la colonne j de la matrice représente la transition de l'état i vers l'état j de l'automate. Cet élément possède le taux (constant ou fonctionnel) de l'événement local qui correspond à la transition entre les états de l'automate. L'élément diagonal ii (ligne i, colonne i) représente l'ajustement de telle manière que la somme de tous les éléments de la ligne soit égale à *zéro*. L'état global 01 n'est pas un état atteignable (FIG. 2.7). Donc, la transition de l'état 01 vers 11 ne se produit pas. Ainsi, la fonction f_1 est évaluée à 0 pour l'occurrence de l'événement dans l'état 01. En utilisant la somme tensorielle généralisée [55] pour l'obtention de la matrice Q_l, on a la matrice suivante :

$$Q_l = \left(\begin{array}{cc|cc|cc} \text{-}\alpha & 0 & \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

À la différence de la partie locale du descripteur Markovien qui est obtenue d'une façon directe (une matrice pour chaque automate du modèle), la partie synchronisante est composée, pour chaque événement, de deux ensembles de matrices : *positives* et *négatives*.

Les matrices positives représentent l'occurrence des événements synchronisants, tandis que les matrices négatives représentent l'ajustement diagonal. Donc, chaque événement synchronisant est associé à une paire de matrice (positive et négative) pour représenter son occurrence pour chaque automate concerné. Les matrices positives et négatives synchronisantes associées à l'automate i possèdent aussi la taille n_i (en fonction du nombre d'états de chaque automate $\mathcal{A}^{(i)}$).

La matrice positive $Q^{(i)}_{e_k^+}$ représente l'occurrence de l'événement e_k dans l'automate $\mathcal{A}^{(i)}$. Pour l'exemple dans FIG. 2.6, le modèle SAN a deux événements synchronisants : e_2 et e_3. Donc, les matrices positives pour les événements synchronisants de ce modèle sont :

$$Q_{e_2^+} = Q^{(1)}_{e_2^+} \underset{g}{\otimes} Q^{(2)}_{e_2^+} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \beta \\ 0 & 0 & 0 \end{pmatrix} \bigotimes_g \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & \beta \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

$$Q_{e_3^+} = Q^{(1)}_{e_3^+} \underset{g}{\otimes} Q^{(2)}_{e_3^+} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \mu\pi_1 & \mu\pi_2 & 0 \end{pmatrix} \bigotimes_g \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \mu\pi_1 & 0 & \mu\pi_2 & 0 & 0 & 0 \end{array}\right)$$

Les matrices positives sont remplies de la même façon que les matrices locales, où l'élément ij représente la transition de l'état i vers l'état j de l'automate. Cependant, un événement synchronisant concerne deux ou plusieurs automates. Donc, pour chaque événement synchronisant, il faut associer deux (ou plus) matrices pour représenter l'occurrence de cet événement.

On peut prendre la convention que la matrice positive correspondante à l'automate de plus petit indice de l'événement reçoit le taux d'occurrence de l'événement. Les autres matrices des automates concernés de cet événement synchronisant reçoivent la valeur 1 pour chaque transition entre les états locaux. Par exemple, pour l'événement synchronisant e_2, la matrice positive $Q^{(1)}_{e_2^+}$ reçoit le taux d'occurrence β dans la position 12 (ligne 1, colonne 2) équivalent à la transition de l'état 1 vers l'état 2 de l'automate $\mathcal{A}^{(1)}$; et la matrice positive $Q^{(2)}_{e_2^+}$ reçoit la valeur 1 dans la position 01 pour représenter la possibilité de la transition de l'état 0 vers l'état 1 de l'automate $\mathcal{A}^{(2)}$ (cf. FIG. 2.6).

De façon analogue, on met en place les transitions globales pour l'événement synchronisant e_3. Néanmoins, cet événement possède des probabilités de routage, qui doivent apparaître dans les matrices. Ainsi, les probabilités π_1 et π_2 apparaissent dans les positions des matrices équivalentes aux transitions qui causent le changement d'état de l'automate.

L'exemple présenté dans FIG. 2.6 est un petit modèle SAN, où les deux événements synchronisants e_2 et e_3 apparaissent dans tous les automates du modèle. Pour des exemples plus complexes, où un

événement synchronisant n'apparaît pas dans tous les automates, les matrices positives et négatives de cet événement sont égales à matrices *identités*, *i.e.*, des matrices qui conservent l'état de l'automate pour les automates non concernés par l'occurrence de l'événement.

La somme de toutes les matrices positives des événements synchronisants représente la matrice positive de la partie synchronisante du descripteur Markovien, *i.e.*, pour cet exemple, la partie positive du descripteur est représentée par la matrice Q_{e^+} qui est la somme des matrices positives $Q_{e_2^+}$ et $Q_{e_3^+}$ des événements synchronisants e_2 et e_3 respectivement.

$$Q_{e^+} = Q_{e_2^+} + Q_{e_3^+} = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & \beta \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \mu\pi_1 & 0 & \mu\pi_2 & 0 & 0 & 0 \end{array}\right)$$

Les matrices négatives des événements sont obtenues de façon à ce que la somme des matrices positives et négatives soit des matrices dont la somme des éléments de chaque ligne soit égale à 0. Les matrices négatives possèdent des éléments non nuls sur les positions diagonales. Les matrices négatives pour les événements synchronisants e_2 et e_3 sont :

$$Q_{e_2^-} = Q_{e_2^-}^{(1)} \underset{g}{\otimes} Q_{e_2^-}^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\beta & 0 \\ 0 & 0 & 0 \end{pmatrix} \bigotimes_g \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & -\beta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

$$Q_{e_3^-} = Q_{e_3^-}^{(1)} \underset{g}{\otimes} Q_{e_3^-}^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\mu \end{pmatrix} \bigotimes_g \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\mu \end{array}\right)$$

La partie négative est aussi obtenue par la somme de toutes les matrices négatives des événements synchronisants. Donc, la partie négative du descripteur est représenté par la matrice Q_{e^-} :

$$Q_{e^-} = Q_{e_2^-} + Q_{e_3^-} = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & -\beta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\mu \end{array}\right)$$

Le descripteur Markovien est représenté par deux parties : locale et synchronisante. La partie locale Q_l représente les transitions locales et la partie synchronisante, divisée en la partie positive Q_{e^+} et la

partie négative Q_{e^-}, représente les transitions des événements synchronisants. Donc, le descripteur Q qui représente un modèle SAN est obtenu par la somme de ses parties :

$$Q = Q_l + Q_{e^+} + Q_{e^-} = \left(\begin{array}{cc|cc|cc} -\alpha & 0 & \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & -\beta & 0 & 0 & \beta \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \mu\pi_1 & 0 & \mu\pi_2 & 0 & 0 & -\mu \end{array}\right)$$

Comme le descripteur Markovien sera représenté dans un format tensoriel, il est important de remarquer que seulement les matrices locales $Q_l^{(i)}$ et les matrices positives $Q_{e^+}^{(i)}$ et négatives $Q_{e^-}^{(i)}$ des événements synchronisants sont stockées. Ainsi, seulement les petites matrices de taille n_i sont manipulées en permettant d'avoir une représentation compacte et structurée du modèle. Généralement, le descripteur Markovien Q n'est pas représenté de façon entière, sachant qu'il y a des méthodes numériques performantes pour utiliser le descripteur dans un format tensoriel [55, 10].

2.2 Description formelle des SAN à temps continu

Dans cette section, on va présenter d'une façon formelle les notations et définitions de base utilisés tout au long de cette thèse pour le formalisme SAN. Dans ce qui suit, on présente les restrictions imposées à ces définitions précisant le concept de SAN *bien défini*, ainsi que la génération du descripteur Markovien obtenu à partir d'un modèle SAN.

2.2.1 Définitions de base

On va considérer dans cette thèse la formalisation d'un modèle SAN comprenant N automates et E événements synchronisants.

✍ **Soit**

$\mathcal{A}$	l'ensemble des automates ($\|\mathcal{A}\| = N$)[6] ;
$\mathcal{E}$	l'ensemble des événements ;
$\mathcal{F}$	la fonction d'atteignabilité.

L'ensemble d'automates $\mathcal{A}$ comprend N automates nommés $\mathcal{A}^{(i)}$, où $i \in [1..N]$. Tout au long de cette thèse, on adopte la notation $[i..j]$ pour le sous-ensemble de $\mathbb{N}$ (des valeurs discrètes) contenant toutes les valeurs de i jusqu'à j (ces valeurs incluses) et $[i, j]$ pour le sous-ensemble de $\mathbb{R}$ (des valeurs continues) contenant toutes les valeurs de i jusqu'à j (ces valeurs incluses).

[6] On adopte la notation $|\mathcal{X}|$ afin de définir la cardinalité d'un ensemble $\mathcal{X}$.

Définition 2.2.1. *$\mathcal{S}^{(i)}$ est l'ensemble des états (locaux) de l'automate $\mathcal{A}^{(i)}$.*

Définition 2.2.2. *L'espace d'états produit (potentiels) $\hat{\mathcal{S}}$ d'un modèle SAN est défini par le produit cartésien des espaces d'états $\mathcal{S}^{(i)}$, i.e., $\hat{\mathcal{S}} = \prod_{i=1}^{N} \mathcal{S}^{(i)}$*

✍ **Soit**

$x^{(i)}$	l'état local de l'automate $\mathcal{A}^{(i)}$.

Définition 2.2.3. *L'état global $\tilde{x} = (x^{(1)}, \ldots, x^{(N)})$ d'un modèle SAN est le vecteur des états locaux des N automates, où $\tilde{x} \in \hat{\mathcal{S}}$.*

✍ **Soit**

ω	un ensemble d'indices d'automates, où $\omega \subseteq [1..N]$;
$\tilde{x}^{(\omega)}$	le vecteur des états locaux $x^{(i)}$ tel que $i \in \omega$.

Il est intéressant de remarquer que la définition d'un état local d'un automate ($x^{(i)}$) et la définition d'un état global ($\tilde{x}$) peuvent être vues comme des cas particuliers de $\tilde{x}^{(\omega)}$. Un état local $x^{(i)}$ est le cas où $\omega = \{i\}$, et l'état global $\tilde{x}$ est le cas où $\omega = \{1, 2, 3, \ldots, N\}$.

Définition 2.2.4. *$\hat{\mathcal{S}}^{(\omega)}$ est l'espace d'états produit de l'ensemble des états locaux des automates $\mathcal{A}^{(i)}$, où $i \in \omega$.*

Définition 2.2.5. *Un élément fonctionnel $f(\hat{\mathcal{S}}^{(\omega)})$ est une fonction de $\hat{\mathcal{S}}^{(\omega)} \rightarrow \mathbb{R}^{+}$, où l'ensemble d'indices d'automates $\omega \subseteq [1..N]$.*

Notons que les états $x^{(i)}$, où $i \in \omega$, sont les paramètres d'évaluation pour l'élément fonctionnel $f(\hat{\mathcal{S}}^{(\omega)})$, *i.e.*, l'espace d'états $\hat{\mathcal{S}}^{(\omega)}$ est le domaine de définition de la fonction de l'élément fonctionnel f.

✍ **Soit**

$f(\tilde{x}^{(\omega)})$	l'élément fonctionnel $f(\hat{\mathcal{S}}^{(\omega)})$ évalué pour le vecteur $\tilde{x}^{(\omega)}$.

Définition 2.2.6. *Un élément fonctionnel $f_1(\hat{\mathcal{S}}^{(\omega)})$ est* **identique** *à l'élément fonctionnel $f_2(\hat{\mathcal{S}}^{(\omega)})$, si et seulement si $\forall \tilde{x}^{(\omega)} \in \hat{\mathcal{S}}^{(\omega)}$, $f_1(\tilde{x}^{(\omega)}) = f_2(\tilde{x}^{(\omega)})$.*

Les éléments fonctionnels sont utilisés pour définir des taux et probabilités fonctionnels des événements. Tous les taux et probabilités fonctionnels peuvent être considérés comme des éléments fonctionnels, même ceux qui ont des valeurs constantes. Une telle définition ne représente pas une restriction, sachant que des éléments constants peuvent être vus comme des fonctions constantes et $\omega = \emptyset$. Ainsi, tous les éléments d'un modèle SAN peuvent être considérés comme des fonctions de $\hat{\mathcal{S}} \rightarrow \mathbb{R}^{+}$.

Définition 2.2.7. *Dans un modèle SAN, un événement est défini par un identificateur e, où $e \in \mathcal{E}$.*

Définition 2.2.8. *Un tuple d'événement (e, τ_e) est composé de :*

1. *e, l'identificateur de l'événement ;*
2. *τ_e, l'élément fonctionnel défini de $\hat{\mathcal{S}} \rightarrow \mathbb{R}^{+}$, représentant le taux d'occurrence de l'événement e.*

Les définitions 2.2.7 et 2.2.8 précisent les événements d'un modèle SAN. Notamment, la définition 2.2.7 identifie chaque événement du modèle et la définition 2.2.8 associe un taux d'occurrence τ_e à un événement e.

✍ **Soit**

$\tau_e(\tilde{x})$	le taux d'occurrence de l'événement e exprimé par l'élément fonctionnel τ_e évalué pour l'état global $\tilde{x}$.

Définition 2.2.9. *L'ensemble $\mathcal{T}$ contient tous les tuples de transition (e, π_e). Un tuple de transition (e, π_e) est composé de :*

1. *e, l'identificateur de l'événement ;*
2. *π_e, l'élément fonctionnel défini de $\hat{\mathcal{S}} \rightarrow [0, 1]$, représentant la probabilité de routage d'une transition lors de l'occurrence de l'événement e.*

L'ensemble $\mathcal{T}$ contient au moins un tuple de transition pour chaque événement e de l'ensemble des événements $\mathcal{E}$. On utilise $\mathcal{T}^*$ pour définir que $\mathcal{T}^* = \mathcal{T} \cup \{\emptyset\}$.

Définition 2.2.10. *$\mathcal{Q}^{(i)}$ est la fonction de transition de $\mathcal{S}^{(i)} \times \mathcal{S}^{(i)} \rightarrow \mathcal{T}^*$, qui contient les tuples de transition de l'automate $\mathcal{A}^{(i)}$.*

Définition 2.2.11. *$\tilde{\mathcal{Q}}$ est la fonction de transition de $\hat{\mathcal{S}} \times \hat{\mathcal{S}} \rightarrow \mathcal{T}^*$, qui contient les tuples de transition de l'automate global.*

✍ **Soit**

$\mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$	la fonction de transition de l'état local $x^{(i)}$ vers l'état local $y^{(i)}$, qui contient une liste de tuples de transition (e, π_e) dans $\mathcal{T}^*$;
$\tilde{\mathcal{Q}}(\tilde{x}, \tilde{y})$	la fonction de transition de l'état global $\tilde{x}$ vers l'état global $\tilde{y}$, qui contient une liste de tuples de transition (e, π_e) dans $\mathcal{T}^*$;

La fonction de transition $\mathcal{Q}^{(i)}$ d'un automate $\mathcal{A}^{(i)}$ (Définition 2.2.10) indique la relation entre les états de l'automate et les événements qui peuvent être tirés. Cette relation est décrite par les tuples de transition (e, π_e) qui composent l'ensemble de tuples de transition $\mathcal{T}$ (Définition 2.2.9). Chaque tuple de transition définit, outre l'identificateur, la probabilité de l'événement pour une transition. Le nombre de tuples de transition associé à une même transition est égal au nombre d'événements qui peuvent déclencher cette transition.

✍ **Soit** : Etant donné $e \in \mathcal{E}$,

$\mathcal{O}^{(e)}$	l'ensemble d'indices i ($i \in [1..N]$) tel que l'automate $\mathcal{A}^{(i)}$ contienne au moins un tuple de transition avec l'identificateur de l'événement e dans un élément de $\mathcal{Q}^{(i)}$;
$\iota^{(e)}$	l'indice minimum de l'ensemble $\mathcal{O}^{(e)}$, *i.e.*, l'indice le plus petit des automates qui possèdent au moins un tuple de transition de l'événement e.

Définition 2.2.12. *Un événement e est classé comme :*

1. *événement local, si $\|\mathcal{O}^{(e)}\| = 1$;*
2. *événement synchronisant, si $\|\mathcal{O}^{(e)}\| > 1$.*

Définition 2.2.13. *L'ensemble des événements locaux $\mathcal{E}_l$ est défini comme $\mathcal{E}_l = \{e \in \mathcal{E}$ tel que $|\mathcal{O}^{(e)}| = 1\}$.*

Définition 2.2.14. *L'ensemble des événements synchronisants $\mathcal{E}_s$ est défini comme $\mathcal{E}_s = \{e \in \mathcal{E}$ tel que $|\mathcal{O}^{(e)}| > 1\}$. On note $|\mathcal{E}_s| = E$.*

Définition 2.2.15. *L'ensemble des événements $\mathcal{E}$ est défini comme $\mathcal{E} = \mathcal{E}_l \cup \mathcal{E}_s$ et $\mathcal{E}_l \cap \mathcal{E}_s = \emptyset$.*

La définition 2.2.12 classe chaque événement qui peut être un *événement local* ou un *événement synchronisant*. Il est intéressant de remarquer qu'on n'a pas besoin de classer un événement pour le définir, vu que ce classement est seulement utilisé lors de la création des tenseurs du *Descripteur Markovien* (Section 2.2.3, page 30), qui est composé par une partie locale (événements locaux) et d'autre partie synchronisante (événements synchronisants).

Définition 2.2.16. *Un automate $\mathcal{A}^{(i)}$ est défini par :*

1. *un ensemble d'états $\mathcal{S}^{(i)}$;*
2. *une fonction de transition $\mathcal{Q}^{(i)}$.*

✍ **Soit**

$\pi_e(x^{(i)}, y^{(i)})$	la probabilité de routage associée au tuple de transition (e, π_e) dans $\mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$;
$\pi_e(x^{(i)}, y^{(i)})(\tilde{x})$	la probabilité de routage associée au tuple de transition (e, π_e) dans $\mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$ évaluée pour l'état global $\tilde{x}$;
$succ_e(x^{(i)})$	l'ensemble des états successeurs $y^{(i)}$ de $x^{(i)}$ tels que $\mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$ possède un tuple de transition avec l'identificateur e et $\tau_e \neq 0$, $\pi_e(x^{(i)}, y^{(i)}) \neq 0$. L'ensemble des états successeurs de l'événement e à partir de $x^{(i)}$ peut être vide, cas où la transition ne peut pas être tirée dans $x^{(i)}$ par l'événement e.

On peut dire qu'un événement synchronisant e est *réalisable* dans l'état global $\tilde{x}$, si et seulement si $\forall i \in \mathcal{O}^{(e)}$ l'ensemble des états successeurs $y^{(i)} \in succ_e(x^{(i)})$ n'est pas vide et $\tau_e(\tilde{x}) \neq 0$.

Définition 2.2.17. *La fonction d'atteignabilité $\mathcal{F}$ est un élément fonctionnel défini de $\hat{\mathcal{S}} \rightarrow [0..1]$. La fonction associe aux états globaux $\tilde{x} \in \hat{\mathcal{S}}$ la valeur 1 si ils sont* **atteignables** *et la valeur 0 si ils sont* **non-atteignables**.

Définition 2.2.18. *L'espace d'états atteignables $\mathcal{S}$ est le sous-ensemble de $\hat{\mathcal{S}}$ ($\mathcal{S} \subseteq \hat{\mathcal{S}}$) composé de tous les états globaux $\tilde{x}$ tels que $\mathcal{F}(\tilde{x}) = 1$.*

La fonction d'atteignabilité $\mathcal{F}$ s'évalue pour tous les états globaux $\tilde{x} \in \hat{\mathcal{S}}$ d'un modèle SAN et ainsi détermine quels sont les états atteignables de ce modèle. Donc, il est possible d'obtenir l'espace des états atteignables d'un modèle SAN avec cette fonction.

Définition 2.2.19. *Un modèle SAN composé de N automates et $|\mathcal{E}|$ événements est défini par :*

1. *chacun des événements $e \in \mathcal{E}$, ainsi que son tuple d'événement (e, τ_e) et ses tuples de transition $(e, \pi_e) \in \mathcal{T}$;*
2. *chacun des automates $\mathcal{A}^{(i)}$ ($i \in [1..N]$) et leur fonction de transition ;*
3. *la fonction d'atteignabilité $\mathcal{F}$.*

2.2.2 SAN *bien définis*

Les définitions de ce modèle doivent être non ambiguës. Pour cela quelques restrictions doivent être faites pour assurer la propriété Markovienne de l'automate global. Les modèles SAN respectant ces restrictions sont dit SAN *bien définis*.

Restriction 1 *Un automate $\mathcal{A}^{(i)}$ est bien défini, si et seulement si pour tout $\tilde{x} \in \mathcal{S}$, pour tout $x^{(i)} \in \mathcal{S}^{(i)}$ et pour tout $e \in \mathcal{E}$ tel que $succ_e(x^{(i)})$ n'est pas vide et pour $y^{(i)} \in succ_e(x^{(i)})$:*

1.1. $$\left(\sum_{y^{(i)} \in succ_e(x^{(i)})} \pi_e(x^{(i)}, y^{(i)})(\tilde{x}) \right) = 1$$

La restriction 1.1 impose que la somme des probabilités de toutes les transitions en sortant d'un même état doit être égale à 1.

Restriction 2 *Un événement $e \in \mathcal{E}$ est bien défini, si et seulement si :*

2.1. $\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $y^{(i)} \in succ_e(x^{(i)})$ *et* $(e, \pi_e) \in \mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$
si $\exists (e_1, \pi_{e_1}) \in \mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$ *alors* e_1 *est un événement différent de* e.

La restriction 2.1 imposée aux événements exige que l'identificateur d'un événement doit apparaître *une seule fois* dans la liste de tuples de transition d'un état donné à l'autre.

Restriction 3 *La fonction d'atteignabilité $\mathcal{F}$ est bien définie, si et seulement si l'automate global (la chaîne de Markov équivalente) est représenté par un graphe fortement connexe.*

La troisième restriction assure l'irréductibilité de la chaîne de Markov équivalente au modèle SAN et permet d'employer les théorèmes standards.

Restriction 4 *Un modèle SAN est bien défini, si et seulement si :*

4.1. tous ses automates sont bien définis ;

4.2. tous ses événements sont bien définis ;

4.3. sa fonction d'atteignabilité est bien définie.

2.2.3 Descripteur Markovien

Le *descripteur Markovien* est une formule algébrique qui permet d'écrire de façon compacte le générateur infinitésimal de la chaîne de Markov équivalente à un modèle SAN par le biais d'une formulation mathématique [55, 111]. Cette formulation mathématique décrit le générateur infinitésimal de la chaîne de Markov équivalente au modèle SAN à partir des tenseurs (matrices) de transition de chaque automate.

✍ **Soit**

$Q_l^{(i)}$	la matrice qui réunit toutes les transitions des événements locaux de l'automate $\mathcal{A}^{(i)}$;
$Q_{e^+}^{(i)}$	la matrice de synchronisation positive qui représente l'occurrence de l'événement synchronisant e dans l'automate $\mathcal{A}^{(i)}$;

$Q_{e^-}^{(i)}$	la matrice de synchronisation négative qui représente l'ajustement nécessaire pour $Q_{e^+}^{(i)}$.
$Q_k^{(i)}(x^{(i)}, y^{(i)})$	l'élément de la matrice $Q_k^{(i)}$ de la ligne $x^{(i)}$ et la colonne $y^{(i)}$, où $i \in [1..N]$ et $k = l$ (représentant une matrice locale), $k = e^+$ (représentant une matrice de synchronisation positive), ou $k = e^-$ (représentant une matrice de synchronisation négative) pour tout $e \in \mathcal{E}_s$;
$\lvert\mathcal{S}^{(i)}\rvert$	le nombre d'états de l'espace d'états $\mathcal{S}^{(i)}$;
I_{n_i}	la matrice identité de taille n_i, où $i \in [1..N]$ et $n_i = \lvert\mathcal{S}^{(i)}\rvert$.

Ainsi, pour chaque automate $\mathcal{A}^{(i)}$ est associé : une matrice $Q_l^{(i)}$, regroupant tous les taux des tuples de transition des événements locaux de l'ensemble $\mathcal{E}_l$ et $2|\mathcal{E}_s|$ matrices ($Q_{e^+}^{(i)}$ et $Q_{e^-}^{(i)}$), regroupant tous les taux des tuples de transition des événements synchronisants de l'ensemble $\mathcal{E}_s$.

Définition 2.2.20. *Les éléments de la matrice de transition locale $Q_l^{(i)}$ de l'automate $\mathcal{A}^{(i)}$ sont définis par :*

1. $\forall e \in \mathcal{E}_l$, $\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $y^{(i)} \in succ_e(x^{(i)})$ *et* $x^{(i)} \neq y^{(i)}$
$$Q_l^{(i)}(x^{(i)}, y^{(i)}) = \sum_{e \in \mathcal{E}_l} \tau_e \pi_e(x^{(i)}, y^{(i)})$$
2. $\forall x^{(i)} \in \mathcal{S}^{(i)}$
$$Q_l^{(i)}(x^{(i)}, x^{(i)}) = -\sum_{e \in \mathcal{E}_l} \sum_{y^{(i)} \in succ_e(x^{(i)})} \tau_e \pi_e(x^{(i)}, y^{(i)})$$
3. $\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $y^{(i)} \notin succ_e(x^{(i)})$ *et* $x^{(i)} \neq y^{(i)}$
$$Q_l^{(i)}(x^{(i)}, y^{(i)}) = 0$$

La première partie de la définition 2.2.20 correspond aux éléments non-diagonaux de la matrice de transition locale (taux des événements locaux), tandis que la deuxième partie correspond aux éléments diagonaux (l'ajustement diagonal des taux des événements locaux). La troisième partie définit les éléments nuls de la matrice de transition locale.

Définition 2.2.21. *Les éléments de la matrice de synchronisation positive qui représentent l'occurrence de l'événement synchronisant $e \in \mathcal{E}_s$ de l'automate $\mathcal{A}^{(i)}$ sont définis par :*

1. $\forall i \notin \mathcal{O}^{(e)}$
$$Q_{e^+}^{(i)} = I_{|\mathcal{S}^{(i)}|}$$
2. $\forall x^{(\iota^{(e)})}, y^{(\iota^{(e)})} \in \mathcal{S}^{(\iota^{(e)})}$ *tel que* $y^{(\iota^{(e)})} \in succ_e(x^{(\iota^{(e)})})$
$$Q_{e^+}^{(\iota^{(e)})}(x^{(\iota^{(e)})}, y^{(\iota^{(e)})}) = \tau_e \pi_e(x^{(\iota^{(e)})}, y^{(\iota^{(e)})})$$
3. $\forall i \in \mathcal{O}^{(e)}$ *tel que* $i \neq \iota^{(e)}$, $\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $y^{(i)} \in succ_e(x^{(i)})$
$$Q_{e^+}^{(i)}(x^{(i)}, y^{(i)}) = \pi_e(x^{(i)}, y^{(i)})$$
4. $\forall i \in \mathcal{O}^{(e)}$, $\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $y^{(i)} \notin succ_e(x^{(i)})$
$$Q_{e^+}^{(i)}(x^{(i)}, y^{(i)}) = 0$$

La première partie de la définition 2.2.21 correspond aux matrices des automates non concernés par l'événement synchronisant e. La deuxième partie définit les éléments non-nuls (diagonaux et non-diagonaux) de la matrice de l'automate de plus petit indice concerné par l'événement e. La troisième partie définit les éléments non-nuls (diagonaux et non-diagonaux) des matrices des automates non concernés par l'événement e. Finalement, la quatrième partie définit les éléments nuls des matrices des automates concernés par l'événement e.

Définition 2.2.22. *Les éléments de la matrice de synchronisation négative qui représentent l'ajustement nécessaire à l'occurrence de l'événement $e \in \mathcal{E}_s$ de l'automate $\mathcal{A}^{(i)}$ sont définis par :*

1. $\forall i \notin \mathcal{O}^{(e)}$
$$Q_{e^-}^{(i)} = I_{|\mathcal{S}^{(i)}|}$$
2. $\forall x^{(\iota^{(e)})} \in \mathcal{S}^{(\iota^{(e)})}$
$$Q_{e^-}^{(\iota^{(e)})}(x^{(\iota^{(e)})}, x^{(\iota^{(e)})}) = - \sum_{y^{(\iota^{(e)})} \in succ_e(x^{(\iota^{(e)})})} \tau_e \pi_e(x^{(\iota^{(e)})}, y^{(\iota^{(e)})})$$
3. $\forall i \in \mathcal{O}^{(e)}$, $i \neq \iota^{(e)}$ *et* $\forall x^{(i)} \in \mathcal{S}^{(i)}$
$$Q_{e^-}^{(i)}(x^{(i)}, x^{(i)}) = \sum_{y^{(i)} \in succ_e(x^{(i)})} \pi_e(x^{(i)}, y^{(i)})$$
4. $\forall i \in \mathcal{O}^{(e)}$, $\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$ *et* $x^{(i)} \neq y^{(i)}$
$$Q_{e^-}^{(i)}(x^{(i)}, y^{(i)}) = 0$$

Ces matrices sont des matrices diagonales, étant donné qu'il s'agit de l'ajustement diagonal des matrices précédentes.

Définition 2.2.23. *Le générateur Markovien Q correspondant à la chaîne de Markov associée à un modèle SAN bien défini est représenté par la formule tensorielle dénommée Descripteur Markovien [54, 110] :*

$$Q = \bigoplus_{i=1}^{N}{}_g \, Q_l^{(i)} + \sum_{e \in \mathcal{E}_s} \left(\bigotimes_{i=1}^{N}{}_g \, Q_{e^+}^{(i)} + \bigotimes_{i=1}^{N}{}_g \, Q_{e^-}^{(i)} \right) \tag{2.1}$$

Étant donné que toute somme tensorielle est équivalente à une somme de produits tensoriels particuliers (voir Annexe A), le *descripteur Markovien* peut être présenté par :

$$Q = \sum_{j=1}^{(N+2|\mathcal{E}_s|)} \bigotimes_{i=1}^{N}{}_g \, Q_j^{(i)}, \tag{2.2}$$

$$\text{où } Q_j^{(i)} = \begin{cases} I_{|\mathcal{S}^{(i)}|} & \text{pour } j \leq N \text{ e } j \neq i \\ Q_l^{(i)} & \text{pour } j \leq N \text{ e } j = i \\ Q_{e^+_{(j-N)}}^{(i)} & \text{pour } N < j \leq (N+ \mid \mathcal{E}_s \mid) \\ Q_{e^-_{(j-(N+|\mathcal{E}_s|))}}^{(i)} & \text{pour } j > (N+ \mid \mathcal{E}_s \mid) \end{cases}$$

TAB. 2.1 représente les tenseurs de transition nécessaires à l'écriture de l'équation (2.2). La partie supérieure du tableau contient les matrices locales des automates et correspond à la somme tensorielle $\bigoplus_{i=1}^{N} Q_l^{(i)}$. La partie inférieure du tableau comprend d'une part les tenseurs qui contiennent les taux d'occurrence des événements synchronisants (e^+) et d'autre part les tenseurs diagonaux qui contiennent l'ajustement de l'occurrence de ces événements (e^-).

$\sum$	N		$Q_l^{(1)} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(2)}\rvert} \underset{g}{\otimes} \cdots \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(N-1)}\rvert} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(N)}\rvert}$ $I_{\lvert\mathcal{S}^{(1)}\rvert} \underset{g}{\otimes} Q_l^{(2)} \underset{g}{\otimes} \cdots \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(N-1)}\rvert} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(N)}\rvert}$ $\vdots$ $I_{\lvert\mathcal{S}^{(1)}\rvert} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(2)}\rvert} \underset{g}{\otimes} \cdots \underset{g}{\otimes} Q_l^{(N-1)} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(N)}\rvert}$ $I_{\lvert\mathcal{S}^{(1)}\rvert} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(2)}\rvert} \underset{g}{\otimes} \cdots \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(N-1)}\rvert} \underset{g}{\otimes} Q_l^{(N)}$
	$2\lvert\mathcal{E}_s\rvert$	e^+	$Q_{e_1^+}^{(1)} \underset{g}{\otimes} Q_{e_1^+}^{(2)} \underset{g}{\otimes} \cdots \underset{g}{\otimes} Q_{e_1^+}^{(N-1)} \underset{g}{\otimes} Q_{e_1^+}^{(N)}$ $\vdots$ $Q_{e_{\lvert\mathcal{E}_s\rvert}^+}^{(1)} \underset{g}{\otimes} Q_{e_{\lvert\mathcal{E}_s\rvert}^+}^{(2)} \underset{g}{\otimes} \cdots \underset{g}{\otimes} Q_{e_{\lvert\mathcal{E}_s\rvert}^+}^{(N-1)} \underset{g}{\otimes} Q_{e_{\lvert\mathcal{E}_s\rvert}^+}^{(N)}$
		e^-	$Q_{e_1^-}^{(1)} \underset{g}{\otimes} Q_{e_1^-}^{(2)} \underset{g}{\otimes} \cdots \underset{g}{\otimes} Q_{e_1^-}^{(N-1)} \underset{g}{\otimes} Q_{e_1^-}^{(N)}$ $\vdots$ $Q_{e_{\lvert\mathcal{E}_s\rvert}^-}^{(1)} \underset{g}{\otimes} Q_{e_{\lvert\mathcal{E}_s\rvert}^-}^{(2)} \underset{g}{\otimes} \cdots \underset{g}{\otimes} Q_{e_{\lvert\mathcal{E}_s\rvert}^-}^{(N-1)} \underset{g}{\otimes} Q_{e_{\lvert\mathcal{E}_s\rvert}^-}^{(N)}$

TAB. 2.1 – Descripteur Markovien

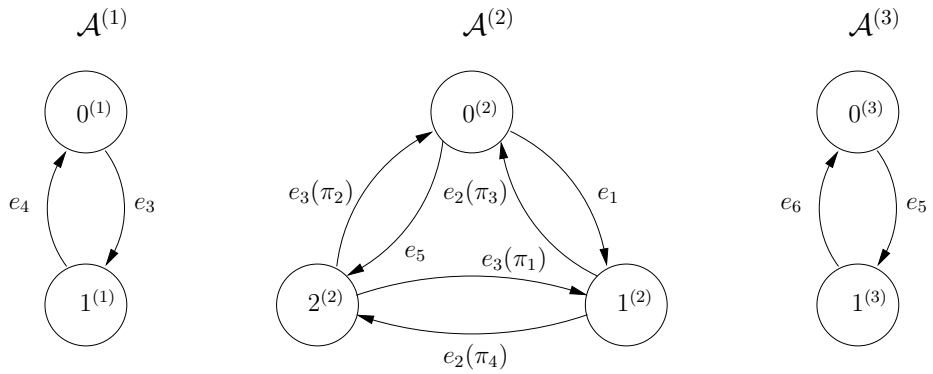
2.2.4 Génération des matrices

Dans cette section, on va présenter un petit exemple afin de montrer l'obtention des matrices du générateur infinitésimal d'un modèle décrit par le formalisme SAN à temps continu. Le déroulement est fait en détail, illustrant pas à pas la génération des tenseurs concernés dans la construction du *descripteur Markovien* du modèle.

2.2.4.1 Description

FIG. 2.8 présente un modèle décrit par le formalisme SAN à temps continu qui possède trois automates, où l'état initial du modèle est $0^{(1)}0^{(2)}0^{(3)}$. L'automate $\mathcal{A}^{(1)}$ possède deux états $0^{(1)}$ et $1^{(1)}$, un événement local e_4 et un événement synchronisant e_3. L'automate $\mathcal{A}^{(2)}$ possède trois états $0^{(2)}$, $1^{(2)}$ et $2^{(2)}$, deux événements locaux e_1 et e_2, et deux événements synchronisants e_3 et e_5. Et le troisième automate ($\mathcal{A}^{(3)}$) possède deux états $0^{(3)}$ et $1^{(3)}$, un événement local e_6 et un événement synchronisant e_5.

Après l'identification des types des événements (locaux et synchronisants), il est possible de générer les tenseurs utilisés pour l'obtention du *descripteur Markovien* de cet exemple.



Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux
loc	e_1	τ_1	syn	e_3	τ_3	syn	e_5	τ_5
loc	e_2	τ_2	loc	e_4	τ_4	loc	e_6	τ_6

FIG. 2.8 – Modèle SAN avec trois automates

2.2.4.2 Tenseurs

Une fois identifié le nombre d'automates ($N = 3$), ainsi que le nombre d'événements synchronisants ($|\mathcal{E}_s|= 2$) du modèle, on peut calculer le nombre de tenseurs (matrices) du *descripteur Markovien* selon l'équation (2.1) :

$$N(1 + 2\,|\mathcal{E}_s|) = 3 \times (1 + 2 \times 2) = 15 \text{ tenseurs}$$

Ensuite, on va démontrer pas à pas l'obtention de tous les tenseurs concernés dans l'équation (2.1). D'abord, on identifie l'ensemble des événements locaux $\mathcal{E}_l = \{e_1, e_2, e_4, e_6\}$ du modèle. Ainsi, on peut construire les matrices locales $Q_l^{(i)}$ correspondants aux automates $\mathcal{A}^{(i)}$:

$Q_l^{(1)} =$

	$0^{(1)}$	$1^{(1)}$
$0^{(1)}$	0	0
$1^{(1)}$	τ_4	-τ_4

$Q_l^{(2)} =$

	$0^{(2)}$	$1^{(2)}$	$2^{(2)}$
$0^{(2)}$	-τ_1	τ_1	0
$1^{(2)}$	$\tau_2\pi_3$	-τ_2	$\tau_2\pi_4$
$2^{(2)}$	0	0	0

$Q_l^{(3)} =$

	$0^{(3)}$	$1^{(3)}$
$0^{(3)}$	0	0
$1^{(3)}$	τ_6	-τ_6

Après cela, on identifie l'ensemble des événements synchronisants $\mathcal{E}_s = \{e_3, e_5\}$. Une fois identifié les événements, on génère les matrices positives qui représentent l'occurrence des événements synchronisants dans chaque automate. Ainsi, on a les matrices positives $Q_{e^+}^{(i)}$ correspondants aux automates $\mathcal{A}^{(i)}$ pour chaque événement :

$$Q^{(1)}_{e_3^+} = \begin{array}{c|cc} & 0^{(1)} & 1^{(1)} \\ \hline 0^{(1)} & 0 & \tau_3 \\ 1^{(1)} & 0 & 0 \end{array} \quad Q^{(2)}_{e_3^+} = \begin{array}{c|ccc} & 0^{(2)} & 1^{(2)} & 2^{(2)} \\ \hline 0^{(2)} & 0 & 0 & 0 \\ 1^{(2)} & 0 & 0 & 0 \\ 2^{(2)} & \pi_2 & \pi_1 & 0 \end{array} \quad Q^{(3)}_{e_3^+} = \begin{array}{c|cc} & 0^{(3)} & 1^{(3)} \\ \hline 0^{(3)} & 1 & 0 \\ 1^{(3)} & 0 & 1 \end{array}$$

$$Q^{(1)}_{e_5^+} = \begin{array}{c|cc} & 0^{(1)} & 1^{(1)} \\ \hline 0^{(1)} & 1 & 0 \\ 1^{(1)} & 0 & 1 \end{array} \quad Q^{(2)}_{e_5^+} = \begin{array}{c|ccc} & 0^{(2)} & 1^{(2)} & 2^{(2)} \\ \hline 0^{(2)} & 0 & 0 & \tau_5 \\ 1^{(2)} & 0 & 0 & 0 \\ 2^{(2)} & 0 & 0 & 0 \end{array} \quad Q^{(3)}_{e_5^+} = \begin{array}{c|cc} & 0^{(3)} & 1^{(3)} \\ \hline 0^{(3)} & 0 & 1 \\ 1^{(3)} & 0 & 0 \end{array}$$

Il est intéressant d'observer que l'automate $\mathcal{A}^{(1)}$ est le premier automate où l'événement synchronisant e_3 est référencé. Toutefois, pour l'événement synchronisant e_5, le premier automate concerné par l'événement est l'automate $\mathcal{A}^{(2)}$. Le taux d'occurrence de chacun de ces événements est présent dans la matrice du premier automate concerné.

Finalement, on construit les matrices négatives qui représentent l'ajustement nécessaire. Donc, on a les matrices négatives $Q^{(i)}_{e^-}$ correspondant aux automates $\mathcal{A}^{(i)}$:

$$Q^{(1)}_{e_3^-} = \begin{array}{c|cc} & 0^{(1)} & 1^{(1)} \\ \hline 0^{(1)} & -\tau_3 & 0 \\ 1^{(1)} & 0 & 0 \end{array} \quad Q^{(2)}_{e_3^-} = \begin{array}{c|ccc} & 0^{(2)} & 1^{(2)} & 2^{(2)} \\ \hline 0^{(2)} & 0 & 0 & 0 \\ 1^{(2)} & 0 & 0 & 0 \\ 2^{(2)} & 0 & 0 & 1 \end{array} \quad Q^{(3)}_{e_3^-} = \begin{array}{c|cc} & 0^{(3)} & 1^{(3)} \\ \hline 0^{(3)} & 1 & 0 \\ 1^{(3)} & 0 & 1 \end{array}$$

$$Q^{(1)}_{e_5^-} = \begin{array}{c|cc} & 0^{(1)} & 1^{(1)} \\ \hline 0^{(1)} & 1 & 0 \\ 1^{(1)} & 0 & 1 \end{array} \quad Q^{(2)}_{e_5^-} = \begin{array}{c|ccc} & 0^{(2)} & 1^{(2)} & 2^{(2)} \\ \hline 0^{(2)} & -\tau_5 & 0 & 0 \\ 1^{(2)} & 0 & 0 & 0 \\ 2^{(2)} & 0 & 0 & 0 \end{array} \quad Q^{(3)}_{e_5^-} = \begin{array}{c|cc} & 0^{(3)} & 1^{(3)} \\ \hline 0^{(3)} & 1 & 0 \\ 1^{(3)} & 0 & 0 \end{array}$$

Une fois obtenu tous les tenseurs du *descripteur Markovien*, on peut finalement construire le générateur infinitésimal Q. Les indices de Q sont composés dans l'ordre lexicographique de la composition des états des automates : $0^{(1)}0^{(2)}0^{(3)}, 0^{(1)}0^{(2)}1^{(3)}, \ldots, 1^{(1)}2^{(2)}1^{(3)}$. Le générateur infinitésimal Q du modèle décrit par le formalisme SAN, obtenu par l'équation (2.2), est présenté dans MAT. 2.1.

MAT. 2.1 Descripteur Markovien

$$
Q = \left(\begin{array}{cccccccccccccccc}
-(\tau_1+\tau_5) & 0 & \tau_1 & 0 & 0 & \tau_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\tau_6 & -(\tau_1+\tau_6) & 0 & \tau_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
\tau_2\pi_3 & 0 & -\tau_2 & 0 & \tau_2\pi_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \tau_2\pi_3 & \tau_6 & -(\tau_2+\tau_6) & 0 & \tau_2\pi_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & -\tau_3 & 0 & \tau_3\pi_2 & 0 & 0 & 0 & \tau_3\pi_1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \tau_6 & -(\tau_3+\tau_6) & 0 & \tau_3\pi_2 & 0 & 0 & 0 & \tau_3\pi_1 & 0 & 0 & 0 & 0 \\
\hline
\tau_4 & 0 & 0 & 0 & 0 & 0 & -(\tau_1+\tau_4+\tau_5) & 0 & \tau_1 & 0 & 0 & 0 & 0 & 0 & 0 & \tau_5 \\
0 & \tau_4 & 0 & 0 & 0 & 0 & \tau_6 & -(\tau_1+\tau_4+\tau_6) & 0 & \tau_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & \tau_4 & 0 & 0 & 0 & \tau_2\pi_3 & 0 & -(\tau_2+\tau_4) & 0 & 0 & 0 & \tau_2\pi_4 & 0 & 0 & 0 \\
0 & 0 & 0 & \tau_4 & 0 & 0 & 0 & \tau_2\pi_3 & \tau_6 & -(\tau_2+\tau_4+\tau_6) & 0 & 0 & 0 & \tau_2\pi_4 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & \tau_4 & 0 & 0 & 0 & 0 & 0 & -\tau_4 & 0 & 0 & 0 & -(\tau_4+\tau_6) & 0 \\
0 & 0 & 0 & 0 & 0 & \tau_4 & 0 & 0 & 0 & 0 & \tau_6 & -(\tau_4+\tau_6) & 0 & 0 & 0 & 0
\end{array}\right)
$$

2.3 Conclusion

Dans ce chapitre, on a présenté le formalisme SAN à temps continu et ses principales caractéristiques. Le formalisme a comme principal atout une démarche modulaire pour décrire un système. De plus, l'utilisation d'éléments fonctionnels (tels que des taux et probabilités fonctionnels) permet une plus grande flexibilité au formalisme pour l'expression des interactions entre les composants du système.

On a aussi expliqué comment un SAN représente une chaîne de Markov, et comment un modèle SAN peut être représenté de une façon compacte dans ce contexte. Cette façon compacte de représenter un modèle est possible grâce au format tensoriel du générateur de la chaîne de Markov équivalente au modèle SAN.

Dans le chapitre suivant, on va présenter quelques exemples décrits par le formalisme SAN à temps continu. Ces exemples seront utilisés dans cette thèse afin d'illustrer les résultats obtenus pour les méthodes proposées (dans le chapitre 5) pour le formalisme SAN.

Chapitre 3

Exemples de modélisation à temps continu

Dans ce chapitre, on présente quelques exemples de modélisation qui sont utilisés dans cette thèse afin d'illustrer l'utilisation du formalisme des Réseaux d'Automates Stochastiques (SAN - *Stochastic Automata Networks*). Les exemples sont modélisés avec une échelle de temps continu. De plus, ces exemples sont aussi utilisés pour tester l'efficacité des méthodes présentées dans le chapitre 5.

Ce chapitre présente quatre exemples. Ces exemples ont aussi pour but d'illustrer plusieurs manières de décrire des itérations dans les modèles. Les exemples développés dans ce chapitre sont :

- Dîner des Philosophes (Section 3.1) ;
- Patron de Service Alterné (Section 3.2) ;
- Atelier avec kanbans (Section 3.3) ;
- Partage de Ressources (Section 3.4).

Le premier exemple illustre le dîner des philosophes : un problème classique de systèmes informatiques. Cet exemple concerne l'ordonnancement de processus et l'allocation de ressources à ces derniers. Cet exemple montre une bonne flexibilité du formalisme SAN pour l'utilisation d'événements locaux avec des taux fonctionnels au lieu d'événements synchronisants.

Le deuxième exemple représente un réseau de files d'attente ouvert qui est largement couvert par la littérature. Les files d'attente ont des capacités finies (avec blocage ou perte) et elles peuvent servir les clients avec un patron de service alterné, *i.e.*, le serveur de la file sert les clients suivant différents taux exponentiels.

Le troisième exemple illustre un modèle SAN d'un atelier avec quatre postes de travail où des *kanbans* (*tickets*) sont utilisés pour contrôler le flux de pièces. Cette méthode, déployée à la fin des années 1950 dans les usines Toyota, est mise en place entre deux postes de travail et limite la production du poste amont aux besoins exacts du poste aval [132].

Enfin, le quatrième exemple illustre aussi l'utilisation des éléments fonctionnels dans un modèle sans événements synchronisants. Le partage des ressources est un exemple où l'approche modulaire des SAN apporte un gain réel.

3.1 Dîner des Philosophes

Le problème du dîner des philosophes (FIG. 3.1) est un problème classique de systèmes informatiques. Il concerne l'ordonnancement des processus et l'allocation des ressources à ces derniers. Ce problème a été énoncé par Dijkstra [47].

FIG. 3.1 – Illustration du problème

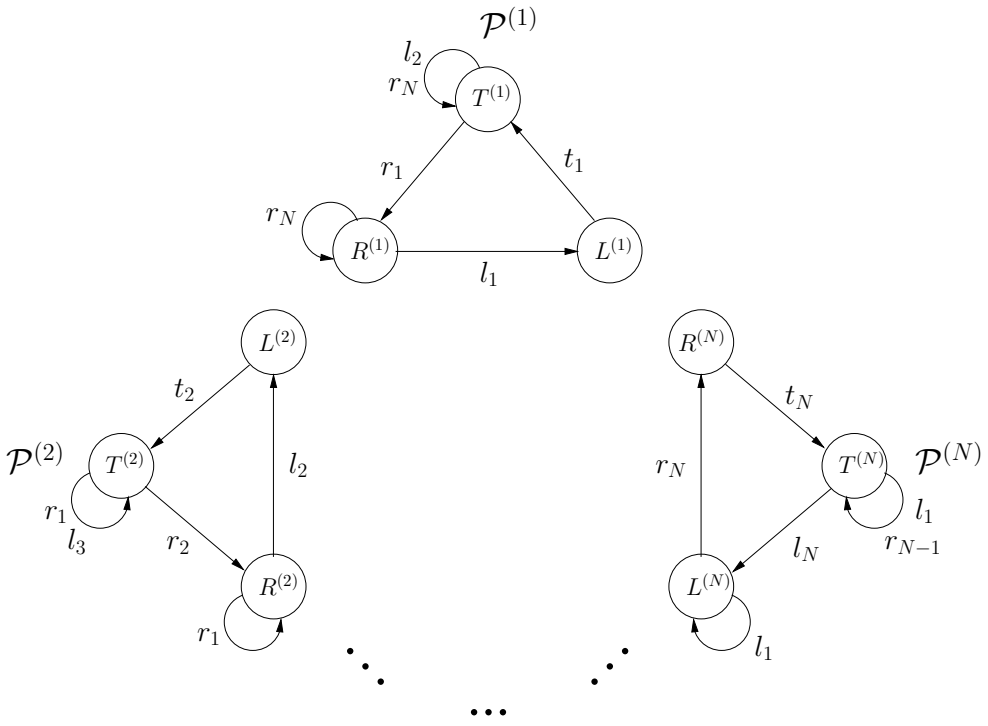
La situation du problème est la suivante : un nombre N de philosophes se trouve autour d'une table ; chaque philosophe a devant lui un plat de spaghetti et une fourchette se trouve à droite de chaque assiette. Quand un philosophe a faim, il va essayer de prendre des fourchettes pour manger. Pour manger, un philosophe a besoin de deux fourchettes : celle qui se trouve à droite de sa propre assiette, et celle qui se trouve à droite de son voisin à gauche, *i.e.*, les fourchettes qui se trouvent autour de sa propre assiette.

Pour construire un modèle SAN de ce problème, on modélise chaque philosophe $\mathcal{P}^{(i)}$ (où $i \in [1..N]$) avec trois états : T (*thinking*) indique que le philosophe pense pendant un temps ; R (*right*) il possède la fourchette qui se trouve à droite de son assiette ; et L (*left*) il possède la fourchette qui se trouve à gauche de son assiette.

Dans FIG. 3.2, on présente le modèle SAN pour le problème du dîner des philosophes, en utilisant des événements (locaux et synchronisants) avec des taux constants.

Le problème consiste à trouver un ordonnancement des philosophes tel qu'ils puissent tous manger, chacun à leur tour. Dans FIG. 3.2, un philosophe $\mathcal{P}^{(i)}$ reste pensif (l'état $T^{(i)}$) pendant un temps et quand il a faim, il va essayer de prendre la fourchette qui se trouve à droite de son assiette (l'état $R^{(i)}$). Alors, pour prendre sa fourchette droite, il faut que la fourchette gauche de son voisin à droite soit disponible, *i.e.*, que son voisin de droite[1] (le philosophe $\mathcal{P}^{(i+1)}$) n'utilise pas sa fourchette gauche (l'état $L^{(i+1)}$).

[1]Le philosophe $\mathcal{P}^{(N)}$ se trouve à gauche de $\mathcal{P}^{(1)}$, ainsi que le voisin à droite du philosophe $\mathcal{P}^{(N)}$ est le philosophe $\mathcal{P}^{(1)}$.



Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux
syn	r_1	α_1	syn	l_1	β_1	loc	t_1	μ_1
syn	r_2	α_2	syn	l_2	β_2	loc	t_2	μ_2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
syn	r_N	α_N	syn	l_N	β_N	loc	t_N	μ_N

FIG. 3.2 – Dîner des Philosophes - modèle SAN (taux constants)

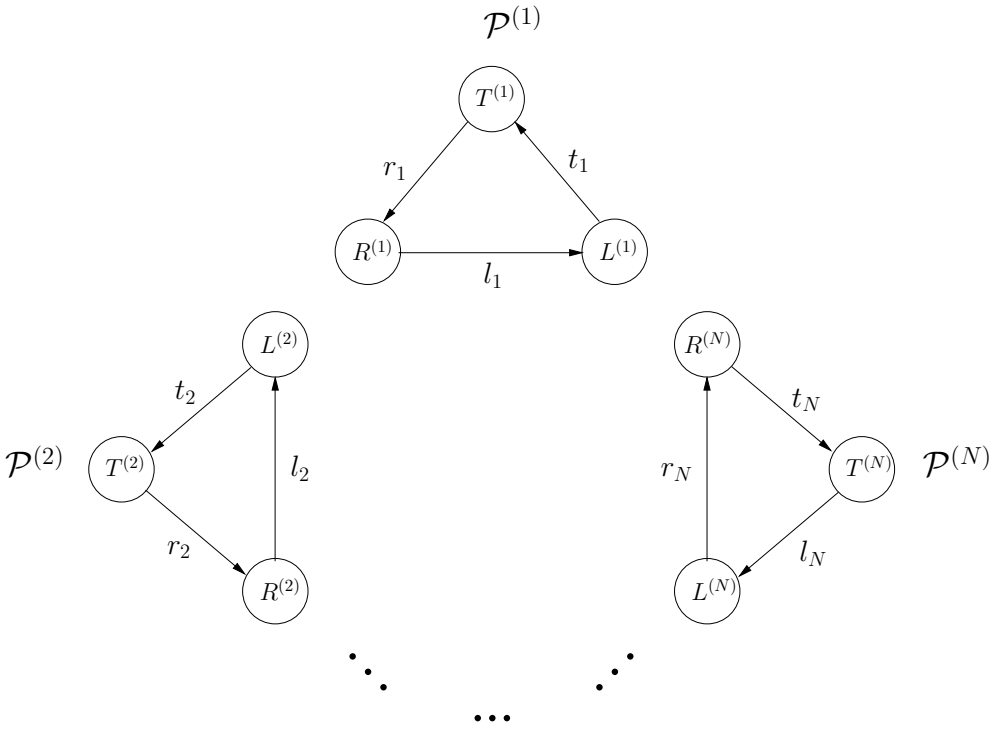
Celui-ci doit se trouver soit dans l'état $T^{(i+1)}$, soit dans l'état $R^{(i+1)}$. Le philosophe $\mathcal{P}^{(i)}$ change de l'état $T^{(i)}$ vers l'état $R^{(i)}$ (*i.e.*, il prend sa fourchette droite) par le tirage de l'événement synchronisant r_i qui synchronise $\mathcal{P}^{(i)}$ et $\mathcal{P}^{(i+1)}$.

Pour manger effectivement, le philosophe $\mathcal{P}^{(i)}$ a besoin de la deuxième fourchette, la fourchette qui se trouve à gauche de son assiette. Dans FIG. 3.2, le philosophe $\mathcal{P}^{(i)}$ prend la fourchette gauche quand il change de l'état $R^{(i)}$ vers l'état $L^{(i)}$ par le tirage de l'événement synchronisant l_i qui synchronise $\mathcal{P}^{(i)}$ et $\mathcal{P}^{(i-1)}$. Pour prendre sa fourchette gauche, il faut que son voisin à gauche (le philosophe $\mathcal{P}^{(i-1)}$) se trouve dans l'état $T^{(i-1)}$, *i.e.*, que le philosophe $\mathcal{P}^{(i-1)}$ ne mange pas et par conséquence qu'il n'utilise pas sa fourchette droite (l'état $R^{(i-1)}$). Une fois que le philosophe a mangé, il revient à l'état $T^{(i)}$, *i.e.*, il change de l'état $L^{(i)}$ vers l'état $T^{(i)}$ par le tirage de l'événement local t_i.

Il est intéressant de remarquer que le dernier philosophe ($\mathcal{P}^{(N)}$) est gaucher, *i.e.*, il va essayer de prendre sa fourchette gauche avant sa fourchette droite, afin d'éviter une situation d'interblocage.

Ce problème du dîner des philosophes peut aussi être modélisé en utilisant seulement des événements locaux et de taux fonctionnel. Dans FIG. 3.3, on présente ce modèle SAN, où les événements synchronisants (r_i et l_i) du modèle SAN présenté dans FIG. 3.2 sont remplacés par des événements lo-

caux. Toutefois, les événement locaux r_i et l_i dans FIG. 3.3 se produisent avec un taux exprimé par les fonctions fr_i et fl_i respectivement.



Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux
loc	r_1	fr_1	loc	l_1	fl_1	loc	t_1	μ_1
loc	r_2	fr_2	loc	l_2	fl_2	loc	t_2	μ_2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
loc	r_N	fr_N	loc	l_N	fl_N	loc	t_N	μ_N

$$fr_1 = (st(\mathcal{P}^{(2)})! = L^{(2)}) \times \alpha_1 \qquad fl_1 = (st(\mathcal{P}^{(N)})! = R^{(N)}) \times \beta_1$$
$$fr_2 = (st(\mathcal{P}^{(3)})! = L^{(3)}) \times \alpha_2 \qquad fl_2 = (st(\mathcal{P}^{(1)}) == T^{(1)}) \times \beta_2$$
$$\vdots \qquad\qquad \vdots$$
$$fr_N = (st(\mathcal{P}^{(1)})! = L^{(1)}) \times \alpha_N \qquad fl_N = (st(\mathcal{P}^{(N-1)}) == T^{(N-1)}) \times \beta_N$$

FIG. 3.3 – Dîner des Philosophes - modèle SAN (taux fonctionnels)

Dans le modèle SAN présenté dans FIG. 3.3, le changement d'état du philosophe $\mathcal{P}^{(i)}$ nécessite l'évaluation des fonctions fr_i et fl_i associées aux taux d'occurrence des événements locaux r_i et l_i respectivement, où les arguments de ces fonctions sont les états des philosophes voisins, *i.e.*, les états des philosophes $\mathcal{P}^{(i+1)}$ et $\mathcal{P}^{(i-1)}$.

3.2 Patron de Service Alterné

Cet exemple décrit un petit réseau de files d'attente ouvert avec quatre files (Q_1, Q_2, Q_3 et Q_4) de capacité (finie) K_1, K_2, K_3 et K_4 respectivement. Dans FIG. 3.4, on présente le routage des clients dans cet exemple, où les clients arrivent dans $Q1$ et Q_2 avec des taux constants λ_1 et λ_2 respectivement. Le départ d'un client de Q_1 vers Q_3 est possible si et seulement il y a de la place disponible dans Q_3 (mécanisme de *blocage*). Le départ d'un client de Q_2 vers Q_3 est possible s'il y a de place disponible dans Q_3 ou, autrement, le client est perdu si la file est pleine (mécanisme de *perte*). Le mécanisme de blocage est aussi utilisé pour les clients qui sortent de Q_3 vers Q_4. Les clients sortent de Q_4 vers l'extérieur du système.

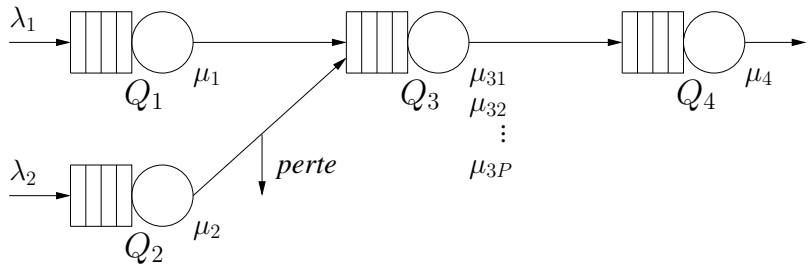
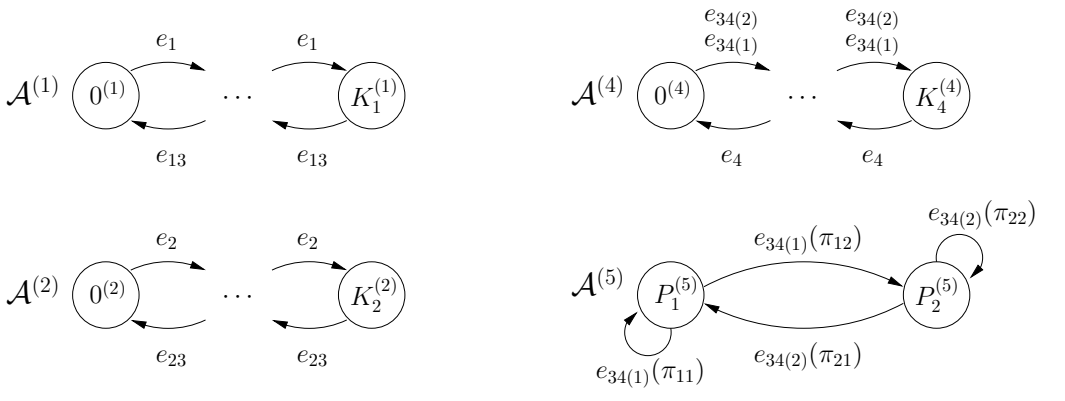


FIG. 3.4 – Patron de Service Alterné (ASP)

Les stations des files Q_1, Q_2 et Q_4 servent les clients avec un taux exponentiel μ_1, μ_2 et μ_4 respectivement. Toutefois, le serveur de la file Q_3 sert les clients avec un *patron de service alterné* (ASP - *Alternate Service Pattern*), *i.e.*, le taux de service de Q_3 change en fonction de P différents patron de service ($\mu_{31}, \ldots, \mu_{3P}$). Lorsque un client est servi par Q_3, la file peut changer son patron de service. Ainsi, quand un client est servi par le patron de service P_i, ce patron de service peut rester pour le client suivant avec une probabilité π_{ii}, ou il peut alterner pour un autre patron de service P_j avec une probabilité π_{ij} (pour tous les patrons de service $P_i : \sum_{j=1}^{P} \pi_{ij} = 1$).

Dans le modèle SAN de cet exemple, chaque file d'attente Q_i est représentée par un automate $\mathcal{A}^{(i)}$ (où $i \in [1..4]$). L'automate $\mathcal{A}^{(5)}$ représente le patron de service courant de la file Q_3. Les événements locaux e_1 et e_2 représentent les arrivées de clients de l'extérieur vers les files Q_1 et Q_2 respectivement, et l'événement local e_4 représente le départ de clients de Q_4 vers l'extérieur. Les événements synchronisants e_{13} et e_{34} représentent le transfert de clients de Q_1 vers Q_3 et Q_3 vers Q_4 respectivement, et l'événement synchronisant e_{23} représente le transfert de clients de Q_2 vers Q_3, et aussi le départ de clients de Q_2 vers l'extérieur (*perte*), lorsque la file Q_3 est pleine.

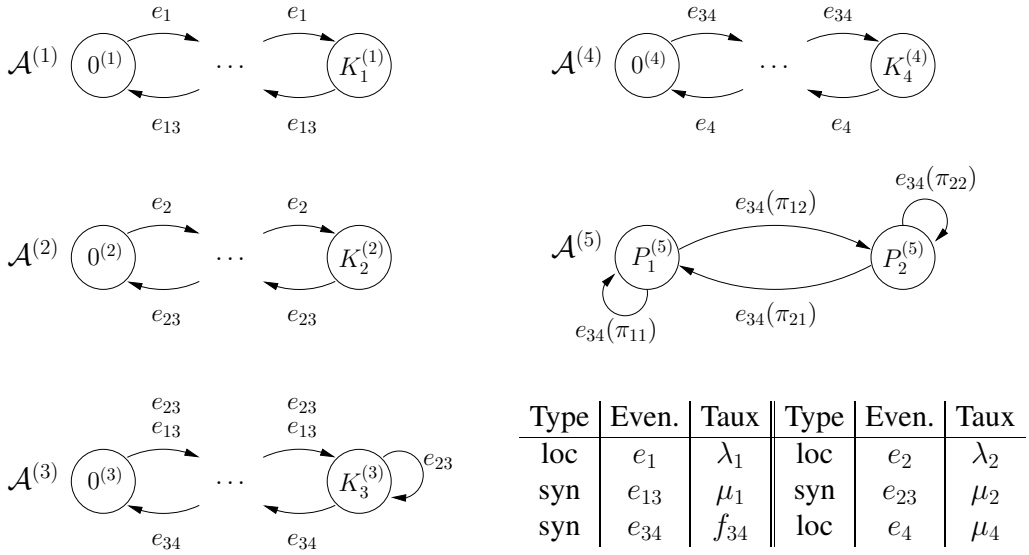
FIG. 3.5 présente le modèle SAN pour cet exemple, avec un nombre de patrons de service $P = 2$. Une extension de ce modèle pour un nombre plus grand de patrons de service correspond à ajouter des états locaux à l'automate $\mathcal{A}^{(5)}$, qui aura toujours P états locaux. Dans FIG. 3.5, tous les événements locaux et synchronisants de ce modèle SAN se produisent avec des taux *constants*. Les événements $e_{34(1)}$ et $e_{34(2)}$ se produisent avec un taux d'occurrence constant égal à μ_{31} et μ_{32} respectivement. Un modèle SAN (avec taux *constants*) de P patron de services aura P événements synchronisants $e_{34(1)}, e_{34(2)}, \ldots, e_{34(P)}$, où chaque événement se produit avec un taux $\mu_{31}, \mu_{32}, \ldots, \mu_{3P}$ respectivement. Il est intéressant de remarquer que l'événement e_{34}, outre qu'il synchronise les automates $\mathcal{A}^{(3)}$ et $\mathcal{A}^{(4)}$, synchronise aussi l'automate $\mathcal{A}^{(5)}$ pour un (possible) échange de patron de service.



Type	Even.	Taux	Type	Even.	Taux
loc	e_1	λ_1	loc	e_2	λ_2
syn	e_{13}	μ_1	syn	e_{23}	μ_2
syn	$e_{34(1)}$	μ_{31}	syn	$e_{34(2)}$	μ_{32}
loc	e_4	μ_4			

FIG. 3.5 – ASP - modèle SAN (taux constants)

Dans FIG. 3.6, on présente un modèle SAN (équivalent au modèle présenté dans FIG. 3.5) avec l'utilisation de fonctions dans les taux d'occurrence des événements. Dans le modèle SAN présenté dans FIG. 3.6, l'événement e_{34} se produit avec un taux exprimé par la fonction f_{34}, qui possède comme arguments les états de l'automate $\mathcal{A}^{(5)}$.



Type	Even.	Taux	Type	Even.	Taux
loc	e_1	λ_1	loc	e_2	λ_2
syn	e_{13}	μ_1	syn	e_{23}	μ_2
syn	e_{34}	f_{34}	loc	e_4	μ_4

$$f_{34} = ((st(\mathcal{A}^{(5)}) == P_1^{(5)}) \times \mu_{31}) + ((st(\mathcal{A}^{(5)}) == P_2^{(5)}) \times \mu_{32})$$

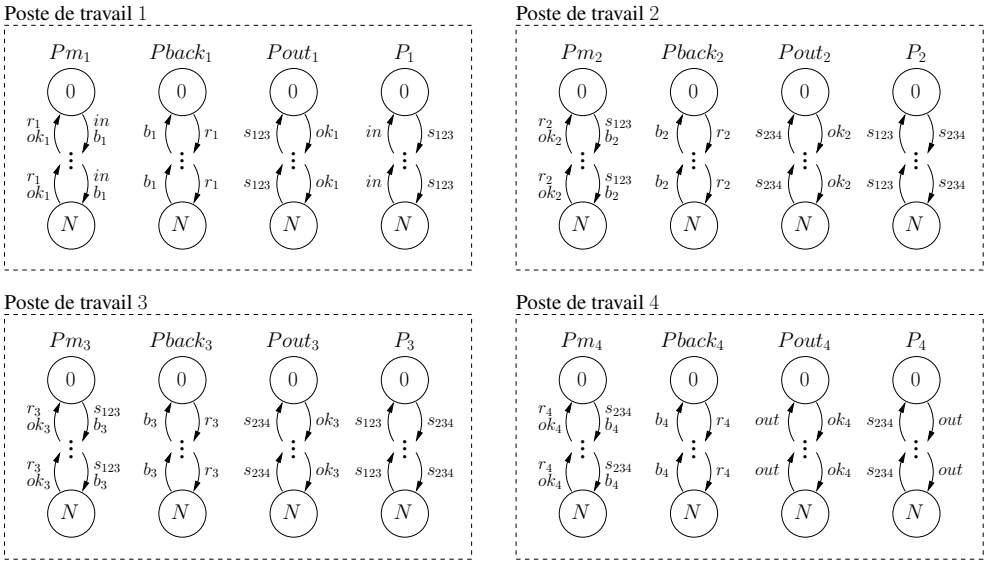
FIG. 3.6 – ASP - modèle SAN (taux fonctionnels)

3.3 Atelier avec kanbans

Cet exemple présente un modèle d'un atelier avec quatre postes de travail où des *kanbans* (*tickets*) sont utilisés pour contrôler le flux de pièces. Le modèle SAN présenté dans FIG. 3.7 a été inspiré du modèle décrit par des *Réseaux de Petri Stochastiques* (SPN - *Stochastic Petri Nets*) et présenté dans [94].

Le flux global de pièces de l'atelier est le suivant : une pièce entre dans le poste de travail 1, après elle est divisée en deux autres pièces qui entrent dans les postes de travail 2 et 3, et ensuite ces deux pièces sont à nouveau assemblées en une seule pièce qui entre dans le poste de travail 4, d'où la pièce sort de l'atelier. Une pièce peut seulement entrer dans un poste de travail s'il y a un *kanban* disponible dans ce poste et, après qu'elle est traitée, elle est examinée afin de savoir si elle doit être retraitée par ce poste. Notamment, une pièce peut seulement sortir du poste de travail 1 s'il y a un *kanban* dans les postes de travail 2 et 3, et une pièce peut seulement sortir du poste de travail 2 (poste de travail 3) s'il y a aussi une pièce prête dans le poste de travail 3 (poste de travail 2) et un *kanban* dans le poste de travail 4.

Dans FIG. 3.7, chaque poste de travail i (où $i \in [1..4]$) de l'atelier est modélisé par quatre automates : Pm_i, indique le nombre de pièces qui sont traitées par le poste simultanément ; $Pback_i$, indique le nombre de pièces qui doivent être retraitées par le poste ; $Pout_i$, indique le nombre de pièces prêtes ; et P_i, indique le nombre de *kanbans* disponibles dans le poste. Chaque automate d'un poste possède $N+1$ états énumérés de 0 à N, où N est le nombre de *kanbans* (*i.e.*, la *capacité*) dans le poste. Dans ce modèle SAN, tous les événements sont synchronisants avec des taux constants. L'état initial du système pour le modèle présenté dans FIG. 3.7 est : pour tout $i \in [1..4]$, l'automate Pm_i est dans l'état 0 ; l'automate $Pback_i$ est dans l'état 0 ; l'automate $Pout_i$ est dans l'état 0 ; et l'automate P_i est dans l'état N.

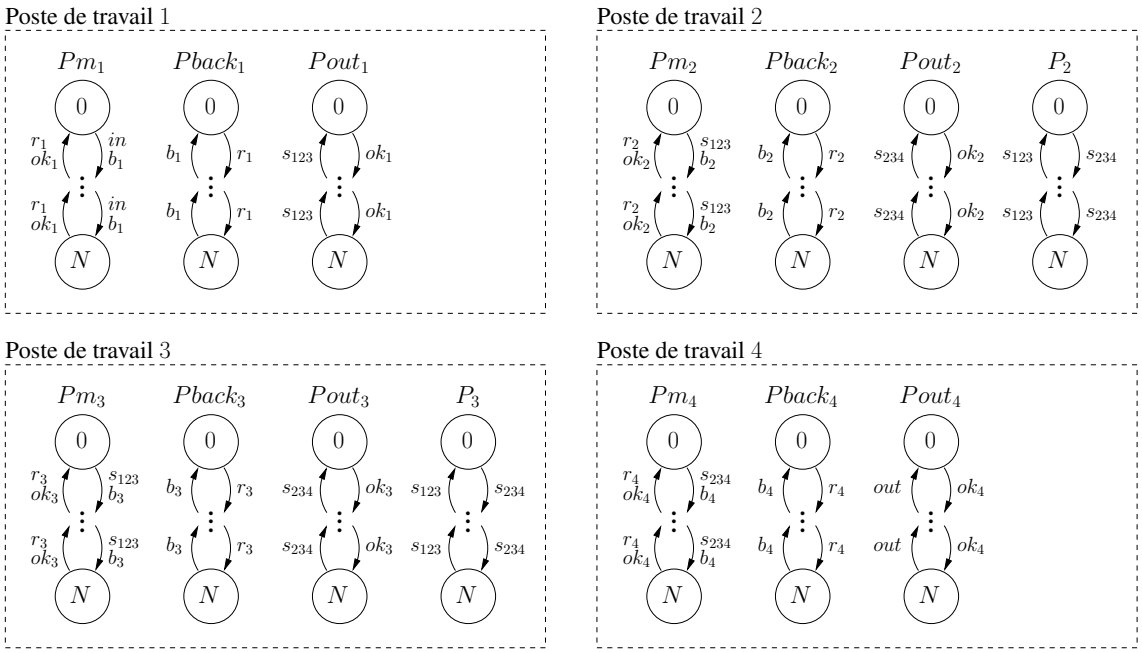


Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux
syn	r_1	α_{r_1}	syn	r_2	α_{r_2}	syn	r_3	α_{r_3}	syn	r_4	α_{r_4}
syn	ok_1	α_{ok_1}	syn	ok_2	α_{ok_2}	syn	ok_3	α_{ok_3}	syn	ok_4	α_{ok_4}
syn	b_1	α_{b_1}	syn	b_2	α_{b_2}	syn	b_3	α_{b_3}	syn	b_4	α_{b_4}
syn	s_{123}	$\alpha_{s_{123}}$	syn	s_{234}	$\alpha_{s_{234}}$				syn	out	α_{out}
syn	in	α_{in}									

FIG. 3.7 – Atelier avec kanbans - modèle SAN (taux constants)

Ce modèle peut aussi être modélisé d'une façon différente, où les automates P_1 et P_4 ne sont plus représentés dans le modèle. Les événements in et out sont modélisés comme des événements locaux, où l'événement in possède un taux *fonctionnel* exprimé par la fonction f_{in}. La fonction f_{in} est utilisée pour assurer que le nombre de *kanbans* du poste de travail 1 n'est pas supérieur à N, *i.e.*, la somme des indices des états des automates Pm_1, $Pback_1$ et $Pout_1$ ne doit pas être supérieure à N.

D'ailleurs, dans ce nouveau modèle, le taux d'occurrence *constant* de l'événement synchronisant s_{234} est remplacé par un taux d'occurrence *fonctionnel* exprimé par la fonction $f_{s_{234}}$. D'une façon analogue à la fonction f_{in}, la fonction $f_{s_{234}}$ est utilisée dans le poste de travail 4. Ce modèle peut être traduit par le SAN présenté dans FIG. 3.8.

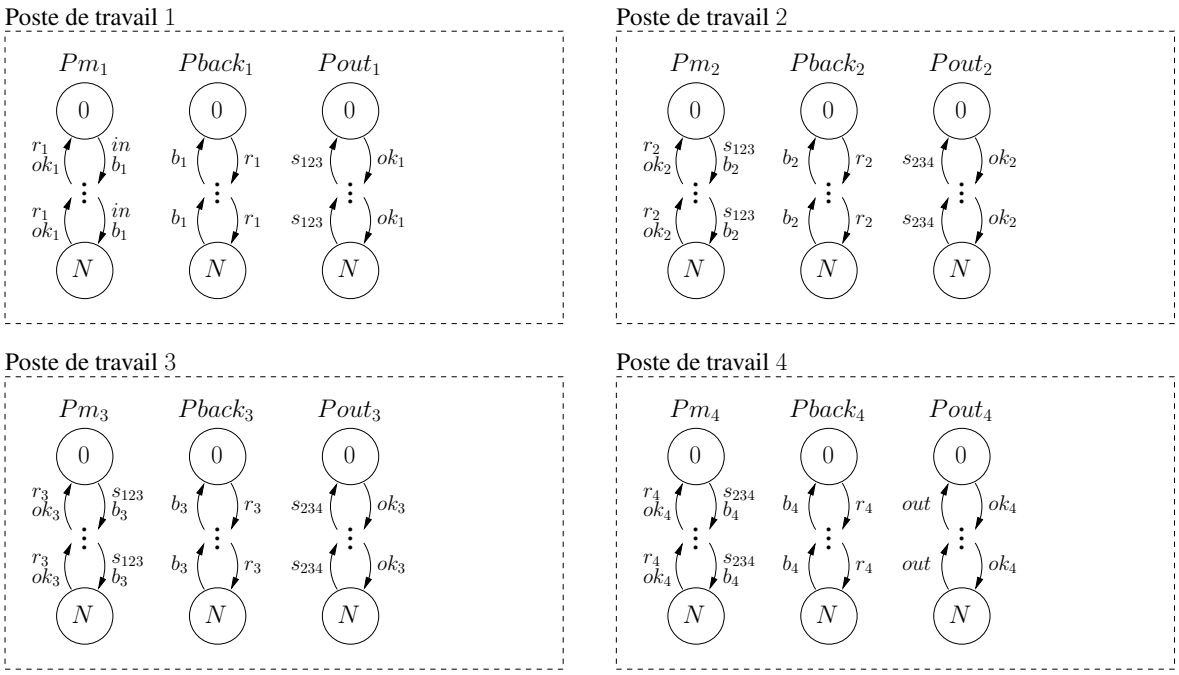


Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux
syn	r_1	α_{r_1}	syn	r_2	α_{r_2}	syn	r_3	α_{r_3}	syn	r_4	α_{r_4}
syn	ok_1	α_{ok_1}	syn	ok_2	α_{ok_2}	syn	ok_3	α_{ok_3}	syn	ok_4	α_{ok_4}
syn	b_1	α_{b_1}	syn	b_2	α_{b_2}	syn	b_3	α_{b_3}	syn	b_4	α_{b_4}
syn	s_{123}	$\alpha_{s_{123}}$	syn	s_{234}	$f_{s_{234}}$				loc	out	α_{out}
loc	in	f_{in}									

$$f_{in} = ((st(Pm_1) + st(Pback_1) + st(Pout_1)) < N) \times \alpha_{in}$$
$$f_{s_{234}} = ((st(Pm_4) + st(Pback_4) + st(Pout_4)) < N) \times \alpha_{s_{234}}$$

FIG. 3.8 – Atelier avec kanbans - modèle SAN (partiellement fonctionnel)

On peut encore modéliser cet exemple de façon à ce que les automates P_2 et P_3 soient éliminés. En utilisant l'idée présentée dans le modèle dans FIG. 3.8, le taux d'occurrence *constant* de l'événement synchronisant s_{123} est remplacé par un taux d'occurrence *fonctionnel* exprimé par la fonction $f_{s_{123}}$. La fonction $f_{s_{123}}$ est utilisée pour assurer que le nombre de *kanbans* des postes de travail 2 et 3 n'est pas supérieur à N. Le modèle SAN pour l'exemple décrit pour ce dernier modèle est présenté dans FIG. 3.9.



Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux
syn	r_1	α_{r_1}	syn	r_2	α_{r_2}	syn	r_3	α_{r_3}	syn	r_4	α_{r_4}
syn	ok_1	α_{ok_1}	syn	ok_2	α_{ok_2}	syn	ok_3	α_{ok_3}	syn	ok_4	α_{ok_4}
syn	b_1	α_{b_1}	syn	b_2	α_{b_2}	syn	b_3	α_{b_3}	syn	b_4	α_{b_4}
syn	s_{123}	$f_{s_{123}}$	syn	s_{234}	$f_{s_{234}}$				loc	out	α_{out}
loc	in	f_{in}									

$$f_{in} = ((st(Pm_1) + st(Pback_1) + st(Pout_1)) < N) \times \alpha_{in}$$

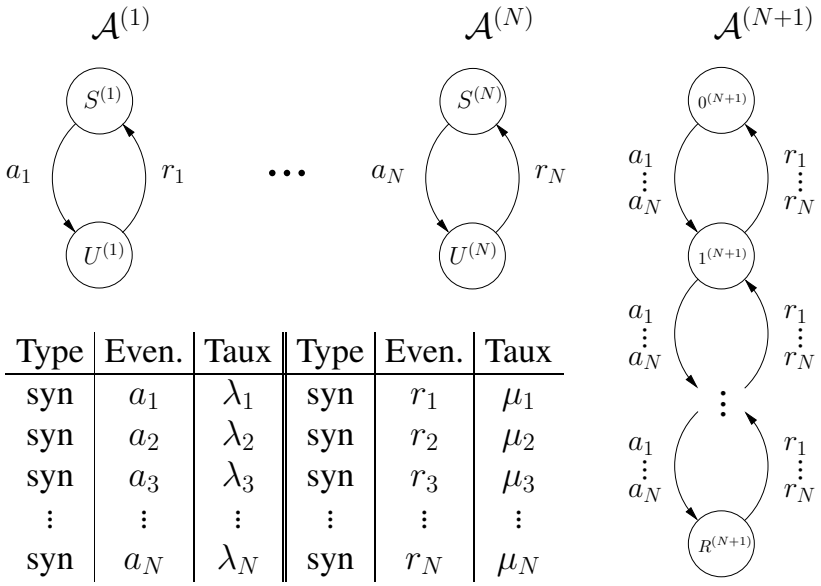
$$f_{s_{234}} = ((st(Pm_4) + st(Pback_4) + st(Pout_4)) < N) \times \alpha_{s_{234}}$$

$$f_{s_{123}} = (((st(Pm_2) + st(Pback_2) + st(Pout_2)) < N) \;\&\&\; ((st(Pm_3) + st(Pback_3) + st(Pout_3)) < N)) \times \alpha_{s_{123}}$$

FIG. 3.9 – Atelier avec kanbans - modèle SAN (taux fonctionnels)

3.4 Partage de Ressources

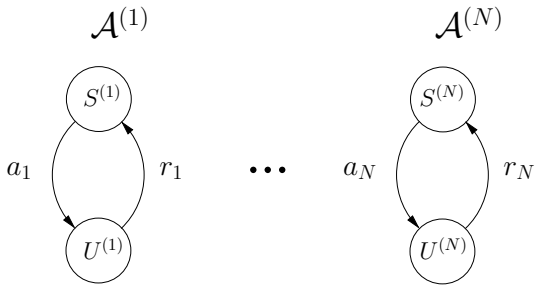
Dans cet exemple N clients distincts se partagent l'utilisation des R ressources communes identiques entre elles. Chacun des clients est représenté par un automate $\mathcal{A}^{(i)}$ (où $i \in [1..N]$), qui possède deux états : $S^{(i)}$ (*sleeping*) état de *repos* ; et $U^{(i)}$ (*using*) état *actif*. Un automate supplémentaire $\mathcal{A}^{(N+1)}$ est utilisé pour représenter l'état de la ressource. Cet automate possède $R + 1$ états, et l'état r ($r \in [0..R]$) signifie que r unités de ressource sont actuellement utilisées. Les événements de prise (a_i) et de relâche (r_i) de ressource sont alors des événements synchronisants entre l'automate du client i ($\mathcal{A}^{(i)}$) et l'automate ressource. En effet, un prise de ressource doit incrémenter l'état de l'automate ressource ($\mathcal{A}^{(N+1)}$), et elle n'est possible que si ce dernier état est différent de R. Au contraire, une relâche de ressource implique un décrément de l'état de l'automate ressource ($\mathcal{A}^{(N+1)}$). Ce modèle, appelé *partage de ressources* (RS - *Resource Sharing*), est illustré dans FIG. 3.10.



Type	Even.	Taux	Type	Even.	Taux
syn	a_1	λ_1	syn	r_1	μ_1
syn	a_2	λ_2	syn	r_2	μ_2
syn	a_3	λ_3	syn	r_3	μ_3
⋮	⋮	⋮	⋮	⋮	⋮
syn	a_N	λ_N	syn	r_N	μ_N

FIG. 3.10 – RS - modèle SAN (taux constants)

Dans FIG. 3.10, tous événements se produisent avec des taux d'occurrence *constants*. Dans FIG. 3.11, on présente un modèle SAN équivalent au modèle SAN présenté dans FIG. 3.10. Ce nouveau modèle SAN (FIG. 3.11) possède seulement des événements locaux. Chaque événement a_i se produit avec un taux exprimé par une fonction f_{u_i} qui contrôle le nombre de ressources utilisées actuellement. Par conséquence, l'automate ressource ($\mathcal{A}^{(N+1)}$) n'existe plus. L'événement a_i se produit seulement s'il y a des ressources disponibles, *i.e.*, si $nb\ [\mathcal{A}^{(1)}..\mathcal{A}^{(N)}]\ U < R$.



Type	Even.	Taux	Type	Even.	Taux
loc	a_1	f_{u_1}	loc	r_1	μ_1
loc	a_2	f_{u_2}	loc	r_2	μ_2
loc	a_3	f_{u_3}	loc	r_3	μ_3
⋮	⋮	⋮	⋮	⋮	⋮
loc	a_N	f_{u_N}	loc	r_N	μ_N

$$f_{u_i} = (nb\ [\mathcal{A}^{(1)}..\mathcal{A}^{(N)}]\ U < R) \times \lambda_i$$

FIG. 3.11 – RS - modèle SAN (taux fonctionnels)

3.5 Conclusion

Dans ce chapitre, on a présenté différents exemples de modélisation à l'aide du formalisme des Réseaux d'Automates Stochastiques (SAN) à temps continu. Nous illustrons à travers ces exemples les différents types d'interaction entre automates, en modélisant certains systèmes de différentes façons.

Les exemples illustrés dans ce chapitre présentent une variété des concepts de modélisation du formalisme SAN suffisante pour les tests ultérieurs.

Chapitre 4

Descripteur d'atteignabilité pour la génération de l'espace d'états atteignables

Pour l'utilisation des méthodes de génération de l'espace d'états atteignables de modèles, il faut tenir compte des structures qui stockent d'une façon performante la relation de transition entre les états du modèle. Une structure qui permet le stockage de ces relations est dénommée *descripteur d'atteignabilité*.

Les matrices qui composent le descripteur d'atteignabilité d'un modèle à temps continu sont basées sur les matrices locales et de synchronisation du descripteur Markovien de ce même modèle. Le descripteur d'atteignabilité est aussi représenté par un format tensoriel généralisé.

Toutefois, comme l'objectif du descripteur d'atteignabilité est de représenter les relations de transition entre les états globaux du modèle (et non à quels taux ils peuvent être atteignables), toutes les informations correspondantes aux taux des événements ne sont plus utilisées. On peut dire que le descripteur d'atteignabilité est un descripteur restreint aux relations d'atteignabilité, sans considérer les taux. En utilisant ce format tensoriel du descripteur d'atteignabilité, il est possible d'obtenir les états atteignables par tirage d'un événement à partir d'un état donné.

Dans ce qui suit, on va donner les notations utilisées pour la représentation d'une matrice globale d'atteignabilité d'un modèle SAN, ainsi que comment obtenir les matrices d'atteignabilité partielles utilisées pour le descripteur. Ensuite, on démontre comment décomposer le descripteur d'atteignabilité dans un format tensoriel afin de l'utiliser pour la génération de l'espace d'états atteignables d'un modèle SAN. Enfin, on présente un exemple d'un petit modèle SAN afin d'illustrer l'obtention du descripteur d'atteignabilité du modèle.

4.1 Matrice globale d'atteignabilité

Une matrice globale d'atteignabilité représente la relation de transition entre les états globaux d'un modèle SAN.

✍ **Rappelons**

$succ_e(\tilde{x})$	l'ensemble des états successeurs $\tilde{y}$ de $\tilde{x}$ tels que $\tilde{\mathcal{Q}}(\tilde{x},\tilde{y})$ possède un tuple de transition avec l'identificateur e et $\tau_e(\tilde{x}) \neq 0$, $\pi_e(\tilde{x},\tilde{y}) \neq 0$. L'ensemble des états successeurs de l'événement e à partir de $\tilde{x}$ peut être vide, cas où la transition ne peut pas être tirée dans $\tilde{x}$ par l'événement e.

Définition 4.1.1. *Une matrice globale d'atteignabilité $\hat{R}$ qui représente la relation de transition entre les états globaux d'un modèle SAN à temps continu a des éléments dans $\mathbb{R}$ qui sont définis par :*

1. $\forall \tilde{x},\tilde{y} \in \hat{\mathcal{S}}$ *tel que* $\tilde{y} \in succ_e(\tilde{x})$ *n'est pas vide,* $e \in \mathcal{E}$
   $\hat{R}(\tilde{x},\tilde{y}) \neq 0$
2. $\forall \tilde{x},\tilde{y} \in \hat{\mathcal{S}}$ *tel que* $\forall e \in \mathcal{E}, \tilde{y} \in succ_e(\tilde{x})$ *est vide*
   $\hat{R}(\tilde{x},\tilde{y}) = 0$

Le but de ce chapitre est de prouver qu'un descripteur d'atteignabilité (défini dans la Section 4.3) d'un modèle SAN à temps continu est une matrice globale d'atteignabilité de ce même modèle. Dans ce qui suit, on va donner les définitions et notations des matrices d'atteignabilité partielles utilisées pour le descripteur d'atteignabilité.

4.2 Matrices d'atteignabilité partielles

Les matrices d'atteignabilité partielles d'un modèle SAN à temps continu sont obtenues à partir des matrices du descripteur Markovien de ce modèle. Une matrice d'atteignabilité partielle représente le relation de transition entre les états *locaux* de chaque automate du modèle. Ainsi, avant de définir comment les matrices d'atteignabilité partielles sont obtenues, on va tenir compte des divers types d'éléments des matrices du descripteur Markovien d'un modèle SAN à temps continu.

✍ **Rappelons**

$Q_k^{(i)}(x^{(i)},y^{(i)})$	l'élément de la matrice $Q_k^{(i)}$ à la ligne $x^{(i)}$ et la colonne $y^{(i)}$, correspondant à l'automate $\mathcal{A}^{(i)}$, où $i \in [1..N]$, avec soit $k = l$ représentant une matrice locale, soit $k = e^+$ représentant une matrice de synchronisation positive pour tout $e \in \mathcal{E}_s$;
$f(\hat{\mathcal{S}}^{(\omega)})$	un élément fonctionnel exprimé par la fonction f qui possède comme l'ensemble d'états d'arguments $\hat{\mathcal{S}}^{(\omega)}$, où $\omega \subseteq [1..N]$.

Définition 4.2.1. *L'élément qui représente la transition de l'état $x^{(i)}$ vers l'état $y^{(i)}$ de l'automate $\mathcal{A}^{(i)}$ (où $i \in [1..N]$) d'une matrice $Q_k^{(i)}$ est dans l'un des cas suivants :*

1. ***nul*** *: $Q_k^{(i)}(x^{(i)},y^{(i)}) = 0$;*
2. ***assimilé-constant*** *: $Q_k^{(i)}(x^{(i)},y^{(i)}) \neq 0$ ou $Q_k^{(i)}(x^{(i)},y^{(i)}) = f(\hat{\mathcal{S}}^{(\omega)})$ tel que le résultat de l'élément fonctionnel $f(\hat{\mathcal{S}}^{(\omega)})$ soit toujours différent de zéro pour tous les états de $\hat{\mathcal{S}}^{(\omega)}$;*
3. ***fonctionnel*** *: $Q_k^{(i)}(x^{(i)},y^{(i)}) = f(\hat{\mathcal{S}}^{(\omega)})$ tel que le résultat de l'élément fonctionnel $f(\hat{\mathcal{S}}^{(\omega)})$ soit égal à zéro au moins pour un état de $\hat{\mathcal{S}}^{(\omega)}$.*

Il est intéressant de remarquer que si le résultat de la fonction d'un élément fonctionnel (taux ou probabilité) d'une matrice est *toujours* différent de *zéro* (*i.e.*, pour tous les états de $\hat{\mathcal{S}}^{(\omega)}$, le résultat évalué est toujours différent de zéro), cet élément fonctionnel peut être vu comme un élément *constant* du point de vue de l'étude de l'atteignabilité car la fonction ne s'annulant pas, l'élément est toujours atteignable. En conséquence, l'élément fonctionnel est équivalent à une valeur constante non-nulle (*e.g.*, la valeur 1), vu que l'objectif d'une matrice d'atteignabilité est de préserver la relation de transition entre les états.

Donc, dans ce chapitre, toutes les fois qu'on parle d'un élément *fonctionnel*, le taux ou la probabilité de cet élément est sûrement exprimée par une fonction dont l'évaluation peut avoir comme résultat la valeur *zéro* pour au moins un état de l'espace d'états de la fonction. Autrement, cet élément peut être traité comme un *constant*.

✍ **Notations**

$\breve{R}_l^{(i)}$	la matrice d'atteignabilité partielle locale qui réunit toutes les transitions des événements locaux de l'automate $\mathcal{A}^{(i)}$, où $i \in [1..N]$;
$\breve{R}_l^{(i)}(x^{(i)}, y^{(i)})$	l'élément de la ligne $x^{(i)}$ et la colonne $y^{(i)}$ de la matrice $\breve{R}_l^{(i)}$ correspondant à l'automate $\mathcal{A}^{(i)}$, où $i \in [1..N]$ et $x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$;
$\breve{R}_e^{(i)}$	la matrice d'atteignabilité partielle de synchronisation (positive) qui indique les transitions de l'événement synchronisant $e \in \mathcal{E}_s$ dans l'automate $\mathcal{A}^{(i)}$, où $i \in [1..N]$;
$\breve{R}_e^{(i)}(x^{(i)}, y^{(i)})$	l'élément de la ligne $x^{(i)}$ et la colonne $y^{(i)}$ de la matrice $\breve{R}_e^{(i)}$ correspondant à l'automate $\mathcal{A}^{(i)}$, où $i \in [1..N]$, $e \in \mathcal{E}_s$ et $x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$.

Toutes les matrices d'atteignabilité partielles d'un modèle SAN sont basées sur les matrices du descripteur Markovien de ce même modèle. Ainsi, les matrices d'atteignabilité partielles locales, ainsi que les matrices d'atteignabilité partielles de synchronisation sont définies à partir des matrices locales et de synchronisation du descripteur Markovien respectivement.

Définition 4.2.2. *Les éléments de la matrice d'atteignabilité partielle locale $\breve{R}_l^{(i)}$ de l'automate $\mathcal{A}^{(i)}$ (où $i \in [1..N]$) sont définis par :*

1. $\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $x^{(i)} \neq y^{(i)}$, *si* $Q_l^{(i)}(x^{(i)}, y^{(i)})$ *est un élément*
   - ***nul***, *alors* $\breve{R}_l^{(i)}(x^{(i)}, y^{(i)}) = 0$
   - ***assimilé-constant***, *alors* $\breve{R}_l^{(i)}(x^{(i)}, y^{(i)}) = 1$
   - ***fonctionnel***, *alors* $\breve{R}_l^{(i)}(x^{(i)}, y^{(i)}) = Q_l^{(i)}(x^{(i)}, y^{(i)})$
2. $\forall x^{(i)} \in \mathcal{S}^{(i)}$
   $\breve{R}_l^{(i)}(x^{(i)}, x^{(i)}) = 0$

La première partie de la définition 4.2.2 correspond aux éléments non-diagonaux de la matrice d'atteignabilité partielle locale, tandis que la deuxième partie correspond aux éléments diagonaux de la matrice (les éléments qui correspondaient aux ajustement des taux des événements). Il est intéressant d'observer que les éléments fonctionnels qui sont définis comme éléments *assimilé-constants* (c.f. Définition 4.2.1) sont remplacés par la valeur constante 1. Ce remplacement est effectué afin d'optimiser le stockage des matrices d'atteignabilité partielles (matrices qui peuvent devenir des matrices booléennes), puisque les valeurs constantes des taux ou probabilités ne sont plus pertinents.

Définition 4.2.3. *Les éléments de la matrice d'atteignabilité partielle de synchronisation $\breve{R}_e^{(i)}$ qui représentent les relations de transition de l'événement synchronisant $e \in \mathcal{E}_s$ de l'automate $\mathcal{A}^{(i)}$ (où $i \in [1..N]$) sont définis par :*

1. $\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $Q_{e^+}^{(i)}(x^{(i)}, y^{(i)})$ *est un élément*
   ***nul**, alors* $\breve{R}_e^{(i)}(x^{(i)}, y^{(i)}) = 0$
   ***assimilé-constant**, alors* $\breve{R}_e^{(i)}(x^{(i)}, y^{(i)}) = 1$
   ***fonctionnel**, alors* $\breve{R}_e^{(i)}(x^{(i)}, y^{(i)}) = Q_{e^+}^{(i)}(x^{(i)}, y^{(i)})$

D'autre part, la définition 4.2.3 ne tient pas compte du remplacement des éléments diagonaux par la valeur 0 (*zéro*), vu que les ajustements pour les événements synchronisants ne sont pas stockés dans les matrices de synchronisation (positives) du descripteur Markovien.

En observant les définitions 4.2.2 et 4.2.3, on constate que les matrices d'atteignabilité partielles (locales ou de synchronisation) ne possèdent que des éléments de valeur 0, 1, ou *fonctionnel*.

Dans ALG. 4.1, on présente l'algorithme pour obtenir, à partir des matrices du descripteur Markovien, les matrices d'atteignabilité partielles utilisées pour le descripteur d'atteignabilité d'un modèle SAN. L'obtention des matrices est réalisée en deux parties. Dans la première partie (lignes 2 - 11), une matrice d'atteignabilité partielle locale est obtenue pour chaque automate du modèle SAN. Dans la deuxième partie (lignes 12 - 21), pour chaque automate, on obtient E (où $E = |\mathcal{E}_s|$) matrices d'atteignabilité partielles de synchronisation correspondantes aux transitions des événements synchronisants pour cet automate.

On remarque qu'un élément $Q^{(i)}(x^{(i)}, y^{(i)})$ est considéré *assimilé-constant* s'il respecte la définition 4.2.1. Pour vérifier cette propriété, il faut évaluer la fonction pour vérifier qu'elle ne s'annule jamais.

4.3 Descripteur d'atteignabilité

Dans cette section, on va présenter des notations de base utilisées pour l'expression d'un descripteur d'atteignabilité d'un modèle SAN à temps continu dont la *formulation tensorielle* est le résultat central de ce chapitre. Cette formulation tensorielle n'est pas quelconque : elle sépare les éléments des matrices suivant les sous-espaces où elles peuvent devenir identiquement nulles. La raison de ceci deviendra claire au Chapitre 5. Nous prenons ceci comme contrainte dans ce chapitre.

De façon analogue au descripteur Markovien d'un modèle SAN, pour chaque automate $\mathcal{A}^{(i)}$ du modèle est associé une matrice d'atteignabilité partielle locale $\breve{R}_l^{(i)}$ (les transitions des événements locaux de l'ensemble $\mathcal{E}_l$) et $|\mathcal{E}_s|$ matrices d'atteignabilité partielles de synchronisation $\breve{R}_e^{(i)}$ (les transitions des événements synchronisants de l'ensemble $\mathcal{E}_s$).

Cependant, l'intérêt principal du descripteur d'atteignabilité est de stocker les relations de transition entre les états du modèle, sans représenter les taux et les ajustements nécessaires à l'occurrence des événements. Ainsi, les éléments diagonaux (éléments négatifs) des matrices d'atteignabilité partielles locales sont des éléments nuls, vu qu'ils n'ont plus besoin de représenter l'ajustement pour les événements locaux. Par conséquent, les matrices d'atteignabilité partielles de synchronisation négatives ne sont pas représentées.

ALG. 4.1 PartialReachabilityMatrices()

1: **for** $i = 1$ to N **do**
2: **for all** $x^{(i)} \in \mathcal{S}^{(i)}$ **do**
3: **for all** $y^{(i)} \in \mathcal{S}^{(i)}$ **do**
4: **if** $(x^{(i)} == y^{(i)})$ **then**
5: $\breve{R}_l^{(i)}(x^{(i)}, y^{(i)}) = 0$; */* L'élément diagonal n'est pas considéré pour les matrices locales */*
6: **else if** ($Q_l^{(i)}(x^{(i)}, y^{(i)})$ est assimilé-constant) **then**
7: $\breve{R}_l^{(i)}(x^{(i)}, y^{(i)}) = 1$; */* Tout élément non-nul ou assimilé-constant est remplacé par 1 */*
8: **else**
9: $\breve{R}_l^{(i)}(x^{(i)}, y^{(i)}) = Q_l^{(i)}(x^{(i)}, y^{(i)})$; */* L'élément fonctionnel est conservé */*
10: **end for**
11: **end for**
12: **for** $e = 1$ to E **do**
13: **for all** $x^{(i)} \in \mathcal{S}^{(i)}$ **do**
14: **for all** $y^{(i)} \in \mathcal{S}^{(i)}$ **do**
15: **if** ($Q_{e^+}^{(i)}(x^{(i)}, y^{(i)})$ est assimilé-constant) **then**
16: $\breve{R}_e^{(i)}(x^{(i)}, y^{(i)}) = 1$; */* Tout élément non-nul ou assimilé-constant est remplacé par 1 */*
17: **else**
18: $\breve{R}_e^{(i)}(x^{(i)}, y^{(i)}) = Q_{e^+}^{(i)}(x^{(i)}, y^{(i)})$; */* L'élément fonctionnel est conservé */*
19: **end for**
20: **end for**
21: **end for**
22: **end for**

Pour l'utilisation des méthodes de génération de l'espace d'états atteignables de modèles SAN (voir Chapitre 5), le descripteur d'atteignabilité doit être représenté par un format tensoriel qui utilise les matrices d'atteignabilité partielles du modèle. Ceci permet d'avoir une fonction "*next-state*" structurée selon l'espace produit.

✍ **Soit**

$f_{l^{(i)},(x^{(i)},y^{(i)})}$ l'élément fonctionnel de la ligne $x^{(i)}$ et la colonne $y^{(i)}$ de la matrice d'atteignabilité partielle locale $\breve{R}_l^{(i)}$ de l'automate $\mathcal{A}^{(i)}$, où $i \in [1..N]$ et $x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$;

$f_{e^{(i)},(x^{(i)},y^{(i)})}$ l'élément fonctionnel de la ligne $x^{(i)}$ et la colonne $y^{(i)}$ de la matrice d'atteignabilité partielle de synchronisation $\breve{R}_e^{(i)}$ de l'événement synchronisant $e \in \mathcal{E}_s$ dans l'automate $\mathcal{A}^{(i)}$, où $i \in [1..N]$ et $x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$;

$f^{-1}(0)$ les états antécédents de la valeur 0 (*zéro*) de la fonction f, *i.e.*, $f^{-1}(0) = \{\tilde{x}^{(\omega)} \in \hat{\mathcal{S}}^{(\omega)} \mid f(\tilde{x}^{(\omega)}) = 0\}$.

On considère que tous les éléments d'une matrice d'atteignabilité partielle peuvent être vus comme des fonctions. Notamment, les éléments nuls et constants sont des fonctions (sans paramètres, *i.e.*, $\omega = \emptyset$) qui ont toujours le même résultat.

Dans les définitions qui suivent, on va définir des relations d'équivalence qui vont classer les fonctions suivant les sous-espaces où elles s'annulent, par type de matrice.

Définition 4.3.1. *Pour les éléments d'une matrice d'atteignabilité partielle locale* $\breve{R}_l^{(i)}$ *de l'automate* $\mathcal{A}^{(i)}$ *(où* $i \in [1..N]$*), on définit une relation d'équivalence* $\mathcal{R}_l^{(i)}$ *tel que* $f_{l^{(i)},(x^{(i)},y^{(i)})} \mathcal{R}_l^{(i)} f_{l^{(i)},(z^{(i)},w^{(i)})} \Leftrightarrow f^{-1}_{l^{(i)},(x^{(i)},y^{(i)})}(0) = f^{-1}_{l^{(i)},(z^{(i)},w^{(i)})}(0)$*, où* $x^{(i)}$*,* $y^{(i)}$*,* $z^{(i)}$*,* $w^{(i)} \in \mathcal{S}^{(i)}$*.*

Définition 4.3.2. *Pour les éléments d'une matrice d'atteignabilité partielle de synchronisation* $\breve{R}_e^{(i)}$ *de l'événement synchronisant* $e \in \mathcal{E}_s$ *dans l'automate* $\mathcal{A}^{(i)}$ *(où* $i \in [1..N]$*), on définit une relation d'équivalence* $\mathcal{R}_e^{(i)}$ *tel que* $f_{e^{(i)},(x^{(i)},y^{(i)})} \mathcal{R}_e^{(i)} f_{e^{(i)},(z^{(i)},w^{(i)})} \Leftrightarrow f^{-1}_{e^{(i)},(x^{(i)},y^{(i)})}(0) = f^{-1}_{e^{(i)},(z^{(i)},w^{(i)})}(0)$*, où* $x^{(i)}$*,* $y^{(i)}$*,* $z^{(i)}$*,* $w^{(i)} \in \mathcal{S}^{(i)}$*.*

En utilisant les relations d'équivalence définies pour les matrices d'atteignabilité partielles, on peut définir des classes d'équivalence sur l'ensemble d'éléments d'une matrice d'atteignabilité partielle (locale ou de synchronisation).

Définition 4.3.3. $C_k^{(i)}$ *est le nombre*[1] *de classes d'équivalence des éléments de la matrice d'atteignabilité partielle* $\breve{R}_k^{(i)}$ *de l'automate* $\mathcal{A}^{(i)}$ *pour la relation d'équivalence* $\mathcal{R}_k^{(i)}$*, où* $i \in [1..N]$ *et pour* $k = l$ *représentant une matrice d'atteignabilité partielle locale, ou bien pour* $k = e$ *représentant une matrice d'atteignabilité partielle de synchronisation pour tout* $e \in \mathcal{E}_s$*.*

Définition 4.3.4. $\mathbb{C}_{k,c}^{(i)}$ *est la classe d'équivalence d'indice* c *(où* $c \in [1..C_k^{(i)}]$*) des éléments de la matrice d'atteignabilité partielle* $\breve{R}_k^{(i)}$ *de l'automate* $\mathcal{A}^{(i)}$ *pour la relation d'équivalence* $\mathcal{R}_k^{(i)}$*, où* $i \in [1..N]$ *et pour* $k = l$ *représentant une matrice d'atteignabilité partielle locale, ou bien pour* $k = e$ *représentant une matrice d'atteignabilité partielle de synchronisation pour tout* $e \in \mathcal{E}_s$*.*

On impose que si une matrice possède seulement des éléments nuls, alors la classe d'équivalence de ces éléments n'est pas considérée. Par exemple, soit $\breve{R}_l^{(1)}$ une matrice d'atteignabilité partielle locale nulle de l'automate $\mathcal{A}^{(1)}$, le nombre de classes d'équivalence des éléments de cette matrice est égal à *zéro*, *i.e.*, $C_l^{(1)} = 0$.

Dans ALG. 4.2, on présente l'algorithme pour obtenir les classes d'équivalence des éléments des matrices d'atteignabilité partielles d'un modèle SAN. L'obtention des classes d'équivalence est réalisée en deux parties : la partie consacrée aux matrices locales (lignes 2 - 23) et la partie consacrée aux matrices synchronisantes (lignes 24 - 47).

Par rapport à la partie locale, pour chaque élément non-nul d'une matrice locale (ligne 5), on vérifie si cet élément se trouve déjà dans une classe d'équivalence existante (lignes 6 - 12). Pour réaliser cette vérification, on vérifie si l'élément est en relation avec un autre élément g d'une classe d'équivalence déjà existante (ligne 9). Si l'élément n'appartient à aucune classe d'équivalence, on crée une nouvelle classe d'équivalence (lignes 14 - 15) et on vérifie quels autres éléments de la matrice sont en relation avec lui (ligne 18). De cette façon, lorsqu'on crée une nouvelle classe d'équivalence, on cherche tous les éléments de la matrice qui appartiennent à cette classe (ligne 13 - 21). Cette démarche est appliquée de façon analogue pour les matrices synchronisantes (lignes 24 - 47).

[1] La classe d'équivalence d'élément *nul* n'est pas considérée.

ALG. 4.2 EquivalenceClasses()

for $i = 1$ to N **do**
  $C_l^{(i)} = 0$;
  for all $x^{(i)} \in \mathcal{S}^{(i)}$ **do**
    for all $y^{(i)} \in \mathcal{S}^{(i)}$ **do**
      if $(f_{l^{(i)},(x^{(i)},y^{(i)})} \neq 0)$ **then**
        NewClass $\leftarrow$ **true** ;
        for $c = 1$ to $C_l^{(i)}$ **do**
          $g \leftarrow Pick(\mathbb{C}_{l,c}^{(i)})$; /* *Prend un élément de* $\mathbb{C}_{l,c}^{(i)}$ */
          if $(f_{l^{(i)},(x^{(i)},y^{(i)})} \mathcal{R}_l^{(i)} g)$ **then**
            NewClass $\leftarrow$ **false** ;
            break ;
        end for
        if (*NewClass*) **then**
          $C_l^{(i)} = C_l^{(i)} + 1$;
          $\mathbb{C}_{l,C_l^{(i)}}^{(i)} \leftarrow \emptyset$;
          for all $z^{(i)} \in \mathcal{S}^{(i)}$ **do**
            for all $w^{(i)} \in \mathcal{S}^{(i)}$ **do**
              if $(f_{l^{(i)},(x^{(i)},y^{(i)})} \mathcal{R}_l^{(i)} f_{l^{(i)},(z^{(i)},w^{(i)})})$ **then**
                $\mathbb{C}_{l,C_l^{(i)}}^{(i)} \leftarrow \mathbb{C}_{l,C_l^{(i)}}^{(i)} \cup \{f_{l^{(i)},(z^{(i)},w^{(i)})}\}$; /* *Ajoute* $f_{l^{(i)},(z^{(i)},w^{(i)})}$ *à classe* $\mathbb{C}_{l,C_l^{(i)}}^{(i)}$ */
            end for
          end for
      end for
    end for
    for $e = 1$ to E **do**
      $C_e^{(i)} = 0$;
      for all $x^{(i)} \in \mathcal{S}^{(i)}$ **do**
        for all $y^{(i)} \in \mathcal{S}^{(i)}$ **do**
          if $(f_{e^{(i)},(x^{(i)},y^{(i)})} \neq 0)$ **then**
            NewClass $\leftarrow$ **true** ;
            for $c = 1$ to $C_e^{(i)}$ **do**
              $g \leftarrow Pick(\mathbb{C}_{e,c}^{(i)})$; /* *Prend un élément de* $\mathbb{C}_{e,c}^{(i)}$ */
              if $(f_{e^{(i)},(x^{(i)},y^{(i)})} \mathcal{R}_e^{(i)} g)$ **then**
                NewClass $\leftarrow$ **false** ;
                break ;
            end for
            if (*NewClass*) **then**
              $C_e^{(i)} = C_e^{(i)} + 1$;
              $\mathbb{C}_{e,C_e^{(i)}}^{(i)} \leftarrow \emptyset$;
              for all $z^{(i)} \in \mathcal{S}^{(i)}$ **do**
                for all $w^{(i)} \in \mathcal{S}^{(i)}$ **do**
                  if $(f_{e^{(i)},(x^{(i)},y^{(i)})} \mathcal{R}_e^{(i)} f_{e^{(i)},(z^{(i)},w^{(i)})})$ **then**
                    $\mathbb{C}_{e,C_e^{(i)}}^{(i)} \leftarrow \mathbb{C}_{e,C_e^{(i)}}^{(i)} \cup \{f_{e^{(i)},(z^{(i)},w^{(i)})}\}$; /* *Ajoute* $f_{e^{(i)},(z^{(i)},w^{(i)})}$ *à classe* $\mathbb{C}_{e,C_e^{(i)}}^{(i)}$ */
                end for
              end for
          end for
        end for
      end for
    end for
end for

✍ **Soit**

$\breve{R}_{l,c}^{(i)}$ la matrice d'atteignabilité partielle locale de l'automate $\mathcal{A}^{(i)}$ (où $i \in [1..N]$) qui possède seulement les éléments de la classe $\mathbb{C}_{l,c}^{(i)}$ d'indice c (où $c \in [1..C_l^{(i)}]$) ;

$\breve{R}_{e,c}^{(i)}$ la matrice d'atteignabilité partielle de synchronisation de l'événement synchronisant $e \in \mathcal{E}_s$ dans l'automate $\mathcal{A}^{(i)}$ (où $i \in [1..N]$) qui possède seulement les éléments de la classe $\mathbb{C}_{e,c}^{(i)}$ d'indice c (où $c \in [1..C_e^{(i)}]$).

Définition 4.3.5. *Les éléments de la matrice d'atteignabilité partielle locale $\breve{R}_{l,c}^{(i)}$ de l'automate $\mathcal{A}^{(i)}$ (où $i \in [1..N]$) sont définis par :*

1. $\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $f_{l^{(i)},(x^{(i)},y^{(i)})} \in \mathbb{C}_{l,c}^{(i)}$ *est un représentant de cette classe*
   $\breve{R}_{l,c}^{(i)}(x^{(i)}, y^{(i)}) = f_{l^{(i)},(x^{(i)},y^{(i)})}$
2. $\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $f_{l^{(i)},(x^{(i)},y^{(i)})} \notin \mathbb{C}_{l,c}^{(i)}$
   $\breve{R}_{l,c}^{(i)}(x^{(i)}, y^{(i)}) = 0$

Définition 4.3.6. *Les éléments de la matrice d'atteignabilité partielle de synchronisation $\breve{R}_{e,c}^{(i)}$ de l'événement synchronisant $e \in \mathcal{E}_s$ dans l'automate $\mathcal{A}^{(i)}$ (où $i \in [1..N]$) sont définis par :*

1. $\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $f_{e^{(i)},(x^{(i)},y^{(i)})} \in \mathbb{C}_{e,c}^{(i)}$ *est un représentant de cette classe*
   $\breve{R}_{e,c}^{(i)}(x^{(i)}, y^{(i)}) = f_{e^{(i)},(x^{(i)},y^{(i)})}$
2. $\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $f_{e^{(i)},(x^{(i)},y^{(i)})} \notin \mathbb{C}_{e,c}^{(i)}$
   $\breve{R}_{e,c}^{(i)}(x^{(i)}, y^{(i)}) = 0$

En utilisant les classes d'équivalence d'une matrice d'atteignabilité partielle, on peut représenter la matrice par une somme de matrices qui ne possèdent que des éléments d'une seule classe d'équivalence.

Propriété 4.3.1. *Une matrice d'atteignabilité partielle locale $\breve{R}_l^{(i)}$ de l'automate $\mathcal{A}^{(i)}$ (où $i \in [1..N]$) peut être représentée par la somme de $C_l^{(i)}$ matrices d'atteignabilité partielles locales, utilisant la relation d'équivalence $\mathcal{R}_l^{(i)}$ (Définition 4.3.1, page 56) définie pour l'ensemble d'éléments de la matrice.*

Démonstration. Tout élément de la matrice $\breve{R}_l^{(i)}$ appartient à **une seule** classe d'équivalence $\mathbb{C}_{l,c}^{(i)}$ d'indice c, où $c \in [1..C_l^{(i)}]$. En effet, $\forall c_1, c_2 \in [1..C_l^{(i)}], \mathbb{C}_{l,c_1}^{(i)} \cap \mathbb{C}_{l,c_2}^{(i)} = \emptyset$.

Les éléments de la classe d'équivalence $\mathbb{C}_{l,c}^{(i)}$ d'indice c (où $c \in [1..C_l^{(i)}]$) figurent dans une matrice $\breve{R}_{l,c}^{(i)}$ définie par :

$\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$,
$f_{l^{(i)},(x^{(i)},y^{(i)})} \in \mathbb{C}_{l,c}^{(i)} \Rightarrow \breve{R}_{l,c}^{(i)}(x^{(i)}, y^{(i)}) = f_{l^{(i)},(x^{(i)},y^{(i)})}$
$f_{l^{(i)},(x^{(i)},y^{(i)})} \notin \mathbb{C}_{l,c}^{(i)} \Rightarrow \breve{R}_{l,c}^{(i)}(x^{(i)}, y^{(i)}) = 0$

Ainsi, on a que :

$\forall x^{(i)}, y^{(i)} \in \mathcal{S}^{(i)}$,

$\breve{R}^{(i)}_{l,c}(x^{(i)}, y^{(i)}) = f_{l^{(i)},(x^{(i)},y^{(i)})} \Rightarrow \forall c_1 \in [1..C^{(i)}_l]$, où $c_1 \neq c$, $\breve{R}^{(i)}_{l,c_1}(x^{(i)}, y^{(i)}) = 0$

$\breve{R}^{(i)}_{l,c}(x^{(i)}, y^{(i)}) = 0 \Rightarrow \exists c_1 \in [1..C^{(i)}_l]$, où $c_1 \neq c$, tel que $\breve{R}^{(i)}_{l,c_1}(x^{(i)}, y^{(i)}) = f_{l^{(i)},(x^{(i)},y^{(i)})}$ et

$\forall c_2 \in [1..C^{(i)}_l]$, où $c_2 \neq c_1$, $\breve{R}^{(i)}_{l,c_2}(x^{(i)}, y^{(i)}) = 0$

Donc, on a finalement :

$$\breve{R}^{(i)}_{l} = \sum_{c=1}^{C^{(i)}_l} \breve{R}^{(i)}_{l,c} \tag{4.1}$$

❑

De façon analogue, on peut appliquer cette propriété pour les matrices d'atteignabilité partielles de synchronisation.

Propriété 4.3.2. *Une matrice d'atteignabilité partielle de synchronisation $\breve{R}^{(i)}_{e}$ de l'événement $e \in \mathcal{E}_s$ dans l'automate $\mathcal{A}^{(i)}$ (où $i \in [1..N]$) peut être représentée par la somme de $C^{(i)}_e$ matrices d'atteignabilité partielles de synchronisation, utilisant la relation d'équivalence $\mathcal{R}^{(i)}_e$ (Définition 4.3.2, page 56) définie pour l'ensemble des éléments de la matrice et*

$$\breve{R}^{(i)}_{e} = \sum_{c=1}^{C^{(i)}_e} \breve{R}^{(i)}_{e,c} \tag{4.2}$$

Ensuite, dans ALG. 4.3, on présente l'algorithme pour la décomposition des matrices d'atteignabilité partielles d'un modèle SAN par classe d'équivalence. La décomposition des matrices est faite en deux parties : (1) décomposition des matrices d'atteignabilité partielles locales (lignes 2 - 12) et (2) décomposition des matrices d'atteignabilité partielles de synchronisation (lignes 13 - 25).

L'idée principale de ALG. 4.3 est de décomposer une matrice d'atteignabilité partielle (locale ou de synchronisation) en plusieurs matrices en fonction du nombre de classes d'équivalence des éléments de cette matrice. Pour chaque matrice, on analyse les classes d'équivalence de cette matrice. On prend un élément quelconque d'une classe d'équivalence et on vérifie tous les autres éléments de la matrice qui sont en relation avec cet élément, conservant ces éléments dans une nouvelle matrice d'atteignabilité partielle.

En utilisant les matrices d'atteignabilité partielles d'un modèle SAN, on peut représenter la matrice globale d'atteignabilité $\hat{R}$ par une formule tensorielle généralisée équivalente. Cette formulation tensorielle généralisée est dénommée *descripteur d'atteignabilité* d'un modèle SAN. Ensuite, dans le théorème 4.3.1, on démontre que le descripteur d'atteignabilité R d'un modèle SAN est une matrice globale d'atteignabilité $\hat{R}$ de ce modèle.

ALG. 4.3 SplitPartialReachabilityMatrices()

1: **for** $i = 1$ to N **do**
2: **for** $c = 1$ to $C_l^{(i)}$ **do**
3: $g \leftarrow Pick(\mathbb{C}_{l,c}^{(i)})$; /* *Prend un élément de* $\mathbb{C}_{l,c}^{(i)}$ */
4: **for all** $x^{(i)} \in \mathcal{S}^{(i)}$ **do**
5: **for all** $y^{(i)} \in \mathcal{S}^{(i)}$ **do**
6: **if** $(f_{l^{(i)},(x^{(i)},y^{(i)})} \mathcal{R}_l^{(i)} g)$ **then**
7: $\breve{R}_{l,c}^{(i)}(x^{(i)}, y^{(i)}) = f_{l^{(i)},(x^{(i)},y^{(i)})}$; /* *Préserve* $f_{l^{(i)},(x^{(i)},y^{(i)})}$ *dans* $\breve{R}_{l,c}^{(i)}(x^{(i)}, y^{(i)})$ */
8: **else**
9: $\breve{R}_{l,c}^{(i)}(x^{(i)}, y^{(i)}) = 0$; /* *Ignore* $f_{l^{(i)},(x^{(i)},y^{(i)})}$ *dans* $\breve{R}_{l,c}^{(i)}(x^{(i)}, y^{(i)})$ */
10: **end for**
11: **end for**
12: **end for**
13: **for** $e = 1$ to E **do**
14: **for** $c = 1$ to $C_e^{(i)}$ **do**
15: $g \leftarrow Pick(\mathbb{C}_{e,c}^{(i)})$; /* *Prend un élément de* $\mathbb{C}_{e,c}^{(i)}$ */
16: **for all** $x^{(i)} \in \mathcal{S}^{(i)}$ **do**
17: **for all** $y^{(i)} \in \mathcal{S}^{(i)}$ **do**
18: **if** $(f_{e^{(i)},(x^{(i)},y^{(i)})} \mathcal{R}_e^{(i)} g)$ **then**
19: $\breve{R}_{e,c}^{(i)}(x^{(i)}, y^{(i)}) = f_{e^{(i)},(x^{(i)},y^{(i)})}$; /* *Préserve* $f_{e^{(i)},(x^{(i)},y^{(i)})}$ *dans* $\breve{R}_{e,c}^{(i)}(x^{(i)}, y^{(i)})$ */
20: **else**
21: $\breve{R}_{e,c}^{(i)}(x^{(i)}, y^{(i)}) = 0$; /* *Ignore* $f_{e^{(i)},(x^{(i)},y^{(i)})}$ *dans* $\breve{R}_{e,c}^{(i)}(x^{(i)}, y^{(i)})$ */
22: **end for**
23: **end for**
24: **end for**
25: **end for**
26: **end for**

✍ **Rappelons**

$\mathcal{O}^{(e)}$ l'ensemble d'indices i ($i \in [1..N]$) tel que l'automate $\mathcal{A}^{(i)}$ contienne au moins un tuple de transition avec l'identificateur de l'événement e dans un élément de $\mathcal{Q}^{(i)}$;

Théorème 4.3.1. *Le descripteur d'atteignabilité R d'un modèle SAN à temps continu représenté par la formule tensorielle généralisée :*

$$R = \bigoplus_{i=1}^{N}{}_g \breve{R}_l^{(i)} + \sum_{e \in \mathcal{E}_s} \bigotimes_{i=1}^{N}{}_g \breve{R}_e^{(i)} \tag{4.3}$$

est une matrice globale d'atteignabilité.

Démonstration. Les éléments de l'algèbre tensorielle utilisés pour cette démonstration sont explicités dans l'Annexe A.

Notons $\mathcal{T}_l$ et $\mathcal{T}_e$ les parties dites *locale* et *synchronisante* respectivement de l'équation (4.3), où :

$$\mathcal{T}_l = \bigoplus_{i=1}^{N}{}_g \breve{R}_l^{(i)} \tag{4.4}$$

$$\mathcal{T}_e = \bigotimes_{i=1}^{N}{}_g \breve{R}_e^{(i)} \tag{4.5}$$

En utilisant seulement des produits tensoriels généralisés pour l'équation (4.4), on a :

$$\begin{aligned} \mathcal{T}_l \;=\; & \breve{R}_l^{(1)} \underset{g}{\otimes} I_{|\mathcal{S}^{(2)}|} \underset{g}{\otimes} \ldots \underset{g}{\otimes} I_{|\mathcal{S}^{(N)}|} + \ldots + \\ & I_{|\mathcal{S}^{(1)}|} \underset{g}{\otimes} \ldots \underset{g}{\otimes} \breve{R}_l^{(i)} \underset{g}{\otimes} \ldots \underset{g}{\otimes} I_{|\mathcal{S}^{(N)}|} + \ldots + \\ & I_{|\mathcal{S}^{(1)}|} \underset{g}{\otimes} \ldots \underset{g}{\otimes} I_{|\mathcal{S}^{(N-1)}|} \underset{g}{\otimes} \breve{R}_l^{(N)} \end{aligned}$$

Pour tout $i \in [1..N]$, on note

$$\mathcal{T}_l^{(i)} = I_{|\mathcal{S}^{(1)}|} \underset{g}{\otimes} \ldots \underset{g}{\otimes} \breve{R}_l^{(i)} \underset{g}{\otimes} \ldots \underset{g}{\otimes} I_{|\mathcal{S}^{(N)}|} \tag{4.6}$$

Alors,

$$\mathcal{T}_l = \sum_{i=1}^{N} \mathcal{T}_l^{(i)}$$

Et par conséquent,

$$R = \sum_{i=1}^{N} \mathcal{T}_l^{(i)} + \sum_{e \in \mathcal{E}_s} \mathcal{T}_e$$

Utilisant l'équation (4.6), pour tout $\tilde{x}, \tilde{y} \in \hat{\mathcal{S}}$, on a

$$\mathcal{T}_l^{(i)}(\tilde{x}, \tilde{y}) = \breve{R}_l^{(i)}(x^{(i)}, y^{(i)})(\tilde{x}) \times \prod_{j=1, j\neq i}^{N} \delta(x^{(j)}, y^{(j)}) \tag{4.7}$$

où $\delta(x, y)$ est la fonction Kronecker définie comme suit :

$$\delta(x, y) = \begin{cases} 1 & \text{si } x = y \\ 0 & \text{si } x \neq y \end{cases}$$

D'où, pour tout $\tilde{x}, \tilde{y} \in \hat{\mathcal{S}}$, pour tout $i \in [1..N]$

$$\mathcal{T}_l^{(i)}(\tilde{x}, \tilde{y}) = \begin{cases} \breve{R}_l^{(i)}(x^{(i)}, y^{(i)})(\tilde{x}) & \text{si } \tilde{y} \in succ_e(\tilde{x}) \text{ n'est pas vide}, e \in \mathcal{E}_l \\ 0 & \text{si } \tilde{y} \in succ_e(\tilde{x}) \text{ est vide}, e \in \mathcal{E}_l \end{cases} \tag{4.8}$$

Utilisant l'équation (4.5), pour tout $\tilde{x}, \tilde{y} \in \hat{\mathcal{S}}$, pour tout $e \in \mathcal{E}_s$, on a

$$\mathcal{T}_e(\tilde{x}, \tilde{y}) = \prod_{i \in \mathcal{O}^{(e)}} \breve{R}_e^{(i)}(x^{(i)}, y^{(i)})(\tilde{x}) \tag{4.9}$$

Remarquons que si l'automate $\mathcal{A}^{(i)}$ n'est pas concerné par l'événement synchronisant e (*i.e.*, $i \in \mathcal{O}^{(e)}$), alors $\breve{R}_e^{(i)} = I_{|\mathcal{S}^{(i)}|}$. Donc, l'équation (4.9) implique que :

$$\mathcal{T}_e(\tilde{x}, \tilde{y}) = \prod_{i \in \mathcal{O}^{(e)}} \breve{R}_e^{(i)}(x^{(i)}, y^{(i)})(\tilde{x}) \times \prod_{i \notin \mathcal{O}^{(e)}} \delta(x^{(i)}, y^{(i)}) \tag{4.10}$$

D'où, pour tout $\tilde{x}, \tilde{y} \in \hat{\mathcal{S}}$, pour tout $e \in \mathcal{E}_s$

$$\mathcal{T}_e(\tilde{x}, \tilde{y}) = \begin{cases} \prod_{i \in \mathcal{O}^{(e)}} \breve{R}_e^{(i)}(x^{(i)}, y^{(i)})(\tilde{x}) & \text{si } \tilde{y} \in succ_e(\tilde{x}) \text{ n'est pas vide}, e \in \mathcal{E}_s \\ 0 & \text{si } \tilde{y} \in succ_e(\tilde{x}) \text{ est vide}, e \in \mathcal{E}_s \end{cases} \tag{4.11}$$

Pour démontrer que le descripteur d'atteignabilité R est une matrice globale d'atteignabilité, il faut qu'il respecte la définition 4.1.1.

Car,

1. $\forall \tilde{x}, \tilde{y} \in \hat{\mathcal{S}}$ tel que $\tilde{y} \in succ_e(\tilde{x})$ n'est pas vide,

   si $e \in \mathcal{E}_l$, alors par l'équation (4.8), $\mathcal{T}_l^{(i)}(\tilde{x}, \tilde{y}) = \breve{R}_l^{(i)}(x^{(i)}, y^{(i)})(\tilde{x})$ qui est différente de zéro si $\tilde{y} \in succ_e(\tilde{x})$

   si $e \in \mathcal{E}_s$, alors par l'équation (4.11), $\mathcal{T}_e(\tilde{x}, \tilde{y}) = \prod_{i \in \mathcal{O}^{(e)}} \breve{R}_e^{(i)}(x^{(i)}, y^{(i)})(\tilde{x})$ qui est différente de zéro si $\tilde{y} \in succ_e(\tilde{x})$

qui remplit la condition 1 de la définition 4.1.1.

2. $\forall \tilde{x}, \tilde{y} \in \hat{\mathcal{S}}$ tel que $\forall e \in \mathcal{E}, \tilde{y} \in succ_e(\tilde{x})$ est vide,
   si $e \in \mathcal{E}_l$, alors par l'équation (4.8), $\mathcal{T}_l^{(i)}(\tilde{x}, \tilde{y}) = 0$
   si $e \in \mathcal{E}_s$, alors par l'équation (4.11), $\mathcal{T}_e(\tilde{x}, \tilde{y}) = 0$

ce qui nous remplit la condition 2 de la définition 4.1.1.

❑

On va maintenant développer l'expression du descripteur d'atteignabilité. Étant donné que toute somme tensorielle est équivalente à une somme de produits tensoriels particuliers (voir Annexe A), on peut réécrire l'équation (4.3) de la façon suivante :

$$R = \sum_{j=1}^{N} \bigotimes_{i=1}^{N}{}_g \breve{R}_j^{(i)} + \sum_{e \in \mathcal{E}_s} \bigotimes_{i=1}^{N}{}_g \breve{R}_e^{(i)} \qquad \text{où } \breve{R}_j^{(i)} = \begin{cases} \breve{R}_l^{(i)} & \text{si } j = i \\ I_{|\mathcal{S}^{(i)}|} & \text{si } j \neq i \end{cases} \tag{4.12}$$

En utilisant les propriétés 4.3.1 et 4.3.2 des matrices d'atteignabilité partielles - équations (4.1) et (4.2), on peut encore réécrire l'équation (4.12) et le descripteur est alors :

$$R = \sum_{j=1}^{N} \bigotimes_{i=1}^{N}{}_g \sum_{c=1}^{C_l^{(i)}} \breve{R}_{j,c}^{(i)} + \sum_{e \in \mathcal{E}_s} \bigotimes_{i=1}^{N}{}_g \sum_{c=1}^{C_e^{(i)}} \breve{R}_{e,c}^{(i)} \quad \text{où } \breve{R}_{j,c}^{(i)} = \begin{cases} \breve{R}_{l,c}^{(i)} & \text{si } j = i \\ I_{|\mathcal{S}^{(i)}|} & \text{si } j \neq i \end{cases} \tag{4.13}$$

Propriété 4.3.3. *En utilisant la propriété de la* distributivité du produit tensoriel sur la somme *[55] et avec les classes d'équivalence (Définitions 4.3.3 et 4.3.4), on a la décomposition*

$$R = \sum_{j=1}^{N} \sum_{c=1}^{C_l^{(j)}} \bigotimes_{i=1}^{N}{}_g \breve{R}_{j,c}^{(i)} + \sum_{e \in \mathcal{E}_s} \sum_{c^{(1)}=1}^{C_e^{(1)}} \cdots \sum_{c^{(N)}=1}^{C_e^{(N)}} \bigotimes_{i=1}^{N}{}_g \breve{R}_{e,c^{(i)}}^{(i)} \quad \textit{où } \breve{R}_{j,c}^{(i)} = \begin{cases} \breve{R}_{l,c}^{(i)} & \textit{si } j = i \\ I_{|\mathcal{S}^{(i)}|} & \textit{si } j \neq i \end{cases} \tag{4.14}$$

En observant l'équation (4.14), on présente dans FIG. 4.1 une comparaison du nombre de termes tensoriels du descripteur d'atteignabilité d'un modèle SAN avec le nombre de termes tensoriels du descripteur Markovien de ce modèle.

Il est intéressant de remarquer que si un modèle possède seulement des éléments constants dans ses matrices, alors le descripteur Markovien et le descripteur d'atteignabilité possèdent la même quantité de termes tensoriels (des termes locaux et de synchronisation positive), vu qu'il y a une seule classe d'équivalence pour les éléments constants des matrices d'atteignabilité partielles.

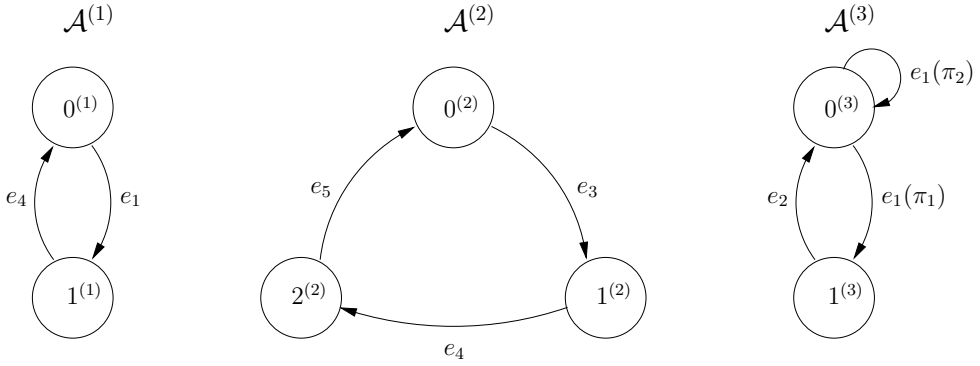
4.4 Exemple d'obtention du descripteur d'atteignabilité

Dans cette section, on présente un petit modèle SAN afin d'illustrer l'obtention des matrices du descripteur d'atteignabilité à partir des matrices du descripteur Markovien du modèle.

Termes tensoriels	
Descripteur Markovien	*Descripteur d'atteignabilité*
$N+\lvert\mathcal{E}_s\rvert$	$\sum_{i=1}^{N} C_l^{(i)} + \sum_{e\in\mathcal{E}_s}\prod_{i=1}^{N} C_e^{(i)}$

FIG. 4.1 – Comparaison du nombre de termes tensoriels des descripteurs

Dans FIG. 4.2, on présente un modèle SAN avec trois automates, où l'état initial du modèle est $0^{(1)}0^{(2)}0^{(3)}$. L'automate $\mathcal{A}^{(1)}$ possède deux états $0^{(1)}$ et $1^{(1)}$ et deux événements synchronisants e_1 et e_4. L'automate $\mathcal{A}^{(2)}$ possède trois états $0^{(2)}$, $1^{(2)}$ et $2^{(2)}$, deux événements locaux e_3 et e_5, et un événement synchronisant e_4. Le troisième automate ($\mathcal{A}^{(3)}$) possède deux états $0^{(3)}$ et $1^{(3)}$, un événement local e_2 et un événement synchronisant e_1.



Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux
syn	e_1	α	loc	e_3	f_3	loc	e_5	λ
loc	e_2	β	syn	e_4	δ			

$f_3 = (st\ \mathcal{A}^{(1)} == 1^{(1)}) \times \gamma$
$\pi_1 = (st\ \mathcal{A}^{(2)} == 0^{(2)})$
$\pi_2 = 1 - \pi_1$

FIG. 4.2 – Modèle SAN

De plus, on peut observer dans FIG. 4.2 que le taux de l'événement local e_3 est exprimé par la fonction f_3 qui dépend de l'automate $\mathcal{A}^{(1)}$. De même, l'événement synchronisant e_1 possède deux probabilités de routage fonctionnelles π_1 et π_2 dans les transitions de l'automate $\mathcal{A}^{(3)}$. Les taux de tous les autres événements du modèle sont exprimés par des valeurs constantes.

À partir des matrices du descripteur Markovien du modèle proposé dans FIG. 4.2, on peut obtenir les matrices (partielles) du descripteur d'atteignabilité de ce même modèle. En utilisant l'algorithme décrit dans ALG. 4.1, les matrices locales du descripteur d'atteignabilité sont présentées dans FIG. 4.3.

En observant les matrices d'atteignabilité locales dans FIG. 4.3, les taux constants λ et β ont été remplacés par la valeur 1, et les éléments diagonaux des matrices $\breve{R}_l^{(2)}$ et $\breve{R}_l^{(3)}$ des automates $\mathcal{A}^{(2)}$ et $\mathcal{A}^{(3)}$ ont été remplacés par la valeur 0. Il y a seulement un élément qui n'a pas changé de valeur : l'élément fonctionnel f_3 dans la matrice $\breve{R}_l^{(2)}$ de l'automate $\mathcal{A}^{(2)}$. La fonction f_3 de l'élément reste représentée dans la matrice pour être évaluée ultérieurement pour les états globaux du modèle (voir FIG. 4.7).

matrices locales du descripteur Markovien

$$Q_l^{(1)} = \begin{vmatrix} 0 & 0 \\ 0 & 0 \end{vmatrix} \qquad Q_l^{(2)} = \begin{vmatrix} -f_3 & f_3 & 0 \\ 0 & 0 & 0 \\ \lambda & 0 & -\lambda \end{vmatrix} \qquad Q_l^{(3)} = \begin{vmatrix} 0 & 0 \\ \beta & -\beta \end{vmatrix}$$

matrices locales du descripteur d'atteignabilité

$$\breve{R}_l^{(1)} = \begin{vmatrix} 0 & 0 \\ 0 & 0 \end{vmatrix} \qquad \breve{R}_l^{(2)} = \begin{vmatrix} 0 & f_3 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{vmatrix} \qquad \breve{R}_l^{(3)} = \begin{vmatrix} 0 & 0 \\ 1 & 0 \end{vmatrix}$$

$$C_l^{(1)} = 0 \qquad C_l^{(2)} = 2 \qquad C_l^{(3)} = 1$$

FIG. 4.3 – Les matrices d'atteignabilité locales du modèle SAN présenté dans FIG. 4.2

De même, les matrices d'atteignabilité de synchronisation peuvent être obtenues à partir des matrices de synchronisation du descripteur Markovien du modèle SAN. Dans FIG. 4.4, on peut observer les matrices du descripteur Markovien des événements synchronisants e_1 et e_4, ainsi que les matrices équivalentes de ces événements pour le descripteur d'atteignabilité du modèle SAN présenté dans FIG. 4.2.

matrices de synchronisation du descripteur Markovien

$$Q_{e_1^+}^{(1)} = \begin{vmatrix} 0 & \alpha \\ 0 & 0 \end{vmatrix} \qquad Q_{e_1^+}^{(2)} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad Q_{e_1^+}^{(3)} = \begin{vmatrix} f_2 & f_1 \\ 0 & 0 \end{vmatrix}$$

matrices de synchronisation du descripteur d'atteignabilité

$$\breve{R}_{e_1}^{(1)} = \begin{vmatrix} 0 & 1 \\ 0 & 0 \end{vmatrix} \qquad \breve{R}_{e_1}^{(2)} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad \breve{R}_{e_1}^{(3)} = \begin{vmatrix} f_2 & f_1 \\ 0 & 0 \end{vmatrix}$$

$$C_{e_1}^{(1)} = 1 \qquad C_{e_1}^{(2)} = 1 \qquad C_{e_1}^{(3)} = 2$$

$$Q_{e_4^+}^{(1)} = \begin{vmatrix} 0 & 0 \\ \delta & 0 \end{vmatrix} \qquad Q_{e_4^+}^{(2)} = \begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{vmatrix} \qquad Q_{e_4^+}^{(3)} = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}$$

$$\breve{R}_{e_4}^{(1)} = \begin{vmatrix} 0 & 0 \\ 1 & 0 \end{vmatrix} \qquad \breve{R}_{e_4}^{(2)} = \begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{vmatrix} \qquad \breve{R}_{e_4}^{(3)} = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}$$

$$C_{e_4}^{(1)} = 1 \qquad C_{e_4}^{(2)} = 1 \qquad C_{e_4}^{(3)} = 1$$

FIG. 4.4 – Les matrices d'atteignabilité de synchronisation du modèle SAN présenté dans FIG. 4.2

De façon analogue, les taux constants α et δ ont aussi été remplacés par la valeur 1 dans les matrices d'atteignabilité de synchronisation dans FIG. 4.4. Les deux éléments fonctionnels f_1 et f_2 (correspondant aux probabilités de routage π_1 et π_2 respectivement) sont conservés dans la matrice d'atteignabilité partielle $\breve{R}_{e_1}^{(3)}$.

On ne tient pas compte des matrices de synchronisation (négatives) des événements synchronisants du descripteur Markovien, sachant que l'objectif du descripteur d'atteignabilité est de ne représenter que les relations de transition entre les états du modèle. Tous les ajustements de l'occurrence des événements ne sont pas analysés, par conséquent les matrices d'atteignabilité de synchronisation négatives des événements ne sont pas représentées.

Une fois que les matrices du descripteur d'atteignabilité sont obtenues, il faut décomposer ces matrices de manière à ce que ses éléments soient en relation conformément à relation d'équivalence définie à partir des états qui annulent (ou non) cette fonction (Définitions 4.3.1 et 4.3.2, page 56).

Comme on peut l'observer dans FIG. 4.3 et FIG. 4.4, il y a seulement deux matrices d'atteignabilité ($\breve{R}_l^{(2)}$ et $\breve{R}_{e_1}^{(3)}$) qui possèdent plus d'une classe d'équivalence (*i.e.*, $C_l^{(2)} = 2$ et $C_{e_1}^{(3)} = 2$). Dans ce cas, ces matrices doivent être décomposées en deux autres matrices. En utilisant l'algorithme présenté dans ALG. 4.3, on présente dans FIG. 4.5 les matrices d'atteignabilité décomposées équivalentes aux matrices $\breve{R}_l^{(2)}$ et $\breve{R}_{e_1}^{(3)}$.

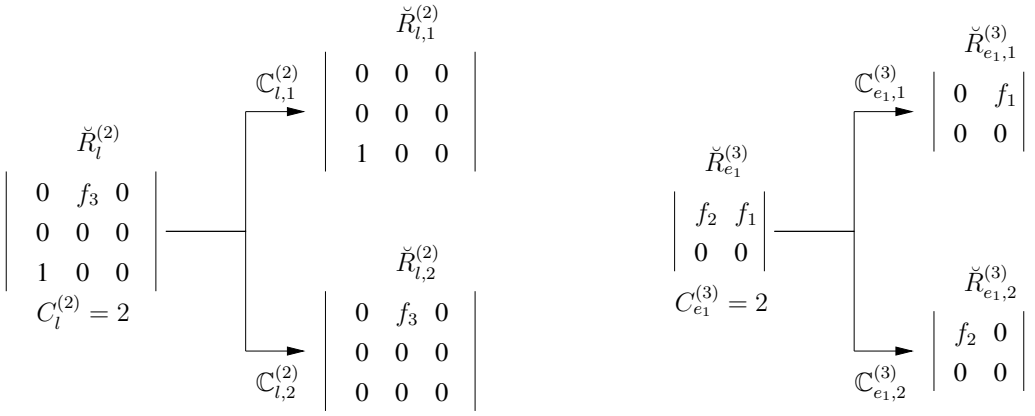


FIG. 4.5 – La décomposition des matrices d'atteignabilité du modèle SAN présenté dans FIG. 4.2

Il faut aussi tenir compte du format tensoriel du descripteur d'atteignabilité. Ce format permet de représenter d'une façon compacte les relations de transition entre les états, en stockant un ensemble de petites matrices au lieu d'une grande matrice globale. Dans FIG. 4.7, on présente la matrice globale d'atteignabilité $\hat{R}$ équivalente au descripteur d'atteignabilité R du modèle SAN présenté dans FIG. 4.2.

Toutefois, comme les matrices sont décomposées, de nouveaux termes tensoriels sont présentés dans le descripteur d'atteignabilité et une nouvelle représentation tensorielle équivalente est obtenue. Dans FIG. 4.6, on présente les termes tensoriels du descripteur d'atteignabilité décomposé pour le modèle présenté dans FIG. 4.2.

En observant FIG. 4.6, on vérifie qu'il y a des matrices qui sont utilisées dans plus qu'un terme tensoriel (*e.g.*, la matrice $\breve{R}_{e_1,1}^{(1)}$ dans le cinquième et sixième terme tensoriel).

Dans FIG. 4.7, on observe que les éléments fonctionnels f_1, f_2 et f_3 sont déjà évalués pour les états globaux du modèle. Par exemple, le résultat de l'évaluation de f_3 est différent de zéro pour l'élément $\hat{R}(6, 8)$ (ligne 6 et colonne 8 de la matrice globale d'atteignabilité $\hat{R}$) correspondant à l'élément global $1^{(1)}0^{(2)}0^{(3)}$ du modèle, tandis que le résultat de l'évaluation pour cette même fonction est égal à zéro pour les éléments $\hat{R}(0, 2)$ et $\hat{R}(1, 3)$.

Il est intéressant de remarquer qu'on peut facilement vérifier quels sont les états globaux atteignables et non-atteignables du modèle en regardant les colonnes de la matrice globale d'atteignabilité, *i.e.*, si une colonne de la matrice possède seulement des éléments nuls, alors l'état global correspondant à cette colonne est un état *non-atteignable*.

Cependant, la matrice globale d'atteignabilité n'est pas utilisée dans son format plein, vu que le format tensoriel du descripteur d'atteignabilité (équivalent à la matrice globale) permet une représentation plus compacte.

	Descripteur d'atteignabilité
$\sum$	$\breve{R}_l^{(1)} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(2)}\rvert} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(3)}\rvert}$
	$I_{\lvert\mathcal{S}^{(1)}\rvert} \underset{g}{\otimes} \breve{R}_l^{(2)} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(3)}\rvert}$
	$I_{\lvert\mathcal{S}^{(1)}\rvert} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(2)}\rvert} \underset{g}{\otimes} \breve{R}_l^{(3)}$
	$\breve{R}_{e_1}^{(1)} \underset{g}{\otimes} \breve{R}_{e_1}^{(2)} \underset{g}{\otimes} \breve{R}_{e_1}^{(3)}$
	$\breve{R}_{e_4}^{(1)} \underset{g}{\otimes} \breve{R}_{e_4}^{(2)} \underset{g}{\otimes} \breve{R}_{e_4}^{(3)}$

	Descripteur d'atteignabilité décomposé
$\sum$	$\breve{R}_{l,1}^{(1)} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(2)}\rvert} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(3)}\rvert}$
	$I_{\lvert\mathcal{S}^{(1)}\rvert} \underset{g}{\otimes} \breve{R}_{l,1}^{(2)} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(3)}\rvert}$
	$I_{\lvert\mathcal{S}^{(1)}\rvert} \underset{g}{\otimes} \breve{R}_{l,2}^{(2)} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(3)}\rvert}$
	$I_{\lvert\mathcal{S}^{(1)}\rvert} \underset{g}{\otimes} I_{\lvert\mathcal{S}^{(2)}\rvert} \underset{g}{\otimes} \breve{R}_{l,1}^{(3)}$
	$\breve{R}_{e_1,1}^{(1)} \underset{g}{\otimes} \breve{R}_{e_1,1}^{(2)} \underset{g}{\otimes} \breve{R}_{e_1,1}^{(3)}$
	$\breve{R}_{e_1,1}^{(1)} \underset{g}{\otimes} \breve{R}_{e_1,1}^{(2)} \underset{g}{\otimes} \breve{R}_{e_1,2}^{(3)}$
	$\breve{R}_{e_4,1}^{(1)} \underset{g}{\otimes} \breve{R}_{e_4,1}^{(2)} \underset{g}{\otimes} \breve{R}_{e_4,1}^{(3)}$

FIG. 4.6 – Les termes tensoriels du descripteur d'atteignabilité du modèle SAN présenté dans FIG. 4.2

$$\hat{R} = \left(\begin{array}{cc|cc|cc||cc|cc|cc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right)$$

FIG. 4.7 – La matrice globale d'atteignabilité du modèle SAN présenté dans FIG. 4.2

4.5 Conclusion

Dans ce chapitre, on a présenté le descripteur d'atteignabilité d'un modèle SAN à temps continu et comment on peut obtenir les matrices d'atteignabilité de ce descripteur à partir des matrices du descrip-

teur Markovien de ce même modèle. On a aussi définit le format adéquat du descripteur d'atteignabilité qui sera utilisé par les méthodes de la génération de l'espace d'états atteignables des modèles SAN (voir Chapitre 5). Il constitue une expression "structurée" de la fonction "*next-state*", nécessaire à la génération d'espace d'états atteignables. Ce format tensoriel du descripteur d'atteignabilité permet un calcul performant de l'espace d'états atteignables.

On a présenté l'algorithme utilisé pour l'obtention des matrices d'atteignabilité du descripteur, ainsi que l'algorithme pour décomposer ces matrices afin d'adapter le format tensoriel du descripteur d'atteignabilité.

Dans le chapitre suivant, on va présenter les nouvelles méthodes de la génération de l'espace d'états atteignables de modèles qui possèdent taux ou probabilités représentés par des fonctions.

Chapitre 5

Génération de l'espace d'états atteignables de modèles qui utilisent des fonctions

Dans ce chapitre, on présente des méthodes de génération de *l'espace d'états atteignables* (RSS - *Reachable State Space*) de modèles qui utilisent des *fonctions*. On commence d'abord par faire référence à des travaux antérieurs [1, 94, 23, 30] qui sont réécrits dans le contexte du formalisme des *Réseaux d'Automates Stochastiques* (SAN) et ensuite on présente un travail original pour la génération du RSS de modèles qui possèdent des taux et probabilités fonctionnels.

Dans les modèles SAN, le *taux d'occurrence d'un événement*, ainsi que la *probabilité de routage d'une transition lors de l'occurrence d'un événement* peuvent être exprimés par une *fonction*. Les travaux précédents de l'*algorithme de Saturation* [32, 30], ainsi que l'*algorithme de Parcours en Largeur* (BFS - *Breadth-First Search*) [94] considèrent l'utilisation de fonctions, où les arguments de ces fonctions sont limités à l'espace d'états local d'un sous-système (*i.e.*, fonctions qui peuvent être évaluées à priori). Les nouvelles méthodes de génération de l'espace d'états atteignables présentées dans ce chapitre permettent l'utilisation des ***fonctions sans hypothèse sur les arguments***.

L'utilisation des *Diagrammes de Décision Binaire* (BDD - *Binary Decision Diagrams*) [18] a eu une grande influence sur la vérification de modèles de systèmes. Les BDD ont fourni des gains de performance pour la génération de l'espace d'états de modèles en utilisant des techniques traditionnelles explicites [25]. Les vérificateurs de modèles symboliques basés sur les BDD [35] sont capables de vérifier automatiquement des propriétés temporelles de circuits complexes et de contrôleurs synchrones. Toutefois, ces vérificateurs ne sont pas directement performants pour des modèles composés de sous-systèmes qui interagissent et se synchronisent entre eux par des événements partagés (*e.g.*, des modèles de systèmes décrits par formalismes structurés de haut-niveau). D'une façon générale, ces modèles souffrent du problème de l'*explosion combinatoire de l'espace d'états*.

La génération de l'espace d'états est le principal défi pour la grande majorité des outils de vérification formelle (par exemple, le vérificateur de modèles [37]). L'ensemble des *états initiaux* et la fonction "*next-state*" sont utilisés par les générateurs traditionnels de l'espace d'états symboliques afin d'obtenir l'espace d'états du système modélisé représenté sur la forme d'un BDD. La représentation de l'espace d'états atteignables par un BDD d'un modèle est faite par l'application itérative de la fonction "*next-state*" en utilisant l'ensemble d'états initiaux représenté aussi par un BDD.

Parmi les différents générateurs explicites de l'espace d'états (pour lesquels on observe une augmentation linéaire de l'utilisation de la mémoire en fonction du nombre d'états explorés), l'algorithme BFS a la caractéristique d'offrir une augmentation et une réduction de la mémoire utilisée pendant l'exécution. En général, bien avant que l'algorithme ne finisse, le *pic* de mémoire utilisée pour représenter l'espace d'états par un BDD est à même d'être stocké dans la mémoire d'un ordinateur moderne.

Les modèles de systèmes décrits par des formalismes structurés possèdent des représentations structurées des états et de la fonction "*next-state*". Les états sont représentés par des vecteurs, où chaque position du vecteur représente l'état d'un sous-système du système modélisé. L'ensemble des vecteurs d'états peut être naturellement représenté par des *Diagrammes de Décision Multi-valués* (MDD - *Multi-valued Decision Diagrams*) [82]. Un MDD est une extension de la structure de BDD, appliquée à une logique non-binaire.

Dans ce qui suit, on décrit comment un espace d'états peut être représenté par un MDD. Dans la section 5.2, on montre comment les taux et probabilités fonctionnels des modèles peuvent être aussi représentés par des MDD. Dans la section 5.3, on montre comment peut se faire la génération du MDD qui représente l'ensemble d'états pour lesquels l'évaluation de toutes les fonctions concernées par un terme tensoriel est différente de zéro. La décomposition préalable proposée dans le chapitre 4 nous amènera toujours dans ce cas. Dans les sections 5.4 et 5.5, on présente les algorithmes *BFS* et de *Saturation* respectivement, qui sont les méthodes de génération du RSS de modèles qui utilisent des fonctions. Pour l'implantation, dans la section 5.6, on présente en détail les structures de donnés utilisées pour la représentation et manipulation des noeuds d'un MDD. Et finalement on conclut ce chapitre par la section 5.7.

5.1 Espace d'états représenté par des MDD

Les *Diagrammes de Décision Binaire* (BDD - *Binary Decision Diagrams*) [18] sont des graphes acycliques orientés (DAG - *Directed Acyclic Graph*) avec deux noeuds terminaux représentés par les valeurs 0 et 1. En initiant un chemin à partir de la racine du graphe, on peut représenter des fonctions booléennes. Chaque noeud non-terminal du graphe représente une variable booléenne avec deux noeuds fils (pour les valeurs 0 et 1) qui représentent l'évaluation de la fonction booléenne pour *faux* ou *vrai* respectivement.

L'étude des BDD [105] a émergé comme une alternative pour le stockage de l'espace d'états atteignables de modèles de *Réseaux de Petri Stochastiques* (SPN - *Stochastic Petri Nets*), réduisant le stockage de l'espace d'états pour résoudre le problème de l'explosion combinatoire pour des systèmes complexes. Ces travaux ont inspiré les premières études [94, 95] d'une nouvelle structure dénommée *Diagramme de Décision Multi-valués* (MDD - *Multi-valued Decision Diagram*) [82].

Un MDD est une extension de la structure de BDD, appliquée à une logique non-binaire. D'abord, les MDD ont été représentés par un sous-groupe de vecteurs [33], où chaque vecteur représente un niveau (*i.e.*, un composant ou sous-système) d'un modèle SPN. Plus tard, l'utilisation de vecteurs pour représenter la structure de chaque niveau a été abandonnée, car les vecteurs ne sont pas des structures performantes pour exécuter des tâches comme la recherche et l'insertion.

Des concepts basiques préalablement étudiés pour les BDD, tels que l'*ordre* et la *réduction* de l'espace d'états, sont aussi à l'origine du format compact d'un MDD. Les deux structures, BDD et MDD,

trouvent leur format canonique si leurs espaces d'états sont bien ordonnés et réduits. Des règles de réduction [105] sont appliquées sur une structure ordonnée [19] pour éliminer des états redondants. Le choix du critère du meilleur ordre est obtenu à partir de l'implémentation de tous les ordres possibles, jusqu'à ce que la structure présente un format compact qui représente l'espace d'états concerné.

Une fois le MDD ordonné et réduit, il est dénommé *Diagramme de Décision Multi-valués Ordonné et Réduit* (ROMDD - *Reduced and Ordered Multi-valued Decision Diagram*), et considéré dans un format canonique. Notons que, pour des raisons de simplification, dans le contexte de cette thèse, on va simplement référencer un ROMDD comme un MDD, vu que toutes les structures MDD seront dans le format canonique.

Les modèles structurés basés sur l'occurrence d'événements permettent la représentation d'espaces d'états via leur fonction caractéristique qui est une fonction de $\mathcal{S}^{(N)} \times \mathcal{S}^{(N-1)} \times \cdots \times \mathcal{S}^{(1)} \rightarrow \{\mathbf{0}, \mathbf{1}\}$, où N est le nombre de sous-systèmes du modèle.

✍ **Soit**

$\mathcal{S}^{(l)}$	l'espace d'état local d'un sous-système $l \in [1..N]$, et un élément de $\mathcal{S}^{(l)}$ est noté $x^{(l)} \in [0..n_l\text{-}1]$, où $n_l = \lvert\mathcal{S}^{(l)}\rvert$ et N est égal au nombre de sous-systèmes du modèle ;
$\tilde{x}$	l'état global représenté par N variables $(x^{(N)}, \ldots, x^{(1)})$ d'un modèle structuré de N sous-systèmes.

Remarque : Un espace d'états $\mathcal{S} \subset \mathcal{S}^{(N)} \times \mathcal{S}^{(N-1)} \times \cdots \times \mathcal{S}^{(1)}$ peut être représenté par un MDD via sa fonction caractéristique $f_{\mathcal{S}}$ tel que $f_{\mathcal{S}}(\tilde{x}) = 1 \Leftrightarrow \tilde{x} \in \mathcal{S}$.

Un sous-espace d'états $\mathcal{S}$ de $\mathcal{S}^{(N)} \times \mathcal{S}^{(N-1)} \times \cdots \times \mathcal{S}^{(1)}$ peut être alors naturellement représenté par un MDD. En effet, un MDD peut représenter des fonctions du type :

$$\{0, 1, \ldots, n_N - 1\} \times \{0, 1, \ldots, n_{N-1} - 1\} \times \cdots \times \{0, 1, \ldots, n_1 - 1\} \rightarrow \{\mathbf{0}, \mathbf{1}\}.$$

Dans le reste de la section, le MDD représente (ou on dira "code") les états d'un ensemble $\mathcal{S}$ inclus dans $\mathcal{S}^{(N)} \times \mathcal{S}^{(N-1)} \times \cdots \times \mathcal{S}^{(1)}$.

Définition 5.1.1. *Un diagramme de décision multi-valués (MDD) est un multi-graphe acyclique orienté avec des arcs étiquetés qui a les propriétés suivantes :*

- *les noeuds sont disposés sur* $N + 1$ niveaux*, où* $p.lvl$ *indique le niveau du noeud* p *;*
- *le niveau* N *possède un seul noeud* non-terminal r *(la* racine*), alors que les niveaux* $N - 1$ *jusqu'à* 1 *possèdent un ou plusieurs noeuds non-terminaux ;*
- *le niveau* 0 *possède deux noeuds* terminaux *:* $\mathbf{0}$ *et* $\mathbf{1}$ *;*
- *un noeud non-terminal* p *au niveau* l *possède* n_l *arcs qui pointent vers des noeuds au niveau* $l - 1$*. Les arcs sortant du niveau* l *sont étiquetés par les états de* $\mathcal{S}^{(l)}$*, et pour un arc d'étiquette* $x^{(l)} \in \mathcal{S}^{(l)}$ *du noeud* p *qui pointe vers le noeud* q*, on note* $p[x^{(l)}] = q$ *;*
- *Il n'y a pas de noeuds* doubles *(voir Définition 5.1.2).*

Définition 5.1.2. *Soit deux noeuds non-terminaux* p *et* p' *au niveau* l *d'un MDD. On dit que* p *et* p' *sont noeuds* doubles *si et seulement si* $p[x^{(l)}] = p'[x^{(l)}]$ *pour tout* $x^{(l)} \in \mathcal{S}^{(l)}$.

FIG. 5.1 présente le MDD d'un espace d'états $\mathcal{S}$, sous-espace de l'espace produit d'un système divisé en quatre sous-systèmes. Le sous-système $\mathcal{S}^{(4)}$ a quatre états $\{0, 1, 2, 3\}$, les sous-systèmes $\mathcal{S}^{(3)}$ et $\mathcal{S}^{(1)}$ ont deux états chacun $\{0, 1\}$, et le sous-système $\mathcal{S}^{(2)}$ a trois états $\{0, 1, 2\}$.

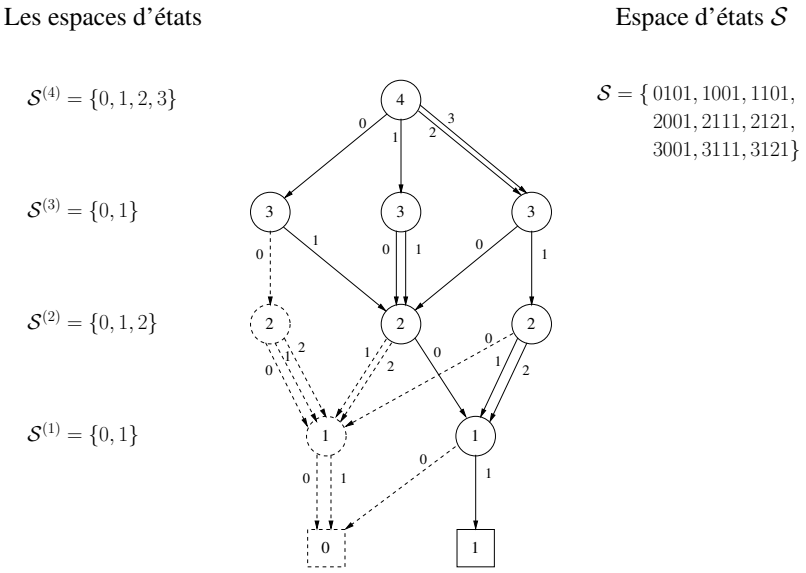


FIG. 5.1 – Exemple de représentation d'un espace d'états $\mathcal{S}$ par un MDD

Dans FIG. 5.1, les noeuds *non-terminaux* sont représentés par des *cercles* et les noeuds *terminaux* sont représentés par des *carrés*. Un élément de $\mathcal{S}$ est un état obtenu à partir des étiquettes des arcs sur un chemin allant de la racine vers le noeud terminal $\mathbf{1}$. Les noeuds et les arcs en pointillés sont sur des chemins qui n'atteignent que le noeud terminal $\mathbf{0}$ (pas dans $\mathcal{S}$), tandis que les autres arcs sont sur des chemins qui codent les états de $\mathcal{S}$. Désormais, pour une question de clarté, on montre seulement les arcs qui représentent le codage de l'espace d'états $\mathcal{S}$ d'un MDD, et les autres noeuds et les arcs pointillés, ainsi que le noeud terminal $\mathbf{0}$ (*zéro*) ne seront pas dessinés.

Les opérations d'union et d'intersection pour les MDD peuvent être effectuées via des opérations logiques sur les fonctions caractéristiques. Par exemple, l'union d'ensembles d'états correspond à la disjonction des MDD. La complexité de ces opérations est dépendante du nombre de noeuds des MDD qui sont les paramètres de l'opération et non du nombre d'états représentés par les MDD [30].

Définition 5.1.3. *À partir d'un noeud p au niveau l d'un MDD et d'une séquence $\sigma = (x^{(l)}, \ldots, x^{(l')})$ d'états locaux qui atteint un noeud p' du niveau l' (où $l > l'$), on définit récursivement $p[\sigma]$ de la façon suivante :*

$$p[\sigma] = \begin{cases} p & \text{si } \sigma = (), \text{ la séquence est vide} \\ p[x^{(l)}][\sigma'] & \text{si } \sigma = (x^{(l)}, \sigma'), \text{ où } x^{(l)} \in \mathcal{S}^{(l)} \\ p' & \text{si } \sigma = (x^{(l)}, \ldots, x^{(l')}) \end{cases}$$

Définition 5.1.4. *Étant donné un noeud p au niveau l, l'ensemble d'états $\mathcal{B}(p)$ représente les sous-états "dessous" p ("below" p), soit $\mathcal{B}(p) = \{\sigma \in \mathcal{S}^{(l)} \times \cdots \times \mathcal{S}^{(1)} : p[\sigma] = \mathbf{1}\}$.*

Définition 5.1.5. *Étant donné un noeud p au niveau l, l'ensemble d'états $\mathcal{A}(p)$ représente les sous-états qui* atteignent *le noeud p au niveau l, i.e., "dessus" p ("above" p), soit $\mathcal{A}(p) = \{\sigma \in \mathcal{S}^{(N)} \times \cdots \times \mathcal{S}^{(l+1)} : r[\sigma] = p\}$, où le noeud r est la racine du MDD.*

Un autre détail important dans la représentation de l'espace d'états des modèles structurés par un MDD est la numérotation d'*arrière-vers-l'avant* (N à 1) des sous-systèmes au lieu d'un ordre croissant naturel. Ce codage inversé des sous-systèmes est dû au fait qu'en optant pour cette convention [135] les noeuds terminaux du MDD se trouveront toujours au niveau 0 (*zéro*).

5.2 MDD associés à des taux et des probabilités fonctionnels

L'utilisation des taux et probabilités fonctionnels est l'une des caractéristiques des modèles SAN. Afin de profiter de cette caractéristique, il est important tout d'abord de déterminer comment les taux et probabilités fonctionnels pourront être utilisées au sein des méthodes proposées dans ce chapitre.

Par définition, un MDD est une structure utilisée pour représenter des fonctions (ou des caractéristique de ces fonctions) sur un ensemble fini de variables. Ainsi, on peut représenter par un MDD l'ensemble d'états d'une fonction où la fonction ne s'annule pas. C'est de cette information dont nous aurons besoin par la suite.

La génération d'un MDD pour représenter des espaces d'états de modèles SAN est réalisé de façon simple, grâce à la structure modulaire du formalisme SAN, où les automates sont représentés par des niveaux du MDD et les états des automates sont représentés par des arcs.

✍ **Rappelons**

ω	un ensemble d'indices d'automates, où $\omega \subseteq [1..N]$;
$\tilde{x}^{(\omega)}$	le vecteur des états locaux $x^{(i)}$ tel que $i \in \omega$;
$\hat{\mathcal{S}}^{(\omega)}$	l'espace d'états produit de l'ensemble d'états locaux des automates $\mathcal{A}^{(i)}$ ($i \in \omega$) ;
$f(\tilde{x}^{(\omega)})$	un élément fonctionnel $f(\hat{\mathcal{S}}^{(\omega)})$ évalué pour le vecteur $\tilde{x}^{(\omega)}$.

Définition 5.2.1. *$FuncMdd_{id}$ est le MDD qui représente l'ensemble d'états pour lesquels l'évaluation de la fonction f_{id} est différente de zéro (i.e., pour tout $\tilde{x}^{(\omega_{id})} \in \hat{\mathcal{S}}^{(\omega_{id})}$, $f_{id}(\tilde{x}^{(\omega_{id})}) \neq 0$).*

Par exemple, FIG. 5.2 présente les MDD $FuncMdd$ de deux fonctions booléennes (f_1 et f_2) d'un modèle SAN qui a trois automates, où chaque automate a trois états qui sont $\{0, 1, 2\}$.

Comme on peut l'observer dans FIG. 5.2, le domaine de la fonction f_1 tient compte de l'espace d'états local des automates $\mathcal{A}^{(1)}$ et $\mathcal{A}^{(2)}$ (le domaine $\hat{\mathcal{S}}^{(\omega_1)}$, où $\omega_1 = \{1, 2\}$), alors que le domaine de la fonction f_2 tient compte des automates $\mathcal{A}^{(2)}$ et $\mathcal{A}^{(3)}$ (le domaine $\hat{\mathcal{S}}^{(\omega_2)}$, où $\omega_2 = \{2, 3\}$).

✍ **Soit**

F	le nombre de fonctions distinctes d'un modèle SAN.

Chaque fonction (taux ou probabilité) d'un modèle SAN possède un identificateur id unique, où $id \in [1..F]$. On considère que si deux fonctions f_i et f_j possèdent des identificateurs différents (*i.e.*, $i \neq j$), elles ne sont pas identiques (voir Définition 2.2.6, page 27).

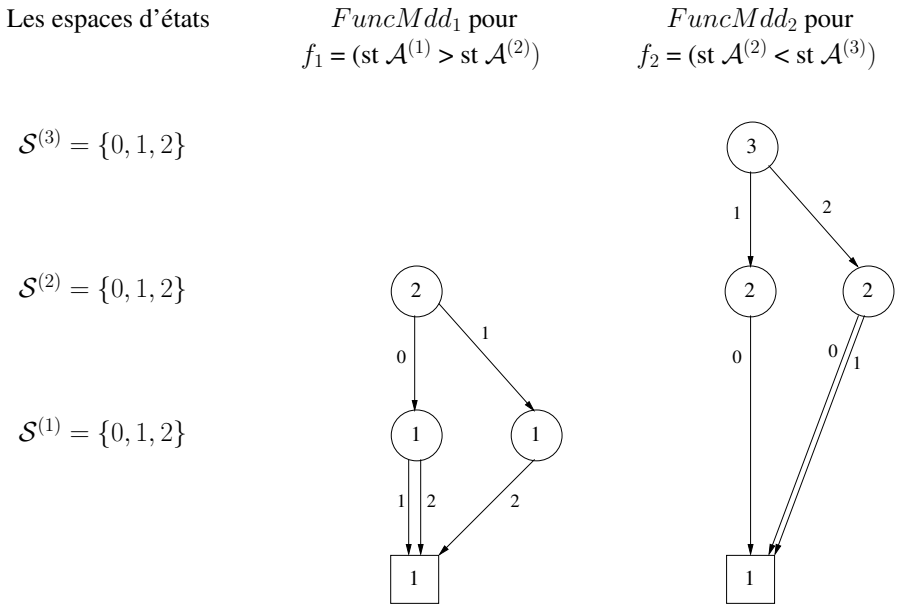


FIG. 5.2 – Taux fonctionnels représentés par des MDD

Le MDD $FuncMdd_{id}$ associé à la fonction f_{id} peut être produit par l'algorithme décrit dans ALG. 5.1. L'obtention du MDD $FuncMdd_{id}$ associé à une fonction f_{id} est une tâche relativement simple, vu qu'il est obtenu par l'union de tous les états pour lesquels l'évaluation de la fonction est différente de zéro. Autrement dit, évaluant une fonction sur tous les états de son domaine, il est possible de produire un MDD associé à cette fonction en faisant l'union de tous les états pour lesquels l'évaluation de la fonction est différente de zéro.

✍ **Soit**

id	l'identificateur d'une fonction du modèle, où $id \in [1..F]$;
StateToMdd($\tilde{x}^{(\omega)}$)	la procédure (ALG. 5.2) qui construit un MDD à partir de l'état $\tilde{x}^{(\omega)}$;
Union(M_1, M_2)	la procédure qui réalise l'union entre les MDD M_1 et M_2.

ALG. 5.1 FuncMdd(*in* id : *integer*) : *MDD*

```
1: /* Initialisation du MDD associé à la fonction f_id */
2: FuncMdd_id ← ∅ ;
3: /* Parcours des états du domaine de la fonction f_id */
4: for each x̃^(ω_id) ∈ Ŝ^(ω_id) do
5:    /* Union des états pour lesquels l'évaluation de la fonction est différente de zéro */
6:    if (f_id(x̃^(ω_id)) ≠ 0) then
7:       StateMdd ← StateToMdd(x̃^(ω_id)) ;
8:       FuncMdd_id ← Union(FuncMdd_id, StateMdd) ;
9: end for
10: return FuncMdd_id ;
```

Il est intéressant de remarquer que dans ALG. 5.1, on évalue seulement les états du domaine de la fonction f_{id} (*i.e.*, l'évaluation est faite pour tout $\tilde{x}^{(\omega_{id})} \in \hat{\mathcal{S}}^{(\omega_{id})}$), au lieu de tout $\tilde{x} \in \hat{\mathcal{S}}$ du modèle.

ALG. 5.2 StateToMdd(*in* $\tilde{x}^{(\omega)}$: *state*) : *node*

```
1: /* p un noeud initialisé au noeud terminal 1 */
2: p ← 1 ;
3: /* Prendre chaque état local x^(l) de l'état global x̃^(ω) */
4: for each l ∈ ω do
5:    /* Création d'un noeud au niveau l du MDD */
6:    r ← NewNode(l) ;
7:    r[x^(l)] ← p ;
8:    p ← r ;
9: end for
10: /* Retourne le noeud ("racine") du MDD qui représente l'état global x̃^(ω) */
11: return r ;
```

Remarque : La décomposition proposée dans le chapitre 4 (Section 4.3, page 54) des matrices d'atteignabilité selon les classes d'équivalence, assure que toutes les fonctions d'une matrice d'atteignabilité d'un terme tensoriel ont le même "noyau" (*i.e.*, le même ensemble d'arguments où elles s'annulent). Donc, toutes les fonctions f_{id} ($id \in [1..F]$) qui sont taux de transition d'une matrice d'atteignabilité $\breve{R}$ ont des $FuncMdd_{id}$ égaux. En conséquence, pour l'une de ces matrices $\breve{R}$ et étant donné l'une des fonctions f_{id} de cette matrice et son MDD associé $FuncMdd_{id}$, la matrice $\breve{R}$ est une matrice *nulle* lorsque ses fonctions sont évaluées pour $\tilde{x} \notin FuncMdd_{id}$.

✍ **Soit**

$\breve{R}(\tilde{x})$ la matrice d'atteignabilité $\breve{R}$ évaluée pour l'état global $\tilde{x}$.

Propriété 5.2.1. *Étant donné une matrice d'atteignabilité $\breve{R}$ du descripteur d'atteignabilité, une fonction f_{id} qui est taux de transition de cette matrice $\breve{R}$ en position $\tilde{x}, \tilde{y}$, on aura :*

$$\tilde{x} \notin FuncMdd_{id} \Leftrightarrow \breve{R}(\tilde{x}) = f_{id}(\tilde{x}) = 0$$

Démonstration. Sachant que tout élément d'une matrice d'atteignabilité $\breve{R}$ appartient à **une même** classe d'équivalence, tous éléments fonctionnels de la matrice $\breve{R}$ ont le même noyau (*i.e.*, le même ensemble d'arguments où l'évaluation des éléments fonctionnels est égale à zéro) et conséquemment le même MDD $FuncMdd$.

Ainsi, on a que :

$\tilde{x} \notin FuncMdd_{id}$, $f_{id}(\tilde{x}) = 0 \Rightarrow \breve{R}(\tilde{x}, \tilde{y}) = 0$

$\tilde{x} \in FuncMdd_{id}$, si $\breve{R}(\tilde{x}) = f_{id}(\tilde{x}) \neq 0 \Rightarrow \breve{R}(\tilde{x}, \tilde{y}) \neq 0$

❑

Remarque : Cette condition que toutes les fonctions d'une même matrice aient le même noyau est, semble-t-il, une condition trop forte. Il suffirait que les fonctions d'une même ligne de la matrice aient le même noyau. La preuve de cette conjecture sera l'objet d'un travail ultérieur.

Bien que la génération d'un MDD soit une tâche apparemment simple, parfois, elle peut avoir un coût de calcul très élevé. Par exemple, soit un modèle SAN avec 32 automates de 2 états chacun, si le

domaine d'une fonction f_{id} est l'ensemble des 32 automates, il faut évaluer les 2^{32} états de $\hat{\mathcal{S}}^{(\omega_{id})}$ pour construire son MDD. De plus, il y aura encore un coût de calcul pour effectuer l'union entre le MDD associé à la fonction ($FuncMdd_{id}$) et le MDD de chaque état pour lequel l'évaluation de la fonction f_{id} est différente de zéro ($StateMdd$).

Donc, la génération des MDD $FuncMdd$ associés aux fonctions du modèle est la partie qui demande le plus de calcul, vu qu'à l'heure actuelle nous n'avons pas de façon performante de produire le MDD associé à une fonction sans analyser tout l'espace d'états de la fonction. Bien que cette partie ne soit pas très performante, les méthodes pour la génération de l'espace d'états atteignables en utilisant des MDD pour le formalisme SAN ont donné de bons résultats pour différentes classes d'exemples (voir Chapitre 6).

5.3 MDD associés à des fonctions d'un terme tensoriel

Comme on l'a présenté dans le chapitre 4 (Propriété 4.3.3, page 63), le descripteur d'atteignabilité R représente les relations de transition entre les états globaux d'un modèle SAN et il est représenté par la formule tensorielle :

$$R = \sum_{j=1}^{N} \sum_{c=1}^{C_l^{(j)}} \bigotimes_{i=1}^{N}{}_g \breve{R}_{j,c}^{(i)} + \sum_{e \in \mathcal{E}_s} \sum_{c^{(1)}=1}^{C_e^{(1)}} \cdots \sum_{c^{(N)}=1}^{C_e^{(N)}} \bigotimes_{i=1}^{N}{}_g \breve{R}_{e,c^{(i)}}^{(i)} \quad \text{où } \breve{R}_{j,c}^{(i)} = \begin{cases} \breve{R}_{l,c}^{(i)} & \text{si } j = i \\ I_{|\mathcal{S}^{(i)}|} & \text{si } j \neq i \end{cases} \tag{5.1}$$

Le terme tensoriel de base qui nous intéresse est

$$\bigotimes_{i=1}^{N}{}_g \breve{R}^{(i)} \tag{5.2}$$

où les autres indices ont été omis (pour les matrices d'atteignabilité partielles) afin de simplifier la notation.

Définition 5.3.1. *Soit un terme tensoriel* $t = \bigotimes_{i=1}^{N}{}_g \breve{R}^{(i)}$*, ce terme tensoriel est classé comme :*

- **assimilé-constant***, si tous les éléments non-nuls des matrices d'atteignabilité partielles* $\breve{R}^{(i)}$ *de ce terme sont* assimilé-constants *(voir Définition 4.2.1, page 52) ;*
- **fonctionnel***, s'il y a au moins un élément* fonctionnel *(annulable) dans les matrices d'atteignabilité partielles* $\breve{R}^{(i)}$ *de ce terme (voir Définition 4.2.1, page 52).*

L'objectif de cette section est de montrer comment on peut générer un seul MDD associé aux éléments fonctionnels qui apparaissent dans les matrices d'un terme tensoriel du descripteur d'atteignabilité d'un modèle SAN.

Comme on l'a démontré dans le chapitre 4 (FIG. 4.1), le nombre de termes tensoriels d'un descripteur d'atteignabilité R est :

$$\sum_{i=1}^{N} C_l^{(i)} + \sum_{e \in \mathcal{E}_s} \prod_{i=1}^{N} C_e^{(i)} \tag{5.3}$$

où $C_l^{(i)}$ et $C_e^{(i)}$ sont les nombres de classes d'équivalence des éléments des matrices d'atteignabilité partielles $\breve{R}_l^{(i)}$ et $\breve{R}_e^{(i)}$ de l'automate $\mathcal{A}^{(i)}$ respectivement (voir Définition 4.3.3, page 56).

Les transitions entre les états globaux d'un modèle SAN provoqués par un événement (local ou synchronisant) peuvent être représentées dans *un ou plusieurs* termes tensoriels du descripteur d'atteignabilité. On parle de terme tensoriel (du descripteur d'atteignabilité) *associé* à un événement du modèle.

Définition 5.3.2. *Soit un événement $e \in \mathcal{E}$ d'un modèle SAN, on définit $\mathbb{T}_e$ l'ensemble des termes tensoriels du descripteur d'atteignabilité R d'un modèle SAN associés à l'événement e de la façon suivante :*

Étant donné la formule donnée dans la propriété 4.3.3 (page 63)

$$R = \sum_{j=1}^{N} \sum_{c=1}^{C_l^{(j)}} \bigotimes_{i=1}^{N}{}_g \ \breve{R}_{j,c}^{(i)} + \sum_{e \in \mathcal{E}_s} \sum_{c^{(1)}=1}^{C_e^{(1)}} \cdots \sum_{c^{(N)}=1}^{C_e^{(N)}} \bigotimes_{i=1}^{N}{}_g \ \breve{R}_{e,c^{(i)}}^{(i)}$$

On définit

– *pour $e \in \mathcal{E}_l$, les termes tensoriels associés à cet événement sont*
pour $j \in \mathcal{O}^{(e)}$ et tout indice $c \in [1..C_l^{(j)}]$ de classe d'équivalence

$$t_e = \bigotimes_{i=1}^{N}{}_g \ \breve{R}_{j,c}^{(i)} \in \mathbb{T}_e$$

– *pour $e \in \mathcal{E}_s$, les termes tensoriels associés à cet événement sont*
pour toutes les suites $(c^{(1)}, ..., c^{(N)}) \in \{\{1, ..., C_e^{(1)}\} \times \cdots \times \{1, ..., C_e^{(N)}\}\}$ les

$$t_e = \bigotimes_{i=1}^{N}{}_g \ \breve{R}_{e,c^{(i)}}^{(i)} \in \mathbb{T}_e$$

Il est intéressant de remarquer que les événements locaux d'une même classe d'équivalence d'un automate possèdent le même terme tensoriel associé à eux.

Définition 5.3.3. *L'ensemble des termes tensoriels $\mathbb{T}$ du descripteur d'atteignabilité R d'un modèle SAN est défini comme $\mathbb{T} = \bigcup_{e \in \mathcal{E}} \mathbb{T}_e$.*

Définition 5.3.4. *Pour un terme tensoriel t, $TFMdd_t$ est le MDD qui représente l'ensemble d'états pour lesquels l'évaluation de toutes les fonctions des éléments fonctionnels qui apparaissent dans les matrices d'atteignabilité du terme tensoriel $t \in \mathbb{T}$ est différente de zéro.*

Dans ALG. 5.3, on présente l'algorithme qui génère tous les $TFMdd$ du descripteur d'atteignabilité d'un modèle SAN.

✍ **Soit**

$\breve{R}_{t_e}^{(i)}$ la matrice d'atteignabilité correspondante à l'automate $\mathcal{A}^{(i)}$ du terme tensoriel $t_e \in \mathbb{T}$ associé à l'événement $e \in \mathcal{E}$;

FuncId($\breve{R}(x, y)$) la procédure qui rend l'identificateur de la fonction de l'élément fonctionnel de la matrice $\breve{R}$ de la ligne x et la colonne y ;

Intersect(M_1, M_2) la procédure qui réalise l'intersection entre les MDD M_1 et M_2.

On rappelle que ces termes tensoriels ont été fabriqués (voir Chapitre 4, Définitions 4.3.3 et 4.3.4 - page 56 - et Propriété 4.3.3 - page 63) de façon à ce que les fonctions d'une même matrice ont le même $FuncMdd$. C'est-à-dire que pour $t_e = \bigotimes_{i=1}^{N}{}_g \ \breve{R}_{t_e}^{(i)}$, les fonctions de la matrice $\breve{R}_{t_e}^{(i)}$ ont le même MDD noté $FuncMdd_{t_e}^{(i)}$, et donc ce qu'on cherche (l'espace où aucune $\breve{R}_{t_e}^{(i)}$ n'est nulle) est égale à l'intersection de tous les $FuncMdd_{t_e}^{(i)}$. Ce qui explique que dans ALG. 5.3 on ait (ligne 5) une boucle de 1 à N.

ALG. 5.3 TensorFuncMdd()

```
1:  for each e ∈ ℰ do
2:    for each t_e ∈ 𝕋 do
3:      /* Initialisation du MDD (TFMdd_{t_e}) associé à des fonctions du terme tensoriel t_e */
4:      TFMdd_{t_e} ← 1 ;
5:      for i = 1 to N do
6:        /* Initialisation de "found" pour vérifier si un élément fonctionnel a été trouvé dans la matrice R̆_{t_e}^{(i)} */
7:        found ← false ;
8:        /* Parcours des éléments de la matrice R̆_{t_e}^{(i)} du terme tensoriel t_e */
9:        for all x^{(i)} ∈ 𝒮^{(i)} do
10:         for all y^{(i)} ∈ 𝒮^{(i)} do
11:           if (R̆_{t_e}^{(i)}(x^{(i)}, y^{(i)}) est un élément fonctionnel) then
12:             /* Indique qu'un élément fonctionnel a été trouvé */
13:             found ← true ;
14:             TFMdd_{t_e} ← Intersect(TFMdd_{t_e}, FuncMdd_{t_e}^{(i)}) ;
15:             break ;   /* Sortie de la boucle : "for all y^{(i)} ∈ 𝒮^{(i)}" */
16:         end for
17:         if (found = true) then
18:           break ;   /* Sortie de la boucle : "for all x^{(i)} ∈ 𝒮^{(i)}" */
19:       end for
20:     end for
21:   end for
22: end for
```

Remarque : Étant donné un terme tensoriel *assimilé-constant* $t \in \mathbb{T}$, le MDD $TFMdd_t$ de ce terme est égal à $\mathbf{1}$, *i.e.*, $TFMdd_t = \hat{\mathcal{S}}$.

5.4 Génération symbolique traditionnelle

Avant de procéder plus avant, nous introduisons l'état de l'art en matière de génération symbolique d'espace d'états atteignables pour des modèles à états discrets généraux.

Définition 5.4.1. *Un modèle discret est défini par le t-uple* $(\hat{\mathcal{S}}, \mathcal{N}, \mathcal{S}^{init})$*, où*
- $\hat{\mathcal{S}}$ *est l'espace d'états produit (potentiels), i.e., l'ensemble d'états qui contient tous les états* $\tilde{x}$ potentiellement *valides pour le système modélisé ;*
- $\mathcal{N}$ *est la fonction "*next-state*" (*$\hat{\mathcal{S}} \rightarrow 2^{\hat{\mathcal{S}}}$*), i.e., où* $\mathcal{N}(\tilde{x})$ *détermine l'ensemble des états qui peuvent être atteignables à partir de l'état global* $\tilde{x}$ *;*
- $\mathcal{S}_{init}$ *est l'ensemble d'états initiaux, où* $\mathcal{S}_{init} \subseteq \hat{\mathcal{S}}$*.*

Un modèle à états discrets peut être vu comme un *graphe orienté*, où les *noeuds* du graphe sont les *états* du modèle et les *arcs* sont les *transitions* entre les états. D'une façon générale, les modèles décrits par des formalismes structurés de haut niveau ne sont pas directement représentés par un graphe, et le graphe peut être calculé à partir de la structure de haut niveau de la description du système.

Typiquement, les modèles exprimés par un formalisme structuré de haut niveau sont divisés en N sous-systèmes, où l'espace d'états produit $\hat{\mathcal{S}} = \mathcal{S}^{(N)} \times \cdots \times \mathcal{S}^{(1)}$ et $\mathcal{S}^{(i)}$ est l'espace d'états du i-ème sous-système. L'état (*global*) $\tilde{x}$ du système modélisé peut être décrit par le N-uple $(x^{(N)}, \ldots, x^{(1)})$ des états *locaux* de chaque sous-système.

La génération de l'espace d'états atteignables $\mathcal{S}$ a comme objectif de trouver, à partir d'un ensemble d'états initiaux, les états globaux *atteignables* de l'espace d'états $\hat{\mathcal{S}}$, où $\mathcal{S} \subseteq \hat{\mathcal{S}}$. Le travail pour trouver l'espace d'états atteignables n'est pas vraiment difficile, vu qu'il correspond à l'obtention d'un état atteignable suivant à partir d'un état donné. La génération *explicite* de l'espace d'états atteignables examine pour chaque état atteignable initial quel est son ensemble d'états atteignables, donc le temps et la complexité d'exécution de ce travail est linéaire en fonction du nombre d'états atteignables du système modélisé [30].

Les modèles de systèmes informatiques ont généralement un très grand nombre d'états, par exemple des modèles avec des milliard d'états. Ces modèles décrits par des formalismes structurés de haut niveau, parfois, peuvent être apparemment simples, *i.e.*, chaque sous-système a une structure simple et avec peu d'états. Cependant, la taille de l'espace d'états du système augmente exponentiellement en fonction du nombre de sous-systèmes du modèle. Un défi constant est d'élaborer des méthodes performantes capables de calculer l'espace d'états atteignables de modèles, de plus en plus rapidement et en utilisant le moins de mémoire.

Pour réaliser ce travail, on présente dans la section 5.4.2 (ALG. 5.5, page 81) l'*algorithme de parcours en largeur* (BFS - *Breadth-First Search*) qui calcule, à partir d'un ensemble d'états initiaux, l'espace d'états atteignables du modèle comme une clôture réflexive et transitive de la fonction "*next-state*" du modèle :

$$\mathcal{S} = \mathcal{S}_{init} \cup \mathcal{N}(\mathcal{S}_{init}) \cup \mathcal{N}^2(\mathcal{S}_{init}) \cup \ldots = \mathcal{N}^*(\mathcal{S}_{init}),$$

où la fonction "*next-state*" est étendue pour supporter comme paramètre des ensembles d'états, *i.e.*, $\mathcal{N}(\mathcal{X}) = \bigcup_{\tilde{x} \in \mathcal{X}} \mathcal{N}(\tilde{x})$.

5.4.1 MDD associés à des termes tensoriels

En pratique, pour utiliser l'algorithme BFS dans le contexte des SAN, il faut être capable de représenter la fonction "*next-state*" et des grands ensembles d'états d'une manière compacte dans la mémoire de l'ordinateur, de manipuler de façon performante les opérations (union et intersection) de ces structures, ainsi que d'appliquer directement la fonction "*next-state*" sur ces représentations.

D'abord, pour l'utilisation de l'algorithme BFS, il faut représenter les états de départ de transitions possibles représentées dans un terme tensoriel. Cet ensemble d'états dans un terme tensoriel t_e est représenté par un MDD noté par $EnableMdd_{t_e}$.

Définition 5.4.2. *Soit un événement $e \in \mathcal{E}$ d'un modèle SAN et un terme tensoriel $t_e \in \mathbb{T}$, on appelle $EnableMdd_{t_e}$ le sous-espace des $\tilde{x}$ de $\hat{\mathcal{S}}$ pour lequel il y a une entrée différente de zéro sur la ligne correspondante à $\tilde{x}$ dans le terme t_e. Ce sous-espace (représenté par un MDD) est l'ensemble des états de départ de transitions possibles représentées dans le terme tensoriel t_e.*

Définition 5.4.3. *Soit un événement $e \in \mathcal{E}$ d'un modèle SAN et un terme tensoriel $t_e \in \mathbb{T}$, on définit le MDD $TensorMdd_{t_e}$:*

- *t_e est um terme tensoriel **assimilé-constant** : $TensorMdd_{t_e} = EnableMdd_{t_e}$*
- *t_e est um terme tensoriel **fonctionnel** : $TensorMdd_{t_e} = EnableMdd_{t_e} \cap TFMdd_{t_e}$*

Dans ALG. 5.4, on présente l'algorithme pour générer le $TensorMdd_{t_e}$ de chaque terme tensoriel t_e du descripteur d'atteignabilité d'un modèle SAN.

ALG. 5.4 TensorMdd()

```
1:  for each e ∈ E do
2:    for each t_e ∈ T do
3:      EnableMdd_{t_e} ← 1 ;    /* Initialisation du MDD EnableMdd_{t_e} */
4:      for l = 1 to N do
5:        if (R̆_{t_e}^{(l)} n'est pas une matrice identité) then
6:          p ← NewNode(l) ;    /* Création d'un noeud p au niveau l */
7:          /* Parcours des éléments de la matrice R̆_{t_e}^{(l)} du terme tensoriel t_e */
8:          for all x^{(l)} ∈ S^{(l)} do
9:            for all y^{(l)} ∈ S^{(l)} do
10:             if (R̆_{t_e}^{(l)}(x^{(l)}, y^{(l)}) n'est pas nul) then
11:               p[x^{(l)}] ← 1 ;
12:           end for
13:         end for
14:         EnableMdd_{t_e} ← Intersect(EnableMdd_{t_e}, p) ;
15:     end for
16:     if (t_e est un terme tensoriel fonctionnel) then
17:       TensorMdd_{t_e} ← Intersect(EnableMdd_{t_e}, TFMdd_{t_e}) ;
18:     else
19:       TensorMdd_{t_e} ← EnableMdd_{t_e} ;
20:   end for
21: end for
```

5.4.2 Algorithme de Parcours en Largeur (BFS) pour le calcul du RSS

Dans ALG. 5.5, on présente l'algorithme BFS qui parcourt et manipule des ensembles d'états au lieu de manipuler le modèle état-par-état explicitement. Étant donné un état initial, l'idée de l'algorithme est de chercher de nouveaux états atteignables à partir des transitions représentées dans les matrices des termes tensoriels du descripteur d'atteignabilité du modèle. L'algorithme BFS calcule dans $Mdd_{\mathcal{S}}$, l'espace d'états atteignables du modèle.

✍ **Soit**

$Mdd_{\mathcal{S}}$	le MDD qui représente l'espace d'états atteignables $\mathcal{S}$ du modèle ;
$Enabled$	le MDD qui représente l'ensemble d'états de départ de transitions possibles pour un terme tensoriel ;
$Reached$	le MDD qui représente l'ensemble d'états qui sont atteignables à partir de $Enabled$ par l'occurrence des transitions représentées dans les matrices d'un terme tensoriel ;
RootNode(M)	la procédure qui rend le noeud racine d'un MDD.

ALG. 5.5 BFS(*in* $\tilde{x}_{init}$: *state*)

```
1:  Mdd_S ← StateToMdd(x̃_init) ;   /* Initialisation de Mdd_S avec l'état initial x̃_init */
2:  repeat
3:    Old ← Mdd_S ;   /* Conserve dans Old la valeur courant de Mdd_S */
4:    for each e ∈ E do
5:      for each t_e ∈ T do
6:        /* Enabled représente les états de départ qui se trouvent dans Mdd_S (l'espace d'états atteignables courant)
          des transitions représentées dans les matrices de t_e */
7:        Enabled ← Intersect(Mdd_S, TensorMdd_te) ;
8:        /* Si Enabled ≠ ∅, alors il y a au moins un état dans Mdd_S qui est un état de départ des transitions représentées
          dans t_e */
9:        if (Enabled ≠ ∅) then
10:         /* Reached représente les états atteignables par l'occurrence des transitions de t_e à partir de Enabled */
11:         Reached ← Fire(t_e, RootNode(Enabled)) ;
12:         /* Ajoute les états atteignables (Reached) à Mdd_S */
13:         Mdd_S ← Union(Mdd_S, Reached) ;
14:     end for
15:   end for
16: until (Old = Mdd_S)
```

Dans ALG. 5.5, à partir de l'ensemble d'états courant ($Mdd_{\mathcal{S}}$), on examine quels sont les états de $TensorMdd_{t_e}$ qui permettent l'occurrence des transitions représentées dans le terme tensoriel t_e provenant d'un événement e du modèle. En utilisant l'algorithme *Fire* (ALG. 5.6), on trouve les états qui sont atteignables à partir de $Enabled$ par l'occurrence des transitions représentées dans le terme t_e.

On ajoute les états atteignables ($Reached$) à l'ensemble d'états atteignables courant ($Mdd_{\mathcal{S}}$) afin de trouver ultérieurement d'autres nouveaux états atteignables. L'espace d'états atteignables d'un modèle est considéré dans son état final au moment où aucun autre état atteignable n'est trouvé à partir du $Mdd_{\mathcal{S}}$ courant par l'occurrence des transitions représentées dans les matrices de t_e.

Dans ALG. 5.6, on présente l'algorithme pour la procédure *Fire*. Cet algorithme utilise deux paramètres : t un terme tensoriel, et p un noeud du MDD qui représente les états de départ des transitions représentées dans t, états qui se trouvent dans $Mdd_{\mathcal{S}}$. L'idée de cet algorithme est de retourner le noeud "racine" du MDD qui représente l'ensemble d'états qui sont atteignables de p par l'occurrence des transitions représentées dans t.

Définition 5.4.4. *Soit un événement $e \in \mathcal{E}$ d'un modèle SAN, on définit $\mathcal{O}^{(f_{t_e})}$ l'ensemble des indices d'automates dont l'état est argument des fonctions qui apparaissent dans les matrices du terme tensoriel $t_e \in \mathbb{T}$.*

Définition 5.4.5. *Soit un terme tensoriel t_e associé à un événement $e \in \mathcal{E}$ d'un modèle SAN. L'indice $l \in [1..N]$ d'un automate ($\mathcal{A}^{(l)}$) est concerné par le terme t_e si et seulement si $l \in \mathcal{O}^{(e)} \cup \mathcal{O}^{(f_{t_e})}$.*

Définition 5.4.6. *Bot(t_e) est défini comme l'indice minimum de l'ensemble $\mathcal{O}^{(e)} \cup \mathcal{O}^{(f_{t_e})}$, i.e., l'indice le plus bas des automates concernés par le terme tensoriel t_e provenant d'un événement $e \in \mathcal{E}$.*

Définition 5.4.7. *Top(t_e) est défini comme l'indice maximum de l'ensemble $\mathcal{O}^{(e)} \cup \mathcal{O}^{(f_{t_e})}$, i.e., l'indice le plus élevé des automates concernés par le terme tensoriel t_e provenant d'un événement $e \in \mathcal{E}$.*

Remarque : Soit un noeud p au niveau l d'un MDD, les arcs sortants du noeud indiquent quels sont les états représentés par ce noeud, *i.e.*, si $x^{(l)} \in \mathcal{S}^{(l)}$ est représenté par le noeud p, alors $p[x^{(l)}] \neq \emptyset$.

La procédure *Fire*, présentée dans ALG. 5.6, est un algorithme récursif qui examine l'occurrence des transitions possibles représentées dans le terme t à partir des états de départ représentés dans le noeud p. On examine quels sont les états (locaux) de départ (qui sont représentés par le noeud p) et les transitions représentées dans les matrices de t.

Dans cet algorithme, chaque fois qu'un noeud p est exploré, on crée un noeud r résultant (ligne 7) afin de stocker les états d'arrivée des transitions représentées dans t à partir d'états de départ représentés par p. Pour chaque état (de départ) $x^{(l)}$ non-nul de p, on examine *localement* les transitions représentées dans les matrices de t afin d'obtenir les états (d'arrivée) $y^{(l)}$. Grâce à la décomposition des matrices proposée dans le chapitre 4 du descripteur d'atteignabilité, on peut assurer, en utilisant la propriété 5.2.1, que si $x^{(l)}$ est un état de départ d'une transition dans la matrice $\breve{R}_t^{(l)}$ d'un terme tensoriel t, alors tous éléments (assimilé-constants ou fonctionnels) qui se trouvent dans la ligne $x^{(l)}$ de $\breve{R}_t^{(l)}$ sont états d'arrivée de ces transitions (lignes 16 - 20).

La condition $l < Bot(t)$ est la condition de terminaison de l'algorithme récursif *Fire*, sachant que pour tout automate d'indice l, où $Bot(t) > l \geq 1$, la matrice d'atteignabilité $\breve{R}_t^{(l)}$ est une matrice identité.

Ensuite, on note les variables les plus significatives utilisées dans la procédure *Fire* (ALG. 5.6).

✍ **Notation**

t	un terme tensoriel du descripteur d'atteignabilité ;
p	le noeud du MDD courant de l'espace d'états atteignables, dont les étiquettes (valeurs) d'arcs sont analysées. Au premier appel le noeud p est le noeud racine du MDD $Enabled$ et $l = N$;
r	le noeud du MDD qui stocke les nouveaux états atteignables.

On rappelle que ALG. 5.6 va créer un MDD qui est l'ensemble des états atteignables par des transitions d'un terme tensoriel t à partir des états atteignables courants que l'on a restreint aux états de départ des transitions de t.

ALG. 5.6 Fire(*in* t : *tensor term*, *in* p : *node*) : *node*

```
1:  /* Conserve dans l la valeur courant de p.lvl. Au premier appel l = N */
2:  l ← p.lvl ;
3:  /* Les états des automates d'indice l < Bot(t) (i.e., qui ne sont pas concernés par le terme t) ne sont pas modifiés par
    l'occurrence des transitions de t. C'est la condition de terminaison de l'algorithme récursif */
4:  if (l < Bot(t)) then
5:     return p ;
6:  /* Création d'un noeud r au niveau l qui va stocker les nouveaux états atteignables */
7:  r ← NewNode(l) ;
8:  for all x^(l) ∈ S^(l) do
9:     /* Si x^(l) est un état local courant à considérer */
10:    if (p[x^(l)] ≠ ∅) then
11:       /* Le noeud q sera le noeud du sous-arbre des états atteignables au niveau l − 1 */
12:       q ← Fire(t, p[x^(l)]) ;
13:       /* Si x^(l) est un état de départ, alors tout y^(l) ∈ S^(l), tel que R̆_t^(l)(x^(l), y^(l)) est non-nul
14:       est un état d'arrivée (Propriété 5.2.1) */
15:       /* Construction du noeud r au niveau l */
16:       for all y^(l) ∈ S^(l) do
17:          if (R̆_t^(l)(x^(l), y^(l)) n'est pas nul) then
18:             /* On construit les sous-arbres des états atteignables du noeud r au niveau l */
19:             r[y^(l)] ← Union(q, r[y^(l)]) ;
20:       end for
21: end for
22: return r ;
```

En utilisant l'algorithme BFS avec des espaces d'états représentés par des MDD, il est possible d'obtenir des résultats très probants en ce qui concerne l'utilisation de la mémoire, ainsi que le temps d'exécution pour la génération de l'espace d'états [94]. Toutefois, bien que l'espace d'états représenté par un MDD utilise très peu de mémoire, la quantité de mémoire utilisée par les ensembles d'états pendant l'exécution de l'algorithme peut augmenter très vite. Ce problème provient du résultat des opérations sur les noeuds des MDD, *i.e.*, chaque fois que l'ensemble d'états est actualisé, des nouveaux noeuds du MDD sont ajoutés par l'opération union, plutôt que modifier les noeuds existants.

5.4.3 Bilan de la génération symbolique traditionnelle

L'utilisation de l'algorithme BFS pour le calcul du RSS de modèles qui utilisent des fonctions demande la génération des MDD $TensorMdd$ (ALG. 5.4) associés aux termes tensoriels du descripteur d'atteignabilité du modèle.

Pour les modèles qui n'utilisent pas de fonctions - ou qui utilisent des fonctions dont les arguments de ces fonctions sont limités à l'espace d'états local d'un sous-système, *e.g.*, des modèles décrits par le formalisme des Réseaux de Petri Stochastiques (SPN - *Stochastic Petri Nets*) - la génération d'un MDD $TensorMdd$ est faite en analysant les éléments non-nuls des matrices de ce terme : dans ce cas le MDD

$TensorMdd$ du terme tensoriel est égal au MDD $EnableMdd$ de ce terme. Toutefois, pour les modèles qui utilisent des fonctions sans restriction sur les arguments, il faut tenir compte de l'utilisation du MDD $TFMdd$ (ALG. 5.3), *i.e.*, du MDD associé à des fonctions du terme tensoriel. Dans ce cas, le MDD pertinent, $TensorMdd$ est l'intersection des MDD $TFMdd$ et $EnableMdd$.

En comparant les algorithmes ALG. 5.5 et ALG. 5.6 de cette thèse et les algorithmes pour la génération du RSS de modèles décrits par le formalisme SPN [94], le principe général des algorithmes est le même. Par rapport à la génération du RSS présentée dans [94], l'élément nouveau est *l'ajustement des sous-espaces* (modélisé par les MDD $TFMdd$ et $EnableMdd$) *qu'il faut ajouter à chaque étape de la construction du RSS*. Cette génération des MDD $TFMdd$ (ALG. 5.3), ainsi que la génération des MDD $FuncMdd$ (ALG. 5.1) est *l'idée originale pour la génération du RSS des modèles qui utilisent des fonctions de cette thèse*.

5.5 Génération basée sur la saturation

L'algorithme basé sur la saturation (présenté dans cette section) pour la génération de l'espace d'états atteignables de modèles de systèmes concurrents basés sur des événements repose sur la représentation d'états atteignables par des MDD et la fonction "*next-state*" par des produits de Kronecker. D'autres méthodes de génération basée sur la saturation ont déjà été proposées pour des modèles qui ne représentent pas la fonction "*next-state*" par des produits de Kronecker [93, 34, 133]. Toutefois, dans cette thèse, on s'intéresse à une représentation de Kronecker de la fonction "*next-state*" qui est bien adaptée aux modèles décrits par le formalisme SAN.

Un élément important est le stockage de la fonction "*next-state*". Pour les modèles de systèmes complexes, parfois, il n'y a pas de façon compacte et performante pour stocker le MDD qui représente la fonction "*next-state*". Des méthodes [24] utilisant des partitions conjonctives ou disjonctives, pour la fonction "*next-state*", sont employées pour optimiser l'utilisation de cette fonction. En fait, pour les modèles de systèmes synchrones avec N sous-systèmes, la fonction "*next-state*" peut être exprimée par la conjonction :

$$\mathcal{N}(\tilde{x}) = \mathcal{N}_N(\tilde{x}) \times \cdots \times \mathcal{N}_1(\tilde{x}),$$

où la fonction "*next-state*" $\mathcal{N}_l(\tilde{x})$ indique le changement de l'état local $x^{(l)}$ (l'état local du l-ième sous-système) de l'état global $\tilde{x}$. De la même façon, pour les modèles de systèmes asynchrones, où il y a seulement un sous-système qui change son état local à la fois, la fonction "*next-state*" peut être exprimée par la disjonction :

$$\mathcal{N}(\tilde{x}) = \bigcup_{l=1}^{N} \mathcal{I}_N \times \cdots \times \mathcal{I}_{l+1} \times \mathcal{N}_l(\tilde{x}) \times \mathcal{I}_{l-1} \times \cdots \times \mathcal{I}_1,$$

où $\mathcal{I}_g$ ($g \neq l$) indique que l'état local du g-ième sous-système reste le même. En effet, pour les modèles de systèmes concurrents (*e.g.*, modèles décrits par les formalismes des Réseaux de Petri Stochastiques ou Réseaux de Automates Stochastiques), les événements synchronisants du modèle se produisent normalement dans un petit ensemble de sous-systèmes. Cette *localité* de l'événement est déjà utilisée dans l'algorithme BFS (*i.e.*, seulement pour les niveaux concernés par un terme tensoriel t) et elle est utilisée de façon très performante dans l'algorithme de Saturation (voir la section 5.5.4, page 90).

Dans cette section, on utilise plusieurs fonctions "*next-state*" *locales* au lieu d'une seule fonction "*next-state*" *globale* pour obtenir l'espace d'états atteignables d'un modèle. Grâce à cette démarche, il est possible de manipuler *localement* les structures de donnés, éliminant ainsi des surcoûts en terme de calcul. De même, le tirage d'un événement est considéré comme une opération de *poids faible* à cause de la *localité* (Définition 5.5.3, page 86) de l'événement, puisque pour la grande partie des modèles structurés, le tirage d'événements change l'état local de seulement quelques composants (sous-systèmes) du modèle.

Les opérations (par exemple, l'union ou l'intersection) sur les noeuds d'un MDD déterminent la création des nouveaux noeuds. Afin de réduire le coût de calcul de ces opérations, la génération basée sur la saturation réalise de façon *directe* ("*sur place*") le traitement des noeuds d'un MDD. On dit qu'un noeud d'un MDD est actualisé "sur place" quand la valeur de ses arcs est modifiée sans que soit créé un nouveau noeud. Cette démarche originale pour la manipulation directe des noeuds d'un MDD fournit des gains de performance importants, autant en réduction de l'utilisation de mémoire qu'en coût de calcul.

5.5.1 Représentation de Kronecker de la fonction "*next-state*"

Pour profiter de la structure modulaire des formalismes structurés de haut-niveau pour la génération de l'espace d'états atteignables, on utilise des *expressions de Kronecker avec des matrices d'atteignabilité* pour représenter les fonctions "*next-state*" d'un modèle. Dans ce contexte, on peut profiter de la localité des événements de modèles structurés pour l'algorithme de génération symbolique de l'espace d'états.

Les travaux précédents, initialisés dans [90], représentent la fonction "*next-state*" de modèles structurés par des diagrammes de décision. Ainsi, on représente la fonction "*next-state*" dans un format disjonctif-conjonctif organisé par terme tensoriel t du descripteur d'atteignabilité et sous-système l. Ce format est proche d'un ensemble de $N \times |\mathbb{T}|$ matrices, où N est le nombre de sous-systèmes et $|\mathbb{T}|$ est le nombre de termes tensoriels du descripteur d'atteignabilité du modèle.

Ce format de la fonction "*next-state*" est inspiré des méthodes qui représentent d'une façon compacte la matrice de transition d'une chaîne de Markov à échelle de temps continue en utilisant une somme de *produits de Kronecker* [111]. L'utilisation de cette représentation de Kronecker pour la génération de l'espace d'états a été initialement proposé par [83], mais pour la génération *explicite* de l'espace d'états.

Remarque : La fonction "*next-state*" $\mathcal{N}$ d'un modèle SAN (où $\hat{\mathcal{S}} = \mathcal{S}^{(N)} \times \cdots \times \mathcal{S}^{(1)}$ et $n_l = |\mathcal{S}^{(l)}|$) est décomposée par terme tensoriel $t \in \mathbb{T}$ du descripteur d'atteignabilité R, *i.e.*, $\mathcal{N} = \bigcup_{t \in \mathbb{T}} \mathcal{N}_t$.

Pour obtenir une représentation de Kronecker de la fonction "*next-state*", $\mathcal{N}_t$ est représentée par une matrice carrée $\mathbf{N}_t$ de taille $|\hat{\mathcal{S}}|$, où $\mathbf{N}_t[\tilde{x}, \tilde{y}] \neq 0$ si et seulement si $\tilde{y} \in \mathcal{N}_t(\tilde{x})$ et les états $\tilde{x}$ et $\tilde{y}$ (utilisés comme des indices d'états globaux) sont interprétés comme des nombres naturels de base mixte $\tilde{x} = \sum_{N \geq l \geq 1} x^{(l)} \cdot \prod_{g=1}^{l-1} n_g$ et $\tilde{y} = \sum_{N \geq l \geq 1} y^{(l)} \cdot \prod_{g=1}^{l-1} n_g$ respectivement. Toutefois, la matrice $\mathbf{N}_t$ n'est pas stockée de façon pleine, mais elle est représenté par un *produit généralisé de Kronecker* $\mathbf{N}_t = \mathbf{N}_{N,t} \underset{g}{\otimes} \cdots \underset{g}{\otimes} \mathbf{N}_{1,t}$, où chaque matrice carrée $\mathbf{N}_{l,t}$ de taille n_l est une matrice d'atteignabilité $\breve{R}_t^{(l)}$ du descripteur d'atteignabilité du modèle.

Définition 5.5.1. *On note $\mathcal{N}_{l,t}$ la fonction "*next-state*" (locale) du terme tensoriel $t \in \mathbb{T}$ pour le l-ième automate ($l \in [1..N]$). On dit qu'un état local $y^{(l)}$ est atteignable à partir d'un état local $x^{(l)}$ si et seulement si l'élément de la matrice d'atteignabilité $\check{R}_t^{(l)}$ de la ligne $x^{(l)}$ et la colonne $y^{(l)}$ est différent de zéro, i.e., $y^{(l)} \in \mathcal{N}_{l,t}(x^{(l)}) \Leftrightarrow \check{R}_t^{(l)}(x^{(l)}, y^{(l)}) \neq 0$.*

Définition 5.5.2. *Soit un terme tensoriel $t \in \mathbb{T}$ du descripteur d'atteignabilité d'un modèle SAN, la fonction "*next-state*" $\mathcal{N}_t$ de ce terme est représentée par le produit de N fonctions locales, i.e., $\mathcal{N}_t = \mathcal{N}_{N,t} \times \cdots \times \mathcal{N}_{1,t}$, où $\mathcal{N}_{l,t} : \mathcal{S}^{(l)} \rightarrow 2^{\mathcal{S}^{(l)}}$ pour $N \geq l \geq 1$.*

Comme on l'a dit précédemment, la localité d'un événement est un concept important pour la performance de l'algorithme de la Saturation. Ce concept peut aussi être appliqué sur un terme tensoriel t_e provenant de tout événement e du modèle SAN.

Définition 5.5.3. *Soit un terme tensoriel $t_e \in \mathbb{T}$ du descripteur d'atteignabilité provenant d'un événement $e \in \mathcal{E}$ d'un modèle SAN. On définit la* localité *de t_e par l'ensemble d'indices d'automates qui se trouvent entre le Top(t_e) et le Bot(t_e) de ce terme, i.e., l'ensemble d'indices d'automates $l \in [1..N]$ tel que Top(t_e) $\geq l \geq$ Bot(t_e).*

Remarque : Soit un événement $e \in \mathcal{E}$ d'un modèle SAN et un terme tensoriel $t_e \in \mathbb{T}$, étant donné un indice l d'un automate tel que $l \notin \mathcal{O}^{(e)}$, alors la fonction "*next-state*" locale $\mathcal{N}_{l,t_e}(x^{(l)}) = x^{(l)}$ pour tous $x^{(l)} \in \mathcal{S}^{(l)}$, *i.e.*, $\mathbf{N}_{l,t_e}$ est une matrice identité de taille n_l.

5.5.2 L'idée de noeud saturé

Pour la génération basée sur la saturation, on présente un nouvel ordre de parcours des termes tensoriels du descripteur d'atteignabilité basé sur le tirage exhaustif de tous les événements représentés par ces termes qui affectent un noeud quelconque du MDD (et ses fils, *i.e.*, les noeuds aux niveaux inférieurs) de telle façon que ce noeud trouve son format final : *saturé*. L'analyse des noeuds est considérée *de bas en haut*, *i.e.*, lorsque un noeud est analysé, tous ses fils sont déjà analysés (saturés).

On étend quelques définitions de la fonction "*next-state*" afin de les utiliser dans la génération basée sur la saturation de l'espace d'états atteignables de modèles SAN.

✍ **Soit**

$\mathcal{N}_{l:g,t}$	la fonction "*next-state*" du terme tensoriel $t \in \mathbb{T}$ pour les automates de l à g représentée par le produit des fonctions "*next-state*" locales de ces automates, *i.e.*, $\mathcal{N}_{l:g,t} = \mathcal{N}_{l,t} \times \cdots \times \mathcal{N}_{g,t}$, pour $N \geq l \geq g \geq 1$;
$\mathcal{N}_{l:g,t}(\mathcal{X})$	la fonction "*next-state*" du terme tensoriel $t \in \mathbb{T}$ pour les automates de l à g appliquée à l'ensemble d'états $\mathcal{X}$, *i.e.*, $\mathcal{N}_{l:g,t}(\mathcal{X}) = \bigcup_{\tilde{x} \in \mathcal{X}} \mathcal{N}_{l:g,t}(\tilde{x})$, où $\mathcal{X} \subseteq \mathcal{S}^{(l)} \times \cdots \times \mathcal{S}^{(g)}$ et $N \geq l \geq g \geq 1$;
$\mathcal{N}_{l:g,\mathbb{T}'}(\mathcal{X})$	la fonction "*next-state*" de l'ensemble des termes tensoriels $\mathbb{T}' \subseteq \mathbb{T}$ pour les automates de l à g appliquée à l'ensemble d'états $\mathcal{X}$, *i.e.*, $\mathcal{N}_{l:g,\mathbb{T}'}(\mathcal{X}) = \bigcup_{t \in \mathbb{T}'} \mathcal{N}_{l:g,t}(\mathcal{X})$, où $\mathbb{T}' \subseteq \mathbb{T}$, $\mathcal{X} \subseteq \mathcal{S}^{(l)} \times \cdots \times \mathcal{S}^{(g)}$ et $N \geq l \geq g \geq 1$;
$\mathcal{N}_{\leq l}$	une façon plus courte de représenter $\mathcal{N}_{l:1,\{t \in \mathbb{T}:Top(t) \leq l\}}$.

Définition 5.5.4. *Soit un noeud p au niveau l d'un MDD, ce noeud est* **saturé** *si à partir des états (de départ) représentés par ce noeud le tirage d'un événement quelconque du modèle ne conduit pas à un état inexploré. On peut dire encore que l'ensemble d'états $\mathcal{B}(p)$ d'un noeud p ne change plus après le tirage d'un événement, i.e., que $\mathcal{B}(p) = \mathcal{N}^*_{\leq l}(\mathcal{B}(p))$ est valide.*

On peut facilement démontrer par l'absurde que, si le noeud p est saturé, tous les noeuds parcourus entre le noeud p et $\mathbf{1}$ doivent aussi être saturés [30].

L'algorithme de Saturation est en moyenne de plusieurs ordres de grandeur plus performant en temps *et* utilisation de mémoire [30] que les algorithmes traditionnels de génération basés sur des MDD. Notamment, le *pic de mémoire* utilisée est usuellement plus proche de la quantité de mémoire finale utilisée pour représenter l'espace d'états.

En comparant l'algorithme de Saturation avec les précédents algorithmes de génération de l'espace d'états qui utilisent des MDD [31, 94], on vérifie qu'il élimine des surcoûts en terme de calcul, réduit le nombre moyen de tirages d'événements, et permet une utilisation plus simple et performante du *cache*.

5.5.3 Saturation avec des taux et probabilités fonctionnels

Comme on l'a dit précédemment, la saturation d'un noeud au niveau l du MDD par un terme tensoriel t_e consiste à considérer toutes les transitions possibles représentées dans les matrices de ce terme tensoriel, transitions qui ont comme état de départ un $\tilde{x}$ dont la l-ième composante $x^{(l)}$ est l'étiquette d'un arc sortant de ce noeud.

L'élément nouveau, par rapport à l'état d'art, est que dans les modèles SAN, le taux d'occurrence d'un événement, ainsi que la probabilité de routage d'une transition lors de l'occurrence d'un événement peuvent être exprimés par une *fonction* (sans hypothèse concernant les arguments - les travaux précédents de l'algorithme de Saturation [32, 30] considèrent des fonctions avec des arguments locaux au sous-système).

Définition 5.5.5. *Soit une matrice d'atteignabilité $\breve{R}_{t_e}$ du descripteur d'atteignabilité du terme tensoriel $t_e \in \mathbb{T}$ associé à l'événement $e \in \mathcal{E}$ d'un modèle SAN, si $\breve{R}_{t_e}(x, y) \neq 0$, alors on dira que la transition de l'état x vers l'état y est représentée dans la matrice $\breve{R}_{t_e}$.*

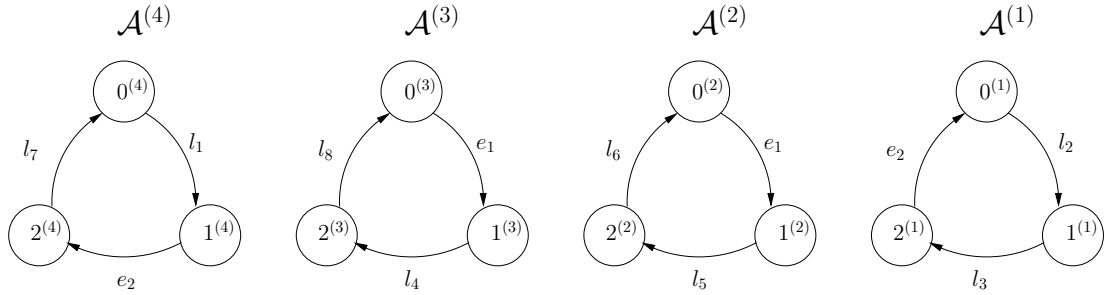
Définition 5.5.6. *Soit un terme tensoriel* **assimilé-constant** $t_e \in \mathbb{T}$ *du descripteur d'atteignabilité associé à l'événement $e \in \mathcal{E}$ d'un modèle SAN, l'état global $\tilde{x} \in \hat{\mathcal{S}}$ est un état de départ d'au moins une transition représentée dans les matrices de t_e si la fonction "*next-state*" à partir de $\tilde{x}$ n'est pas vide, i.e., si $\mathcal{N}_{t_e}(\tilde{x}) \neq \emptyset$.*

Pour les modèles SAN qui utilisent seulement des éléments *constants* (taux ou probabilités), il est facile de déterminer si un état global $\tilde{x} \in \hat{\mathcal{S}}$ est un état de départ d'une transition globale $\tilde{\mathcal{Q}}(\tilde{x}, \tilde{y})$ du modèle (voir Définition 2.2.11, page 28).

En considérant la *localité* (Définition 5.5.3, page 86) d'un terme tensoriel assimilé-constant t_e et que la fonction "*next-state*" de t_e peut être représentée par le produit de N fonctions locales (Définition 5.5.2,

page 86), pour déterminer si l'état global $\tilde{x}$ est un état de départ d'au moins une transition représentée dans les matrices de t_e, il suffit que la fonction "*next-state*" vérifie $\mathcal{N}_{l:g,t_e}(\tilde{x}) \neq \emptyset$ (où $l = \text{Top}(t_e)$ et $g = \text{Bot}(t_e)$), sachant que pour tout automate d'indice $i > \text{Top}(t_e)$ et $i < \text{Bot}(t_e)$, $\mathcal{N}_{i,t_e}(x^{(i)}) = x^{(i)}$.

Par exemple, soit l'état global $\tilde{x} = (x^{(4)}, x^{(3)}, x^{(2)}, x^{(1)})$ du modèle SAN décrit dans FIG. 5.3, pour savoir si $\tilde{x} = (1, 0, 0, 2)$ est un état de départ d'une transition du terme tensoriel t_{e_1} associé à l'événement e_1, il faut que $\mathcal{N}_{l,e_1}(x^{(l)})$ ne soit pas vide pour tout $l \in \mathcal{O}^{(e_1)} = \{3, 2\}$, *i.e.*, $\mathcal{N}_{3,e_1}(0) \neq \emptyset$ et $\mathcal{N}_{2,e_1}(0) \neq \emptyset$, ce qui est le cas ici.



Type	Even.	Taux	Type	Even.	Taux
loc	l_1	α_1	loc	l_2	α_2
loc	l_3	α_3	loc	l_4	α_4
loc	l_5	α_5	loc	l_6	α_6
loc	l_7	α_7	loc	l_8	α_8
syn	e_1	μ_1	syn	e_2	μ_2

FIG. 5.3 – Modèle SAN (taux constants)

$$\underbrace{\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}}_{\breve{R}_{e_1}^{(4)}} \otimes_g \underbrace{\begin{vmatrix} 0 & \mu_1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}}_{\breve{R}_{e_1}^{(3)}} \otimes_g \underbrace{\begin{vmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}}_{\breve{R}_{e_1}^{(2)}} \otimes_g \underbrace{\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}}_{\breve{R}_{e_1}^{(1)}}$$

FIG. 5.4 – Matrices du terme tensoriel t_{e_1} de FIG. 5.3

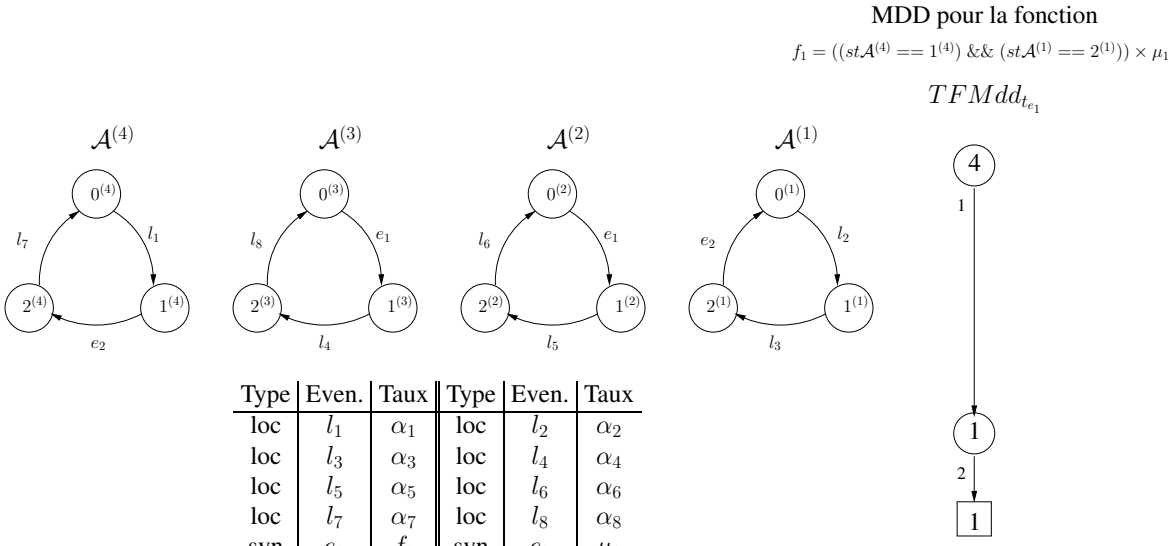
Toutefois, pour les termes tensoriels qui ont des éléments fonctionnels qui s'annulent (*i.e.*, termes tensoriels *fonctionnels*), il faut aussi prendre compte cette possibilité d'annulation.

Définition 5.5.7. *Soit un terme tensoriel* **fonctionnel** $t_e \in \mathbb{T}$ *du descripteur d'atteignabilité associé à l'événement* $e \in \mathcal{E}$ *d'un modèle SAN, l'état global* $\tilde{x} \in \hat{\mathcal{S}}$ *est un état de départ d'au moins une transition représentée dans les matrices de* t_e *si :*

- $\forall l \in \mathcal{O}^{(e)}$, $\mathcal{N}_{l,t_e}(x^{(l)}) \neq \emptyset$
- *et* $\tilde{x} \in TFMdd_{t_e}$

Remarque : Pour vérifier que $\tilde{x} \in TFMdd_{t_e}$, étant donné l'indice de niveau $g_{max} = max\ \mathcal{O}^{(f_{t_e})}$ du $TFMdd_{t_e}$, il suffit que le sous-vecteur d'états $\tilde{x}^{(\mathcal{O}^{(f_{t_e})})}$ de $\tilde{x}$ (*i.e.*, le vecteur des états locaux $x^{(i)}$ tel que $i \in \mathcal{O}^{(f_{t_e})}$) soit une suite d'étiquette dans un chemin du $TFMdd_{t_e}$ allant du niveau g_{max} au noeud terminal **1** au niveau 0.

Par exemple, soit l'état global $\tilde{x} = (x^{(4)}, x^{(3)}, x^{(2)}, x^{(1)})$ du modèle SAN décrit dans FIG. 5.5, pour savoir si $\tilde{x} = (1, 0, 0, 2)$ est un état de départ d'une transition du terme tensoriel t_{e_1} associé à l'événement e_1, outre que $\mathcal{N}_{l,e_1}(x^{(l)})$ ne soit pas vide pour tout $l \in \mathcal{O}^{(e_1)}$, il faut aussi que le résultat de l'évaluation de la fonction f_1 pour l'état $\tilde{x}$ soit différente de zéro.



Type	Even.	Taux	Type	Even.	Taux
loc	l_1	α_1	loc	l_2	α_2
loc	l_3	α_3	loc	l_4	α_4
loc	l_5	α_5	loc	l_6	α_6
loc	l_7	α_7	loc	l_8	α_8
syn	e_1	f_1	syn	e_2	μ_2

FIG. 5.5 – Modèle SAN (fonctionnel)

$$\breve{R}_{e_1}^{(4)} \quad \breve{R}_{e_1}^{(3)} \quad \breve{R}_{e_1}^{(2)} \quad \breve{R}_{e_1}^{(1)}$$

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \bigotimes_g \begin{vmatrix} 0 & f_1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} \bigotimes_g \begin{vmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} \bigotimes_g \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

FIG. 5.6 – Matrices du terme tensoriel t_{e_1} de FIG. 5.5

L'ensemble d'états pour lesquels l'évaluation de la fonction f_1 est différente de zéro est représenté par le MDD $TFMdd_{t_{e_1}}$. En considérant le MDD $TFMdd_{t_{e_1}}$, pour analyser si le résultat de la fonction f_1 est différente de zéro, on examine s'il est possible de parcourir un chemin à partir du noeud racine de $TFMdd_{t_{e_1}}$ jusqu'au noeud terminal **1** dont les arcs sont étiquetés par les états locaux $x^{(4)} = 1$ et $x^{(1)} = 2$, qui sont les arguments de la fonction f_1.

5.5.4 Algorithme de Saturation pour le calcul du RSS

Étant donné un état global initial $\tilde{x} \in \mathcal{S}$ d'un modèle SAN, on peut générer l'espace d'états atteignables (RSS) de ce modèle représenté par le MDD $Mdd_{\mathcal{S}}$ via l'exécution de l'algorithme de Saturation :

$$Mdd_{\mathcal{S}} \leftarrow \text{Generate}(\tilde{x})$$

L'algorithme de Saturation possède en fait quatre procédures principales : *Generate*, *Locals*, *Saturate* et *Fire*. Ensuite, on détaille ces quatre procédures de l'algorithme de Saturation.

Les procédures *Saturate* et *Fire* sont mutuellement récursives. La saturation d'un noeud implique à la recherche de nouveaux états, d'où *Saturate* exécute *Fire*. La récursion mutuelle est nécessaire pour permettre que les noeuds fils (du noeud qui est saturé) soient aussi saturés ; chaque nouveau noeud créé pendant l'exécution de *Fire* sera immédiatement saturé via l'exécution de *Saturate*. La preuve (*proof of correctness*) de l'algorithme de Saturation peut être trouvée dans [30].

5.5.4.1 La procédure *Generate*

Dans ALG. 5.7, on présente l'algorithme de la procédure *Generate*. Cette procédure initialise l'exécution de l'algorithme de Saturation afin de calculer le MDD qui représente l'espace d'états atteignables du modèle.

À partir d'un état global initial $\tilde{x}$, pour chaque niveau l du MDD (où on considère les niveaux *de bas en haut*, *i.e.*, de 1 à N), un noeud est créé, initialisé en fonction de l'état local $x^{(l)}$ et saturé immédiatement.

À la fin de l'exécution de la procédure *Generate* (au niveau N), le noeud r se trouvera dans son format final (*saturé*) et il sera le noeud "racine" du MDD qui représente l'ensemble d'états atteignables du modèle à partir de l'état global initial $\tilde{x}$.

Ensuite, on note les variables les plus significatives utilisées dans la procédure *Generate* (ALG. 5.7).

✍ **Notation**

$\tilde{x}$	l'état initial du modèle SAN utilisé pour démarrer l'exécution de l'algorithme de Saturation ;
p	un noeud auxiliaire ;
r	le noeud qui est créé et initialisé à chaque niveau (automate) du modèle. Lorsque le noeud r se trouve saturé, il est le noeud *racine* du MDD qui servira à calculer l'ensemble d'états atteignables du modèle. Le calcule est terminé au niveau N.

ALG. 5.7 Generate(*in* $\tilde{x}$: *state*) : *node*

```
1:  /* p un noeud initialisé au noeud terminal 1 */
2:  p ← 1 ;
3:  /* Parcours de chaque état local de l'état global x̃ */
4:  for l = 1 to N do
5:      /* Création du noeud r au niveau l */
6:      r ← NewNode(l) ;
7:      /* Initialisation de r en fonction de x^(l), i.e., il y a un arc sortant de r vers p étiqueté par la valeur de x^(l) */
8:      r[x^(l)] ← p ;
9:      /* Saturation du noeud r */
10:     Saturate(r) ;
11:     /* Mise à jour du noeud p pour la saturation du noeud suivant */
12:     p ← r ;
13: end for
14: /* Retourne le noeud ("racine") r du MDD qui représente le RSS du modèle */
15: return r ;
```

5.5.4.2 La procédure *Locals*

La procédure *Locals* prend en paramètre : t (un terme tensoriel du descripteur d'atteignabilité du modèle) ; nf (un noeud du MDD $TFMdd_t$), et p (un noeud du MDD courant de l'espace d'états atteignables).

Dans ALG. 5.8, on présente l'algorithme de la procédure *Locals*. *Locals* examine si les valeurs (étiquettes) des arcs sortant du noeud p au niveau (automate) l sont d'états (locaux) de départ d'au moins une transition représentée dans la matrice $\breve{R}_t^{(l)}$ du terme tensoriel t.

Un terme tensoriel peut être classé comme *assimilé-constant* ou *fonctionnel* (voir Définition 5.3.1, page 76). Dans la section 5.5.3, on a démontré comment on peut déterminer si $\tilde{x}$ est un état de départ d'une transition représentée dans les matrices d'un terme tensoriel (assimilé-constant ou fonctionnel) t, analysant individuellement chaque état local de $\tilde{x} = (x^{(N)}, ..., x^{(1)})$.

Dans ALG. 5.8, pour savoir si les états représentés par p (*i.e.*, les étiquettes des arcs sortant du noeud p) au niveau l sont des états de départ d'au moins une transition représentée dans les matrices du terme t, on examine, pour tous les arcs sortant de p (ligne 7),

1. si $l \in \mathcal{O}^{(f_t)}$, alors l'état de l'automate $\mathcal{A}^{(l)}$ est argument d'une fonction dans les matrices de t. Dans ce cas, pour savoir si $x^{(l)}$ est un état de départ d'une transition, l'algorithme examine l'arc sortant du noeud nf étiqueté par $x^{(l)}$ (ligne 12) pour savoir si l'évaluation de la fonction conduira vers une valeur non nulle (donc on a une transition possible) ;
2. autrement, l'analyse des étiquettes des arcs sortant du noeud p est faite seulement par la fonction "*next-state*" locale $\mathcal{N}_{l,t}$ (ligne 17).

Ensuite, on donne des notations pour les variables les plus significatives utilisées dans la procédure *Locals* (ALG. 5.8).

✍ **Notation**

t	un terme tensoriel du descripteur d'atteignabilité ;
p	un noeud au niveau l du MDD courant de l'espace d'états atteignables ;
nf	un noeud au niveau l du MDD $TFMdd_t$;
Idx	l'ensemble des étiquettes des arcs sortant de p (*i.e.*, des états représentés par p) qui sont d'états de départ d'au moins une transition représentée dans la matrice d'atteignabilité partielle $\breve{R}_t^{(l)}$ du terme tensoriel t.

ALG. 5.8 Locals(*in* t : *tensor term*, *in* nf : *node*, *in* p : *node*) : *set of states*

```
1: /* Conserve dans l la valeur courant de p.lvl */
2: l ← p.lvl ;
3: /* Initialisation de Idx */
4: Idx ← ∅ ;
5: for all x^(l) ∈ S^(l) do
6:    /* L'état x^(l) est une étiquette d'un arc sortant de p */
7:    if (p[x^(l)] ≠ ∅) then
8:       /* Analyse si l'état de l'automate A^(l) est argument d'une fonction quelconque dans les matrices de t */
9:       if (l ∈ O(f_t)) then
10:         /* Si l ∈ O(f_t), alors t est un terme tensoriel fonctionnel */
11:         /* Donc, analyse si le noeud nf possède aussi un arc sortant étiqueté par la valeur x^(l) */
12:         if (nf[x^(l)] ≠ ∅) then
13:            /* x^(l) est un état de départ d'une transition représentée dans les matrices de t */
14:            Idx ← Idx ∪ {x^(l)} ;
15:      else
16:         /* Si l ∉ O(f_t), alors analyse si x^(l) est un état de départ d'une transition via la fonction "next-state" locale */
17:         if (N_{l,t}(x^(l)) ≠ ∅) then
18:            /* x^(l) est un état de départ d'une transition représentée dans les matrices de t */
19:            Idx ← Idx ∪ {x^(l)} ;
20: end for
21: /* Retourne l'ensemble d'états de p qui sont d'états de départ d'une transition représentée dans une matrice de t */
22: return Idx ;
```

5.5.4.3 La procédure *Saturate*

Dans ALG. 5.9, on présente l'algorithme de la procédure *Saturate*. La procédure *Saturate*(p) prend en paramètre un noeud p qui se situe au niveau l du MDD qui représente l'espace d'états atteignables courant et sature ce noeud immédiatement. Dans cette saturation, les valeurs des arcs du noeud p sont modifiées de façon à éviter la création de nouveaux noeuds au niveau l (ce qui avait lieu dans l'algorithme BFS - Section 5.4.2, page 81). On utilise l'expression *sur place* pour parler de ce traitement.

Définition 5.5.8. *Soit* $l \in [1..N]$ *l'indice d'un automate d'un modèle SAN, on définit l'ensemble de termes tensoriels* $\mathbb{T}^{(l)}$ *par* $\mathbb{T}^{(l)} = \{e \in \mathcal{E}, t_e \in \mathbb{T} : Top(t_e) = l\}$.

✍ **Notation**

t_e	un terme tensoriel du descripteur d'atteignabilité, où $t_e \in \mathbb{T}^{(l)}$, *i.e.*, l = *Top*(t_e) ;
nf	un noeud du MDD $TFMdd_{t_e}$;
p	le noeud du MDD courant de l'espace d'états atteignables qui est saturé. Le noeud p est une variable du type *in/out*, *i.e.*, une modification en p indique une modification à la variable qui a été passée comme paramètre à l'exécution de *Saturate*.

Sachant que l'algorithme de Saturation assure que tous les noeuds fils de p sont déjà saturés, la saturation de p implique de considérer toutes les transitions possibles représentées dans les matrices des termes tensoriels $t_e \in \mathbb{T}^{(l)}$, transitions qui ont comme état de départ un $\tilde{x}$ dont la l-ième composante $x^{(l)}$ est l'étiquette d'un arc sortant de ce noeud.

Soit un terme tensoriel $t_e \in \mathbb{T}^{(l)}$ du descripteur d'atteignabilité d'un modèle SAN, la saturation d'un noeud p au niveau l du MDD est faite en trois étapes :

ÉTAPE 1 : découvrir quels sont les sous-vecteurs d'états $\tilde{x}^{(O)} \in \mathcal{B}(p)$ de $\tilde{x}$ qui sont d'états de départ des transitions représentées dans les matrices du terme tensoriel t_e, où O est l'ensemble d'indices d'automates de *Top*(t_e) à *Bot*(t_e), *i.e.*, $O = \{Top(t_e), ..., Bot(t_e)\}$;

ÉTAPE 2 : à partir de ces états de départ $\tilde{x}^{(O)} \in \mathcal{B}(p)$, découvrir les états d'arrivée $\tilde{y}^{(O)}$ des transitions représentées dans les matrices de t_e, en utilisant la fonction "*next-state*" $\mathcal{N}_{t_e}$;

ÉTAPE 3 : ajouter les nouveaux états (atteignables) d'arrivée $\tilde{y}^{(O)}$ au MDD du RSS courant (*i.e.*, ajouter ces états aux noeuds de ce MDD).

Dans ALG. 5.9, ÉTAPE 1 (lignes 14 - 23) correspond à l'exécution de la procédure *Locals* (pour trouver la l-ième composante $x^{(l)}$ de $\tilde{x}^{(O)}$) après l'exécution de la procédure *Fire* (pour trouver les composantes de $\tilde{x}^{(O)}$ d'indices inférieurs à l). La procédure *Fire* sera expliquée ensuite dans la section 5.5.4.4 (page 95). Le noeud nf du $TFMdd_{t_e}$ est initialisé en fonction du niveau l analysé.

ÉTAPE 2, ligne 28 dans ALG. 5.9, correspond à trouver via la fonction "*next-state*" locale $\mathcal{N}_{l,t_e}$ la l-ième composante $y^{(l)}$ de $\tilde{y}^{(O)}$, l'état d'arrivée des transitions représentées dans t_e. Grâce à la propriété 5.2.1 (page 75), si $x^{(l)}$ de $\tilde{x}^{(O)}$ est un état de départ d'une transition de t_e, tout élément (assimilé-constant ou fonctionnel) de $\mathcal{N}_{l,t_e}(x^{(l)})$ est état d'arrivée $y^{(l)}$ de $\tilde{y}^{(O)}$ des transitions de t_e.

Alors, la dernière étape (ÉTAPE 3), lignes 29 - 32 dans ALG. 5.9, correspond à ajouter les états d'arrivée $y^{(l)}$ au noeud p du MDD courant de l'espace d'états atteignables du modèle SAN. Quand on trouve de nouveaux états d'arrivée $y^{(l)}$, *i.e.*, le résultat de l'union de $\mathcal{B}(q)$ et $\mathcal{B}(p[y^{(l)}])$ (stocké dans u - ligne 29) est différent de $\mathcal{B}(p[y^{(l)}]$ (ligne 30), il faut encore saturer le noeud p afin de que la condition $\mathcal{B}(p) = \mathcal{N}^*_{\leq l}(\mathcal{B}(p))$ soit valide. Il faut remarquer que les valeurs des arcs du noeud p sont modifiées *sur place*, évitant ainsi la création de nouveaux noeuds. En ajoutant directement les états d'arrivée $y^{(l)}$ au noeud p, tout l'ensemble d'états qui utilise ce noeud (*i.e.*, tous les chemins de $\mathcal{A}(p)$) tire avantage de cette modification.

ALG. 5.9 Saturate(*in/out* p : *node*)

```
1:  /* Conserve dans l la valeur courant de p.lvl */
2:  l ← p.lvl ;
3:  /* Saturation du noeud p jusqu'à ne plus ajouter des nouveaux états atteignables à ce noeud */
4:  repeat
5:     /* Initialisation de change pour garder en mémoire le fait que le noeud p est modifié */
6:     change ← false ;
7:     for each e ∈ ℰ do
8:        for each t_e ∈ 𝕋^(l) do
9:           /* Le noeud rf est le noeud racine du MDD TFMdd_{t_e} */
10:          rf ← RootNode(TFMdd_{t_e}) ;
11:          /* Prends les valeurs d'étiquettes (états) x^(l) des arcs sortant de p qui sont les états de départ des transitions
             représentées dans les matrices de t_e */
12:          for each x^(l) ∈ Locals(t_e, rf, p) do
13:             /* Initialisation du noeud nf du MDD TFMdd_{t_e} en fonction du noeud rf */
14:             if (t_e est un terme tensoriel fonctionnel) then
15:                if (rf.lvl = l) then
16:                   /* Le noeud nf est un noeud fils du noeud rf */
17:                   nf ← rf[x^(l)] ;
18:                else
19:                   /* Sinon, l > rf.lvl, alors le noeud nf = rf */
20:                   nf ← rf ;
21:             else
22:                /* t_e est assimilé-constant, alors nf = rf = 1 */
23:                nf ← rf ;
24:             /* Analyse si les états représentés par le noeud fils p[x^(l)] sont aussi états de départ des transitions représen-
                tées dans les matrices de t_e */
25:             q ← Fire(t_e, nf, p[x^(l)]) ;
26:             /* Si x^(l) est un état de départ d'une transition de t_e, alors tout y^(l) ∈ 𝒩_{l,t_e}(x^(l)) est un
27:             état d'arrivée (Propriété 5.2.1) */
28:             for each y^(l) ∈ 𝒩_{l,t_e}(x^(l)) do
29:                u ← Union(q, p[y^(l)]) ;
30:                if (u ≠ p[y^(l)]) then
31:                   p[y^(l)] ← u ;
32:                   change ← true ;
33:             end for
34:          end for
35:       end for
36:    end for
37: until (change = false)
```

Grâce à la représentation de la fonction "*next-state*" $\mathcal{N}_{t_e}$ (*i.e.*, la fonction "*next-state*" peut être représentée par le produit de N fonctions locales), l'exploration des états de départ et des états d'arrivée des transitions représentées dans les matrices du terme tensoriel t_e dépend seulement des états des automates d'indices de *Top*(t_e) à *Bot*(t_e) et ne dépend pas des automates de N à *Top*(t_e)+1.

Les procédures *Saturate* et *Fire* sont mutuellement récursives : la saturation du noeud p exécute la procédure *Fire* qui peut créer de nouveaux noeuds aux niveaux inférieurs à l, et ces noeuds doivent être saturés avant que la saturation de p ne continue. On présente maintenant la procédure *Fire*.

5.5.4.4 La procédure *Fire*

La procédure *Fire* prend en paramètre : t (un terme tensoriel du descripteur d'atteignabilité du modèle) ; nf (un noeud du MDD $TFMdd_t$), et p (un noeud du MDD courant du RSS).

On présente la procédure *Fire* dans ALG. 5.10. Ensuite, on donne les notations pour les variables les plus significatives utilisées dans ALG. 5.10.

✍ **Notation**

t	un terme tensoriel du descripteur d'atteignabilité ;
nf	un noeud du MDD $TFMdd_t$;
p	le noeud du MDD courant de l'espace d'états atteignables, dont les étiquettes (valeurs) d'arcs sont analysées ;
s	le noeud du MDD courant de l'espace d'états atteignables qui est le résultat de l'exécution de Fire(t, nf, p).
$nfils$	un noeud du MDD $TFMdd_t$.

La procédure *Fire* (ALG. 5.10) analyse à partir des arcs sortant du noeud p s'il y a une suite d'étiquettes (*i.e.*, les valeurs des arcs sortant des noeuds) qui représente un état de départ pour une transition représentée dans les matrices du terme t. Il est intéressant de remarquer que l'exécution de *Fire* est faite de façon récursive pour tous les noeuds qui se trouvent entre les niveaux Top(t) et Bot(t), *i.e.*, la *localité* du terme tensoriel t (voir Définition 5.5.3). La condition $l < Bot(t)$ est la condition de terminaison de l'algorithme récursif *Fire*, sachant que pour tout automate d'indice l, où $Bot(t) > l \geq 1$, la matrice d'atteignabilité $\breve{R}_t^{(l)}$ est une matrice identité.

Au contraire de la procédure *Saturate*, la procédure *Fire* utilise un nouveau noeud s au lieu de modifier directement les arcs du noeud p, sachant que (a) le noeud p se trouve déjà saturé et (b) il peut pointer vers des autres noeuds au niveau $l + 1$.

Si dans une exécution récursive de *Fire*, s'il n'y a pas d'étiquette $x^{(l)}$ d'arc sortant de p qui représente un état de départ d'une transition représentée dans les matrices de t (*i.e.*, $\nexists x^{(l)} \in$ Locals(t, nf, p)), alors le noeud $s = z_l$ (le noeud spécial[1] au niveau l qui représente l'ensemble vide).

[1] Plus de détails sur le *noeud spécial* z_l se trouve dans la section 5.6, page 97.

Il est intéressant de remarquer qu'avant de retourner le noeud s, la procédure *Fire* exécute *Saturate*(s) afin d'assurer que les noeuds du MDD aux niveaux inférieurs sont saturés.

ALG. 5.10 Fire(*in* t : *tensor term*, *in* nf : *node*, *in* p : *node*) : *node*

```
1: /* Conserve dans l la valeur courant de p.lvl */
2: l ← p.lvl ;     /* Au premier appel l = Top(t) */
3: /* Les états des automates d'indice l < Bot(t) (i.e., qui ne sont pas concernés par le terme t) ne sont pas modifiés par
   l'occurrence des transitions de t. C'est la condition de terminaison de l'algorithme récursif */
4: if (l < Bot(t)) then
5:    return p ;
6: s ← NewNode(l) ;    /* Création du noeud résultant s au niveau l du MDD */
7: /* Prends les valeurs des étiquettes (états) x^(l) des arcs sortant de p qui sont les états de départ des transitions représentées
   dans les matrices de t */
8: for each x^(l) ∈ Locals(t, nf, p) do
9:    /* Initialisation du noeud nfils du MDD TFMdd_t en fonction du noeud nf (paramètre) */
10:   if (t est un terme tensoriel fonctionnel) then
11:      if (nf.lvl = l) then
12:         /* Le noeud nfils est un noeud fils du noeud nf */
13:         nfils ← nf[x^(l)] ;
14:      else
15:         /* Sinon, l > nf.lvl, alors le noeud nfils = nf */
16:         nfils ← nf ;
17:   else
18:      /* t est assimilé-constant, alors nfils = nf = 1 */
19:      nfils ← nf ;
20:   /* Analyse si les états représentés par le noeud fils p[x^(l)] sont aussi états de départ des transitions représentées dans
      les matrices de t */
21:   q ← Fire(t, nfils, p[x^(l)]) ;
22:   /* Si x^(l) est un état de départ d'une transition de t, alors tout y^(l) ∈ N_{l,t}(x^(l)) est un
23:   état d'arrivée (Propriété 5.2.1) */
24:   for each y^(l) ∈ N_{l,t}(x^(l)) do
25:      /* On construit les sous-arbres des états atteignables du noeud s au niveau l */
26:      s[y^(l)] ← Union(q, s[y^(l)]) ;
27:   end for
28: end for
29: /* Saturation du noeud résultant s */
30: Saturate(s) ;
31: return s ;
```

5.5.5 Bilan de la génération basée sur la saturation

La génération basée sur la saturation de l'espace d'états atteignables de modèles qui utilisent des fonctions repose sur la représentation d'états atteignables par des MDD et de la fonction "*next-state*" par des produits de Kronecker. Les algorithmes présentés précédemment pour la génération basée sur la saturation dans ce chapitre se sont inspirés des travaux réalisés pour le formalisme de *Réseaux de Petri Stochastiques* (SPN - *Stochastic Petri Nets*) [32, 30].

Toutefois, ces travaux pour le formalisme SPN ne présentent pas de solution pour les modèles qui utilisent des fonctions lorsque les arguments de ces fonctions ne sont pas limités à l'espace d'états local d'un sous-système. La solution pour l'utilisation de fonctions sans hypothèse sur les arguments est mise en évidence dans les procédures :

– *Locals* (ALG. 5.8, page 92) : lignes 9 - 14 ;

– *Saturate* (ALG. 5.9, page 94) : lignes 14 - 23 ;

– *Fire* (ALG. 5.10, page 96) : lignes 10 - 19.

Ces modifications dans les algorithmes sont possibles grâce à :

1. *Le descripteur d'atteignabilité*. Il est représenté par un *format tensoriel* qui utilise les matrices d'atteignabilité partielles du modèle. Les termes tensoriels du descripteur sont fabriqués de façon à ce que les fonctions d'une même matrice aient le même ensemble d'états où elles peuvent devenir identiquement nulles. Ceci permet d'avoir une fonction "*next-state*" structurée selon l'espace produit ;

2. *Des MDD associés à des taux et des probabilités fonctionnels*. La représentation par un MDD de l'ensemble d'états pour lesquels l'*évaluation d'une fonction* est différente de zéro (ALG. 5.1, page 74) ;

3. *Des MDD associés à des fonctions d'un terme tensoriel*. La représentation par un MDD de l'ensemble d'états pour lesquels l'*évaluation de toutes les fonctions* des éléments fonctionnels qui apparaissent dans les matrices d'atteignabilité d'un terme tensoriel est différente de zéro (ALG. 5.3, page 78).

5.6 Implantation

Dans cette section, on va introduire les types et les structures de données utilisées pour la manipulation des noeuds d'un MDD.

Au contraire de [82] où les MDD sont représentés par des BDD, il est démontré dans [94] que l'utilisation directe des MDD peut fournir des gains de performance pour la génération de l'espace d'états atteignables de modèles. La manipulation d'un état local dans un MDD est faite seulement dans un noeud au lieu de plusieurs noeuds du BDD qui représenteraient le noeud du MDD.

Pour l'implantation, les types de donnés sont simplement des *integers* dans un intervalle adéquat. Comme pour les algorithmes de génération symbolique traditionnels de l'espace d'états, on utilise un *tableau unique* pour la détection des noeuds *doubles* (Définition 5.1.2, page 71) et des *caches* (*cache d'union* et *cache de tirage*) pour éviter le recalcul d'opérations déjà effectuées pour les noeuds du MDD.

Les noeuds sont divisés en N groupes, un pour chaque niveau (automate). A fin d'éviter les noeuds *doubles*, chaque groupe de noeuds est géré par une *table de hachage* (*tableau unique*) de dimension variable qui supporte les insertions et suppressions de noeuds pendant l'exécution, et un accès en *temps constant* pour chaque élément.

✍ **Soit**

UT le *tableau unique* (*table de hachage*) pour les noeuds d'un MDD, où $UT[l]$ est le *tableau unique* pour les noeuds du niveau l du MDD (pour $N \geq l \geq 1$). Les valeurs des arcs d'un noeud sont utilisées comme *clé* pour ce tableau unique, *i.e.*, soit un noeud p au niveau l, on utilise les valeurs des arcs $p[0], \ldots, p[n_l - 1]$ pour trouver le noeud p dans $UT[l]$;

Dans FIG. 5.7, on présente l'implantation des noeuds au niveau l quelconque d'un MDD utilisant des vecteurs extensibles de noeuds et d'arcs.

Le vecteur de noeuds (*node*) est organisé selon les indices des noeuds du MDD, *i.e.*, la partie constante des données du noeud p au niveau l est stockée dans $node[l][p]$. L'indice du noeud p est *unique* dans le niveau l. Un noeud quelconque doit être référencé par une paire (l, p).

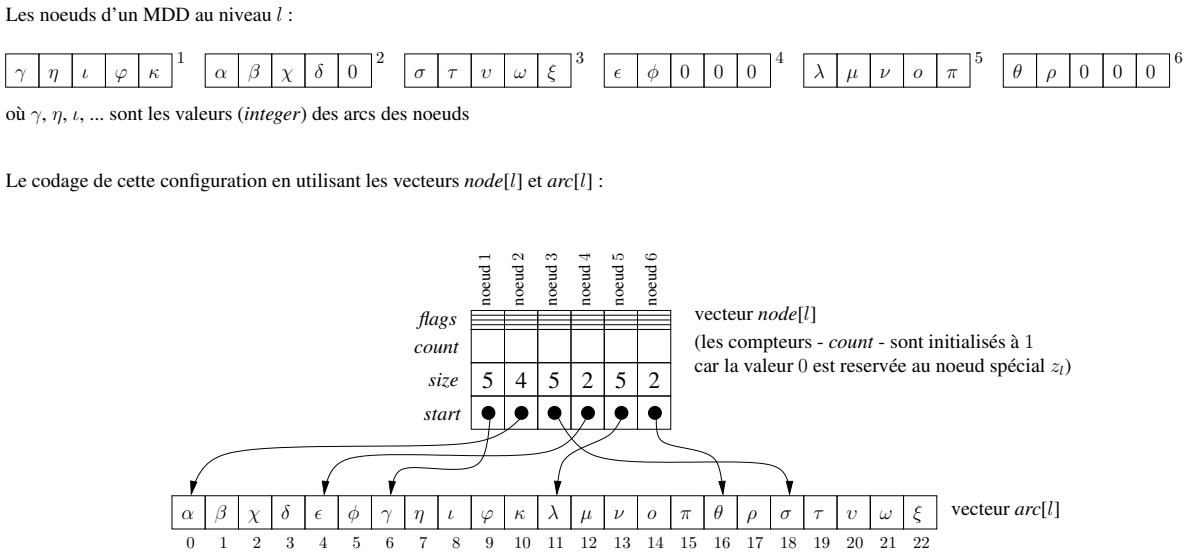


FIG. 5.7 – L'implantation des noeuds d'un MDD utilisant des vecteurs extensibles [30]

On réserve un noeud spécial z_l au niveau l d'un MDD pour indiquer l'ensemble qui va vers le noeud terminal $\mathbf{0}$: $\mathcal{B}(z_l) = \emptyset$. Tous les arcs du noeud z_l pointent vers le noeud z_{l-1}, et z_0 représente le noeud terminal $\mathbf{0}$.

Le vecteur d'arcs est indexé par le champ *start* et *size* du vecteur de noeuds, *i.e.*, si $node[l][p].start = a$ et $node[l][p].size = b$, alors $p[i]$ est stocké dans $arcs[l][a + i]$ pour $0 \leq i < b$. Les arcs sont stockés dans un *format plein tronqué*, *i.e.*, $b < n_l$ indique que $p[b] = \cdots = p[n_l - 1] = z_{l-1}$, où n_l est la taille du noeud au niveau l.

Chaque noeud ayant des caractéristiques différentes par rapport à l'utilisation de la mémoire, des *flags* sont utilisées pour les différencier. Il est intéressant de remarquer que l'indice 0 (*zéro*) du vecteur de noeuds (*node*) de chaque niveau est réservé au noeud spécial z_l.

Soit les vecteurs *node*[l] et *arc*[l] dans FIG. 5.7, on exécute :

$$r \leftarrow \mathit{NewNode}(l)$$

pour créer un nouveau noeud r au niveau l. Dans ce cas, on ajoute un élément au vecteur *node*[l] (*i.e.*, il passe à 7 éléments) et le noeud r est initialisé :

	0	1	2	3	4	
noeud r	0	0	0	0	0	7

Soit le noeud r créé précédemment au niveau l dans FIG. 5.7 et un noeud p au niveau $l-1$, l'exécution :

$$r[0] \leftarrow p$$

change les vecteurs *node*[l] et *arc*[l] de la façon suivante :

node[l][7].*start* = 23	où 23 est la position initiale dans le vecteur *arc*[l] pour le noeud r (*i.e.*, le noeud 7)
node[l][7].*size* = 1	
arc[l][23] = δ	où δ est la clé du noeud p pour le tableau unique UT au niveau $l-1$

et le noeud r est modifié :

	0	1	2	3	4	
noeud r	δ	0	0	0	0	7

✍ **Soit**

UC	le *cache d'union* pour les noeuds d'un MDD, où $UC[l]$ est le *cache* de l'opération union entre les noeuds du niveau l du MDD (pour $N > l \geq 1$). Soit deux noeuds p et q au niveau l, on utilise l'ensemble non-ordonné $\{p, q\}$ comme *clé* pour ce cache afin de récupérer un noeud s qui est le résultat de l'opération union entre les noeuds p et q, *i.e.*, $\mathcal{B}(s) = \mathcal{B}(p) \cup \mathcal{B}(q)$.

Le *cache d'union* (UC) est utilisé pour éviter de recalculer l'opération d'union entre deux noeuds d'un MDD. Ce cache sert dans la procédure *Union* utilisée dans les algorithmes de ce chapitre (*e.g.*, ligne 29 dans ALG. 5.9, page 94). On a deux procédures pour gérer le cache UC :

– *Insert*($UC[l]$, $\{p, q\}$, r) : le noeud r est le noeud résultant de l'opération union entre les noeuds p et q. La procédure *Insert* stocke dans le cache d'union au niveau l ($UC[l]$) le noeud r en utilisant l'ensemble non-ordonné $\{p, q\}$ comme clé ;

– *Find*($UC[l]$, $\{p, q\}$, r) : en utilisant l'ensemble non-ordonné $\{p, q\}$, la procédure *Find* rend dans r le résultat de l'opération d'union entre les noeuds p et q du niveau l.

✍ **Soit**

FC	le *cache de tirage* pour les noeuds d'un MDD, où $FC[l]$ est le *cache* de tirage d'événements pour les noeuds du niveau l du MDD (pour $N > l \geq 1$). Soit un noeud p au niveau l et un terme tensoriel $t_e \in \mathbb{T}$ associé à un événement $e \in \mathcal{E}$. On utilise l'ensemble $\{p, t_e\}$ comme *clé* pour ce cache afin de trouver un noeud s tel que $\mathcal{B}(s) = \mathcal{N}^*_{\leq l}(\mathcal{N}_{t_e}(\mathcal{B}(p)))$, où $Top(t_e) > l \geq Bot(t_e)$.

Le *cache de tirage* (FC) est utilisé pour éviter de réexaminer les transitions représentées dans les matrices d'un terme t pour un même noeud p. Le cache FC est utilisé dans les procédures *Fire* (ALG. 5.6, page 83 et ALG. 5.10, page 96). On a aussi deux procédures pour gérer le cache FC :

– *Insert*($FC[l]$, $\{p, t\}$, r) : le noeud r est le noeud résultant de l'exécution *Fire*(t, p). La procédure *Insert* stocke dans le cache de tirage au niveau l ($FC[l]$) le noeud r en utilisant l'ensemble non-ordonné $\{p, t\}$ comme clé pour ce cache ;

– *Find*($FC[l]$, $\{p, t\}$, r) : en utilisant l'ensemble non-ordonné $\{p, t\}$, la procédure *Find* rendre dans r le résultat de l'exécution *Fire*(t, p) stocké dans ce cache.

Il est intéressant de remarquer que $FC[l]$ ne possède pas d'éléments pour $t \in \mathbb{T}^{(l)}$, vu que l'algorithme sature un noeud en modifiant directement les valeurs des arcs de ce noeud ; pour la même raison, on n'utilise pas les caches en UC et FC au niveau N.

Définition 5.6.1. *Soit un noeud non-terminal p au niveau l d'un MDD, p est un noeud* ***redondant*** *si $p[i] = p[0]$ pour tout $0 \leq i < n_l$.*

Dans cette thèse, on utilise des MDD *quasi-réduits*. Un MDD quasi-réduit est un MDD qui permet l'utilisation des noeuds *redondants*. En effet, il y a des gains importants grâce à l'implantation des MDD (*quasi-réduits*) où les arcs des noeuds pointent d'un niveau l vers un niveau $l-1$ (*i.e.*, la différence entre les niveaux d'un noeud p qui pointe vers un noeud q est égale à 1) :

- les opérations (*e.g.*, l'union entre deux MDD) utilisées dans l'algorithme de Saturation (Section 5.5.4, page 90) sont exécutées sur les noeuds d'un même niveau. D'une façon analogue aux BDD, chaque opération des MDD utilise un *cache* pour réduire la complexité de calcul. Vu que les *caches* sont organisés par niveaux, on n'a pas besoin stocker des informations de niveau des noeuds ;

- les opérations des MDD sont effectuées directement sur les noeuds aux niveaux qui concernent le tirage d'un événement au lieu d'appliquer les opérations sur les états codés par ces noeuds ;

- le champ *count* du vecteur *node*$[l][p]$ représente le nombre d'arcs qui pointent vers le noeud p. Lorsqu'un arc qui pointe vers p est redirigé vers un autre noeud, on diminue la valeur de *node*$[l][p]$.*count* (*i.e.*, *node*$[l][p]$.*count* = *node*$[l][p]$.*count* $-$ 1) et s'il est égal à 0, alors le noeud p est *déconnecté* du MDD et il peut être supprimer du *tableau unique* ultérieurement.

5.7 Conclusion

L'algorithme de Saturation présenté dans ce chapitre est une méthode symbolique basée sur des diagrammes de décision qui peut être utilisée de façon performante pour la génération de l'espace d'états atteignables de modèles SAN. L'idée de base pour obtenir cette performance est d'explorer systématiquement l'effet local des tirages d'événements.

L'algorithme de Saturation utilise des *Diagrammes de Décision Multi-valués* (MDD) qui représentent naturellement la structure des vecteurs d'états définis pour des modèles structurés (modulaires). De plus, il permet l'utilisation d'expressions de Kronecker pour la représentation des fonctions "*next-state*", stockant ces expression via un ensemble de matrices. Ainsi, l'algorithme utilise des opérations de *poids faible* dans la manipulation des MDD, en améliorant l'efficacité par rapport au temps de génération des méthodes traditionnelles (qui ont une seule itération et une fonction "*next-state*" de *poids fort*).

La génération basée sur la saturation est une nouvelle démarche pour la génération symbolique de l'espace d'états atteignables de modèles SAN. L'algorithme de Saturation présenté dans ce chapitre, outre qu'il permet l'utilisation des fonctions sans hypothèse sur les arguments, permet aussi la représentation d'un très grand espace d'états atteignables d'un modèle SAN par un MDD (par exemple, des modèles avec des milliard d'états), alors que la représentation *vectorielle* courante de cet espace d'états atteignables est inefficace, ou même infaisable. Toutefois, la génération des MDD $FuncMdd$ de fonctions sans hypothèse sur les arguments (ALG. 5.1, page 74) peut avoir un coût de calcul très élevé. Le coût de calcul pour générer un MDD $FuncMdd_{id}$ d'une fonction f_{id} est :

$$\sum_{\tilde{x}^{(\omega_{id})} \in \hat{\mathcal{S}}^{(\omega_{id})}} E(\tilde{x}^{(\omega_{id})}) \quad + \sum_{\tilde{x}^{(\omega_{id})} \in \hat{\mathcal{S}}^{(\omega_{id})} / f_{id}(\tilde{x}^{(\omega_{id})}) \neq 0} M(\tilde{x}^{(\omega_{id})}) + U(\tilde{x}^{(\omega_{id})})$$

où $E(\tilde{x}^{(\omega_{id})})$ est le coût de calcul de l'évaluation de la fonction pour l'état $\tilde{x}^{(\omega_{id})}$, et pour tout état $\tilde{x}^{(\omega_{id})}$ pour lequel l'évaluation de la fonction est différente de zéro, $M(\tilde{x}^{(\omega_{id})})$ est le coût de calcul de la génération du MDD qui représente l'état $\tilde{x}^{(\omega_{id})}$ et $U(\tilde{x}^{(\omega_{id})})$ est le coût de calcul de l'opération union entre le MDD $FuncMdd_{id}$ et le MDD qui représente l'état $\tilde{x}^{(\omega_{id})}$.

La décomposition proposée dans le chapitre 4, des matrices d'atteignabilité selon les classes d'équivalence, et la représentation de l'ensemble d'états pour lesquels l'évaluation d'une fonction est différente de zéro par un MDD ont permis le développement d'algorithmes de génération du RSS de modèles qui utilisent fonctions (notamment, pour les modèles SAN). L'utilisation de ces algorithmes n'est pas limitée au formalisme SAN, ces algorithmes peuvent être utilisés pour d'autres formalismes structurés de haut-niveau qui possèdent une représentation généralisée de Kronecker.

En bref, la génération basée sur la saturation amène les noeuds d'un MDD vers le format final (saturé) le plus tôt possible, considérant les niveaux des noeuds *de bas en haut*. Le tirage d'événements profite de la structure modulaire du modèle SAN, ainsi que la représentation de Kronecker des fonctions "*next-state*" locales. La modification des valeurs des arcs des noeuds de façon directe favorise le plus fréquemment possible la réutilisation des noeuds du MDD, au lieu de la génération de plusieurs noeuds qui seront normalement *déconnectés* du MDD (de la même façon que les méthodes classiques basées sur des BDD [90]). En utilisant ces caractéristiques, on peut calculer l'espace d'états atteignables de modèles d'une façon très performante, alors que le pic de mémoire reste relativement bas.

Les résultats expérimentaux pour la génération basée sur la saturation du RSS de modèles SAN est en moyenne de plusieurs ordres de grandeur plus performant en temps et utilisation de mémoire que la génération symbolique traditionnelle.

Dans le chapitre 6, on va présenter les résultats obtenus pour la génération (symbolique traditionnelle et basée sur la saturation) de l'espace d'états atteignables des modèles SAN présentés dans le chapitre 3.

Chapitre 6

Etudes d'exemples à temps continu

Dans ce chapitre, on présente quelques mesures de performance obtenues avec les méthodes et algorithmes que nous avons présentés dans le chapitre 5 pour la génération de l'espace d'états atteignables des modèles qui utilisent des fonctions (notamment, pour les modèles SAN). Ces algorithmes ont été développés dans un module du logiciel PEPS (*Performance Evaluation of Parallel Systems*) [17]. Les mesures de performance de ce chapitre ont été obtenues utilisant le module du logiciel PEPS sur un ordinateur AMD Opteron (avec 8 processeurs double coeur) à 2.2 GHz, sous un système Linux Debian 2.6.23-1-amd64, et avec 32 Giga octets de mémoire vive. Les mesures présentées dans ce chapitre ont été obtenues en utilisant seulement 1 processeur.

Les modèles SAN utilisés dans ce chapitre sont les mêmes que ceux décrits dans le chapitre 3 :

– Dîner des Philosophes (Section 6.1) ;
– Patron de Service Alterné (Section 6.2) ;
– Atelier avec kanbans (Section 6.3) ;
– Partage de Ressources (Section 6.4).

Pour chacun de ces exemples (avec taux constants et fonctionnels), utilisant les algorithmes BFS et de Saturation, on présente les mesures de performance dans un tableau avec les colonnes suivantes :

- (*Noeuds MDD*) **Final** : le nombre final de noeuds du MDD qui représente l'espace d'états atteignables (RSS - *Reachable State Space*) ;
- (*Noeuds MDD*) **Pic** : le pic du nombre de noeuds utilisés pendant la génération du RSS ;
- (*Mémoire*) **Final** : la quantité finale de mémoire utilisée pour stocker le MDD qui représente le RSS ;
- (*Mémoire*) **Pic** : le pic de mémoire utilisé pendant la génération de l'espace d'états atteignables ;
- (*Espace d'états*) **RSS** : le nombre d'états atteignables ;
- (*Espace d'états*) **PSS** : le nombre d'états de l'espace produit (PSS - *Product State Space*) ;

- (*Temps*) **Fonctions**[1] : le temps nécessaire à la génération de tous MDD $FuncMdd$;
- (*Temps*) **RSS** : le temps nécessaire à la génération de l'espace d'états atteignables.

6.1 Dîner des Philosophes

Dans cette section, on présente les mesures de performance obtenues utilisant les deux algorithmes BFS et de Saturation pour la génération du RSS des modèles SAN *taux constants* (FIG. 3.2) et *taux fonctionnels* (FIG. 3.3). N est le nombre de philosophes de l'exemple décrit dans la section 3.1 (page 40).

On présente dans TAB. 6.1 les mesures obtenues pour les modèles SAN (taux constants) utilisant l'algorithme BFS, et dans TAB. 6.2 les mesures obtenues pour les modèles SAN (taux fonctionnels) utilisant aussi l'algorithme BFS.

	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)
N	Final	Pic	Final	Pic	RSS	PSS	RSS
10	35	841	7,312	257,119	5.74×10^{3}	5.90×10^{4}	1.19×10^{-2}
50	195	29,761	36,752	7,940,999	1.17×10^{19}	7.18×10^{23}	2.35×10^{0}
100	395	124,411	73,552	32,683,849	1.62×10^{38}	5.15×10^{47}	6.50×10^{0}
500	1,995	3,221,611	367,952	836,146,905	2.08×10^{191}	3.64×10^{238}	1.60×10^{2}
1,000	3,995	12,943,111	735,952	3,354,275,629	5.09×10^{382}	1.32×10^{477}	9.10×10^{2}

TAB. 6.1 – Dîner des Philosophes (taux constants) - Algorithme BFS

	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
N	Final	Pic	Final	Pic	RSS	PSS	Fonctions	RSS
10	35	713	7,312	218,684	5.74×10^{3}	5.90×10^{4}	1.60×10^{-4}	9.07×10^{-3}
50	195	23,273	36,752	6,365,324	1.17×10^{19}	7.18×10^{23}	7.87×10^{-4}	2.27×10^{0}
100	395	96,473	73,552	26,018,264	1.62×10^{38}	5.15×10^{47}	1.76×10^{-3}	5.44×10^{0}
500	1,995	2,482,073	367,952	662,045,088	2.08×10^{191}	3.64×10^{238}	1.55×10^{-2}	1.26×10^{2}
1,000	3,995	9,964,073	735,952	2,654,078,472	5.09×10^{382}	1.32×10^{477}	4.95×10^{-2}	8.19×10^{2}

TAB. 6.2 – Dîner des Philosophes (taux fonctionnels) - Algorithme BFS

En observant les deux tableaux (TAB. 6.1 et TAB. 6.2), on remarque que le temps pour la génération du RSS pour les modèles SAN (taux fonctionnels) est un peu plus petit que celui des modèles SAN (taux constants). La durée de la génération des MDD $FuncMdd$ est très petit, sachant que les fonctions utilisées dans ce modèle sont évaluées sur un espace d'états très petit. Une fonction f quelconque dans ce modèle possède les états de deux automates seulement comme arguments (*i.e.*, $|\omega_f| = 2$). Vu que l'espace d'états de tous les automates est égal à 3 (trois), la génération d'un MDD $FuncMdd$ quelconque est faite évaluant seulement états du domaine de la fonction, *i.e.*, $3 \times 3 = 9$ états.

Dans les tableaux suivants (TAB. 6.3 et TAB. 6.4), on présente les mesures de performance obtenues utilisant l'algorithme de Saturation pour les modèles SAN (taux constants) et modèles SAN (taux fonctionnels) respectivement.

[1] Les MDD $FuncMdd$ ne sont que utilisés dans les modèles SAN avec taux fonctionnels.

N	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)
	Final	Pic	Final	Pic	RSS	PSS	RSS
10	35	56	7,312	34,448	5.74×10^{3}	5.90×10^{4}	1.54×10^{-3}
50	195	296	36,752	417,251	1.17×10^{19}	7.18×10^{23}	8.96×10^{-3}
100	395	596	73,552	1,445,101	1.62×10^{38}	5.15×10^{47}	1.84×10^{-2}
500	1,995	2,996	367,952	31,627,901	2.08×10^{191}	3.64×10^{238}	1.08×10^{-1}
1,000	3,995	5,996	735,952	124,256,401	5.09×10^{382}	1.32×10^{477}	2.44×10^{-1}

TAB. 6.3 – Dîner des Philosophes (taux constants) - Algorithme de Saturation

N	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
	Final	Pic	Final	Pic	RSS	PSS	Fonctions	RSS
10	35	83	7,312	34,171	5.74×10^{3}	5.90×10^{4}	1.70×10^{-4}	1.25×10^{-3}
50	195	443	36,752	361,048	1.17×10^{19}	7.18×10^{23}	8.12×10^{-4}	7.31×10^{-3}
100	395	893	73,552	1,183,248	1.62×10^{38}	5.15×10^{47}	1.86×10^{-3}	1.51×10^{-2}
500	1,995	4,493	367,952	24,320,848	2.08×10^{191}	3.64×10^{238}	1.60×10^{-2}	8.98×10^{-2}
1,000	3,995	8,993	735,952	94,642,848	5.09×10^{382}	1.32×10^{477}	5.08×10^{-2}	1.81×10^{-1}

TAB. 6.4 – Dîner des Philosophes (taux fonctionnels) - Algorithme de Saturation

Comme on peut l'observer dans TAB. 6.3 et TAB. 6.4, la génération du RSS pour les modèles SAN (taux fonctionnels) est encore un peu plus rapide que celle des modèles SAN (taux constants). On remarque aussi que la mémoire utilisée pour les modèles SAN (taux fonctionnels) est inférieure à la mémoire utilisée pour les modèles SAN (taux constants).

En regardant les quatre tableaux de mesures (TAB. 6.1, TAB. 6.2, TAB. 6.3 et TAB. 6.4), on constate que les mesures obtenues pour les modèles SAN (taux fonctionnels) utilisant l'algorithme de Saturation sont très performants, vu qu'on génère un très grand espace d'états (*e.g.*, pour $N = 1,000$, le RSS est d'ordre de 10^{382} états) en moins d'une seconde.

6.2 Patron de Service Alterné

On présente dans cette section les mesures obtenues utilisant les algorithmes BFS et de Saturation pour la génération du RSS des modèles SAN *taux constants* (FIG. 3.5) et *taux fonctionnels* (FIG. 3.6). P est le nombre des différents patrons de service et K_1, K_2, K_3 et K_4 sont les capacités (finies) des files d'attente Q_1, Q_2, Q_3 et Q_4 respectivement de l'exemple décrit dans la section 3.2 (page 43).

Les mesures obtenues pour les modèles SAN (taux constants) utilisant l'algorithme BFS pour cet exemple sont présentés dans TAB. 6.5, ainsi que les mesures pour les modèles SAN (taux fonctionnels) utilisant aussi l'algorithme BFS sont présentés dans TAB. 6.6.

On remarque que tout l'espace d'états des modèles SAN de cet exemple est atteignable (*i.e.*, RSS = PSS). L'espace d'états atteignables du modèle est représenté par un nombre très petit de noeuds (dans ce cas, 5 noeuds). En comparant les mesures de performance de TAB. 6.5 et TAB. 6.6, on constate que les temps de la génération du RSS pour les modèles avec taux fonctionnels sont plus rapides que ceux des

modèles avec taux constants. Dans ce cas, le modèle SAN (taux fonctionnels) utilise seulement 1 événement synchronisant (e_{34}) au lieu de P événements synchronisants ($e_{34(1)}$, $e_{34(2)}$, ..., $e_{34(P)}$) du modèle SAN (taux constants). Par conséquence, le nombre de termes tensoriels du descripteur d'atteignabilité du modèle SAN (taux fonctionnels) est très inférieur au nombre de termes tensoriels du descripteur d'atteignabilité du modèle SAN (taux constants), ce qui diminue significativement la quantité de calcul.

P	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)
	Final	Pic	Final	Pic	RSS	PSS	RSS
			$\mathbf{K_1 = 300}$	$\mathbf{K_2 = 300}$	$\mathbf{K_3 = 600}$	$\mathbf{K_4 = 600}$	
2	5	21,050	9,352	25,082,174	6.55×10^{10}	6.55×10^{10}	6.12×10^{0}
3	5	21,653	9,356	25,406,745	9.82×10^{10}	9.82×10^{10}	6.66×10^{0}
4	5	22,256	9,360	25,643,920	1.31×10^{11}	1.31×10^{11}	7.39×10^{0}
5	5	22,859	9,364	25,914,851	1.64×10^{11}	1.64×10^{11}	8.24×10^{0}
15	5	28,889	9,404	28,444,965	4.91×10^{11}	4.91×10^{11}	1.39×10^{1}
25	5	34,919	9,444	31,263,383	8.18×10^{11}	8.18×10^{11}	2.10×10^{1}

TAB. 6.5 – Patron de Service Alterné (taux constants) - Algorithme BFS

P	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
	Final	Pic	Final	Pic	RSS	PSS	Fonctions	RSS
			$\mathbf{K_1 = 300}$	$\mathbf{K_2 = 300}$	$\mathbf{K_3 = 600}$	$\mathbf{K_4 = 600}$		
2	5	19,852	9,352	19,272,713	6.55×10^{10}	6.55×10^{10}	1.21×10^{-5}	3.86×10^{0}
3	5	19,852	9,356	19,296,793	9.82×10^{10}	9.82×10^{10}	1.21×10^{-5}	3.87×10^{0}
4	5	19,852	9,360	19,320,881	1.31×10^{11}	1.31×10^{11}	1.21×10^{-5}	3.90×10^{0}
5	5	19,852	9,364	19,344,977	1.64×10^{11}	1.64×10^{11}	1.29×10^{-5}	3.99×10^{0}
15	5	19,852	9,404	19,586,377	4.91×10^{11}	4.91×10^{11}	1.91×10^{-5}	4.18×10^{0}
25	5	19,852	9,444	19,828,577	8.18×10^{11}	8.18×10^{11}	2.41×10^{-5}	4.34×10^{0}

TAB. 6.6 – Patron de Service Alterné (taux fonctionnels) - Algorithme BFS

Comme on peut l'observer dans TAB. 6.6, le temps pour générer le MDD $FuncMdd_{f_{34}}$ de la fonction f_{34} utilisée dans le modèle SAN (taux fonctionnels) est très petit (*i.e.*, d'ordre de 10^{-5}). Le domaine de la fonction f_{34} tient seulement compte de l'espace d'états de l'automate $\mathcal{A}^{(5)}$ qui est dépendant du nombre de patrons de services (P) du modèle. Dans ce contexte, l'évaluation de f_{34} est faite pour les P états de l'automate $\mathcal{A}^{(5)}$.

Dans les tableaux suivants (TAB. 6.7 et TAB. 6.8), on présente les mesures de performance obtenues en utilisant l'algorithme de Saturation pour les modèles SAN (taux constants) et modèles SAN (taux fonctionnels) respectivement.

Les temps pour la génération du RSS des modèles SAN (taux constants et fonctionnels) utilisant l'algorithme de Saturation sont très proches. Les modèles avec taux fonctionnels sont légèrement plus rapide que les modèles avec taux constants.

En regardant les mesures présentées dans cette section, on peut observer que les mesures obtenues pour les modèles SAN (taux fonctionnels) utilisant l'algorithme de Saturation sont les plus performants : on génère un grand espace d'états atteignables (*e.g.*, 10^{11} états) d'un modèle classique de réseaux de files d'attente très rapidement (proche de 1 seconde).

P	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)
	Final	Pic	Final	Pic	RSS	PSS	RSS
			$\mathbf{K_1 = 300}$	$\mathbf{K_2 = 300}$	$\mathbf{K_3 = 600}$	$\mathbf{K_4 = 600}$	
2	5	607	9,352	935,826	6.55×10^{10}	6.55×10^{10}	8.44×10^{-1}
3	5	607	9,356	945,955	9.82×10^{10}	9.82×10^{10}	8.49×10^{-1}
4	5	607	9,360	956,100	1.31×10^{11}	1.31×10^{11}	8.55×10^{-1}
5	5	607	9,364	966,261	1.64×10^{11}	1.64×10^{11}	8.77×10^{-1}
15	5	607	9,404	1,068,751	4.91×10^{11}	4.91×10^{11}	9.22×10^{-1}
25	5	607	9,444	1,172,841	8.18×10^{11}	8.18×10^{11}	1.02×10^{0}

TAB. 6.7 – Patron de Service Alterné (taux constants) - Algorithme de Saturation

P	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
	Final	Pic	Final	Pic	RSS	PSS	Fonctions	RSS
			$\mathbf{K_1 = 300}$	$\mathbf{K_2 = 300}$	$\mathbf{K_3 = 600}$	$\mathbf{K_4 = 600}$		
2	5	607	9,352	925,793	6.55×10^{10}	6.55×10^{10}	1.19×10^{-5}	8.41×10^{-1}
3	5	607	9,356	925,851	9.82×10^{10}	9.82×10^{10}	1.29×10^{-5}	8.45×10^{-1}
4	5	607	9,360	925,917	1.31×10^{11}	1.31×10^{11}	1.31×10^{-5}	8.50×10^{-1}
5	5	607	9,364	925,991	1.64×10^{11}	1.64×10^{11}	1.38×10^{-5}	8.71×10^{-1}
15	5	607	9,404	927,171	4.91×10^{11}	4.91×10^{11}	1.91×10^{-5}	8.85×10^{-1}
25	5	607	9,444	929,151	8.18×10^{11}	8.18×10^{11}	2.29×10^{-5}	9.02×10^{-1}

TAB. 6.8 – Patron de Service Alterné (taux fonctionnels) - Algorithme de Saturation

6.3 Atelier avec Kanbans

Dans cette section, on présente les mesures de performance obtenues utilisant les algorithmes BFS et de Saturation pour la génération du RSS des modèles SAN (taux constants) - FIG. 3.7, des modèles SAN (partiellement fonctionnel) - FIG. 3.8, et des modèles SAN (taux fonctionnels) - FIG. 3.9. N est le nombre de *kanbans* (*capacité*) dans les postes de travail du système décrit dans la section 3.3 (page 45).

Dans TAB. 6.9, TAB. 6.10 et TAB. 6.11, en utilisant l'algorithme BFS, on présente les mesures obtenues pour les modèles SAN (taux constants), modèles SAN (partiellement fonctionnel) et modèles SAN (taux fonctionnels) respectivement.

Dans les modèles SAN (partiellement fonctionnel), les automates P_1 et P_4 ne sont pas représentés, mais remplacés par deux fonctions sont utilisées (f_{in} et $f_{s_{234}}$), où le domaine de chaque fonction est de $(N+1)^3$ états. En comparant les mesures de performance de TAB. 6.9 et TAB. 6.10, on constate que les temps de la génération du RSS pour les modèles SAN (partiellement fonctionnel) sont un peu plus rapides que les modèles SAN (taux constants), même si on considère le temps pour la génération des MDD $FuncMdd_{f_{in}}$ et $FuncMdd_{f_{s_{234}}}$.

Dans les modèles SAN (taux fonctionnels), outre les fonctions f_{in} et $f_{s_{234}}$, on utilise la fonction $f_{s_{123}}$ au lieu des automates P_2 et P_3. Le domaine de cette fonction est de $(N+1)^6$ états et par conséquence la génération du MDD $FuncMdd_{f_{s_{123}}}$ est très lente. D'ailleurs, cela nécessite beaucoup de mémoire pour générer explicitement le $FuncMdd_{f_{s_{123}}}$, limitant la génération du RSS de ce modèle à des $N \leq 25$.

N	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)
	Final	Pic	Final	Pic	RSS	PSS	RSS
5	111	7,420	19,660	2,534,919	2.55×10^{6}	2.82×10^{12}	6.41×10^{-1}
10	256	32,810	43,260	11,619,239	1.01×10^{9}	4.59×10^{16}	1.74×10^{0}
15	451	84,050	77,160	30,783,259	4.70×10^{10}	1.84×10^{19}	3.44×10^{0}
20	696	169,015	122,360	63,741,479	8.05×10^{11}	1.43×10^{21}	8.69×10^{0}
25	991	295,580	179,860	114,418,399	7.68×10^{12}	4.36×10^{22}	1.28×10^{1}
50	3,216	1,828,030	686,860	785,640,999	1.04×10^{16}	2.09×10^{27}	1.13×10^{2}
100	11,416	12,541,055	3,198,360	6,312,076,199	1.73×10^{19}	1.17×10^{32}	3.03×10^{3}

TAB. 6.9 – Atelier avec Kanbans (taux constants) - Algorithme BFS

N	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
	Final	Pic	Final	Pic	RSS	PSS	Fonctions	RSS
5	99	6,536	17,556	2,252,297	2.55×10^{6}	7.84×10^{10}	8.73×10^{-4}	1.72×10^{-1}
10	234	29,911	39,436	10,617,221	1.01×10^{9}	3.80×10^{14}	5.70×10^{-3}	1.51×10^{0}
15	419	78,736	71,416	28,857,045	4.70×10^{10}	7.21×10^{16}	2.82×10^{-2}	3.04×10^{0}
20	654	161,636	114,496	60,804,181	8.05×10^{11}	3.24×10^{18}	8.15×10^{-2}	6.74×10^{0}
25	939	287,236	169,676	110,774,897	7.68×10^{12}	6.45×10^{19}	1.87×10^{-1}	1.11×10^{1}
50	3,114	1,857,611	662,076	793,624,997	1.04×10^{16}	8.05×10^{23}	2.08×10^{0}	9.66×10^{1}
100	11,214	13,160,236	3,129,376	6,662,168,661	1.73×10^{19}	1.15×10^{28}	1.89×10^{1}	2.97×10^{3}

TAB. 6.10 – Atelier avec Kanbans (partiellement fonctionnel) - Algorithme BFS

N	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
	Final	Pic	Final	Pic	RSS	PSS	Fonctions	RSS
5	57	9,939	11,092	2,652,393	2.55×10^{6}	2.18×10^{9}	5.02×10^{-2}	1.10×10^{-1}
10	102	190,559	18,892	44,658,109	1.01×10^{9}	3.14×10^{12}	3.15×10^{0}	8.05×10^{-1}
15	147	1,603,454	27,592	386,186,509	4.70×10^{10}	2.81×10^{14}	4.45×10^{1}	2.15×10^{0}
20	192	7,870,249	37,192	1,990,498,153	8.05×10^{11}	7.36×10^{15}	9.80×10^{2}	5.88×10^{0}
25	237	27,766,319	47,692	7,385,592,319	7.68×10^{12}	9.54×10^{16}	1.42×10^{4}	1.05×10^{1}

TAB. 6.11 – Atelier avec Kanbans (taux fonctionnels) - Algorithme BFS

Si on observe les tableaux des différents modèles SAN (TAB. 6.9, TAB. 6.10 et TAB. 6.11), on constate que les temps pour la génération du RSS sont très proches et la grande différence entre eux est le temps pour la génération des MDD $FuncMdd$ des fonctions des modèles.

Dans TAB. 6.12, TAB. 6.13 et TAB. 6.14, on présente les mesures obtenues utilisant l'algorithme de Saturation pour les modèles SAN (taux constants), modèles SAN (partiellement fonctionnel) et modèles SAN (taux fonctionnels) respectivement.

En observant les mesures de performance présentées dans TAB. 6.12, TAB. 6.13 et TAB. 6.14, on remarque que les temps pour la génération du RSS des modèles SAN utilisant l'algorithme de Saturation sont aussi très proches. On peut aussi observer que la génération du MDD $FuncMdd$ des modèles est le *goulet d'étranglement* de la méthode de la génération.

Dans cet exemple, il est intéressant de remarquer que les modèles SAN qui utilisent des taux fonctionnels ne sont pas toujours les plus performants par rapport aux modèles SAN avec taux constants. Il

N	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)
	Final	Pic	Final	Pic	RSS	PSS	RSS
5	111	142	19,660	55,755	2.55×10^{6}	2.82×10^{12}	2.74×10^{-3}
10	256	287	43,260	96,235	1.01×10^{9}	4.59×10^{16}	1.01×10^{-2}
15	451	482	77,160	153,415	4.70×10^{10}	1.84×10^{19}	2.48×10^{-2}
20	696	727	122,360	228,295	8.05×10^{11}	1.43×10^{21}	4.80×10^{-2}
25	991	1,022	179,860	321,875	7.68×10^{12}	4.36×10^{22}	8.41×10^{-2}
50	3,216	3,247	686,860	1,105,275	1.04×10^{16}	2.09×10^{27}	5.19×10^{-1}
100	11,416	11,447	3,198,360	4,649,575	1.73×10^{19}	1.17×10^{32}	3.51×10^{0}
500	257,016	257,047	209,170,360	241,923,975	7.09×10^{26}	1.58×10^{43}	3.57×10^{2}
1,000	1,014,016	1,014,047	1,501,335,360	1,630,816,975	1.42×10^{30}	1.02×10^{48}	3.38×10^{3}

TAB. 6.12 – Atelier avec Kanbans (taux constants) - Algorithme de Saturation

N	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
	Final	Pic	Final	Pic	RSS	PSS	Fonctions	RSS
5	99	368	17,556	103,313	2.55×10^{6}	7.84×10^{10}	8.53×10^{-4}	3.69×10^{-3}
10	234	1,653	39,436	406,413	1.01×10^{9}	3.80×10^{14}	5.55×10^{-3}	1.52×10^{-2}
15	419	4,638	71,416	1,150,813	4.70×10^{10}	7.21×10^{16}	2.91×10^{-2}	4.02×10^{-2}
20	654	10,073	114,496	2,593,513	8.05×10^{11}	3.24×10^{18}	8.46×10^{-2}	7.88×10^{-2}
25	939	18,708	169,676	5,031,513	7.68×10^{12}	6.45×10^{19}	1.93×10^{-1}	1.47×10^{-1}
50	3,114	136,133	662,076	45,346,013	1.04×10^{16}	8.05×10^{23}	2.14×10^{0}	9.64×10^{-1}
100	11,214	1,042,233	3,129,376	485,397,513	1.73×10^{19}	1.15×10^{28}	1.90×10^{1}	7.20×10^{0}

TAB. 6.13 – Atelier avec Kanbans (partiellement fonctionnel) - Algorithme de Saturation

N	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
	Final	Pic	Final	Pic	RSS	PSS	Fonctions	RSS
5	57	5,367	11,092	1,145,025	2.55×10^{6}	2.18×10^{9}	4.99×10^{-2}	4.28×10^{-3}
10	102	173,922	18,892	39,000,325	1.01×10^{9}	3.14×10^{12}	3.12×10^{0}	1.97×10^{-2}
15	147	1,567,302	27,592	373,397,525	4.70×10^{10}	2.81×10^{14}	4.42×10^{1}	5.96×10^{-2}
20	192	7,807,132	37,192	1,967,349,625	8.05×10^{11}	7.36×10^{15}	9.71×10^{2}	1.94×10^{-1}
25	237	27,668,787	47,692	7,348,569,625	7.68×10^{12}	9.54×10^{16}	1.40×10^{4}	7.01×10^{-1}

TAB. 6.14 – Atelier avec Kanbans (taux fonctionnels) - Algorithme de Saturation

est important de tenir compte du domaine de la fonction au moment de la modélisation du problème. L'utilisation d'une fonction qui possède un grand domaine peut limiter la génération du RSS du modèle à cause de la génération explicite du MDD $FuncMdd$ de cette fonction. Cette limitation est encore plus évidente dans l'exemple suivant.

6.4 Partage de Ressources

On présente dans cette section les mesures de performance obtenues utilisant les algorithmes BFS et de Saturation pour la génération du RSS des modèles SAN : *taux constants* (FIG. 3.10) et *taux fonctionnels* (FIG. 3.11). N est le nombre de clients distincts qui partagent l'utilisation de R ressources communes du système décrit dans la section 3.4 (page 47).

Les mesures obtenues pour les modèles SAN (taux constants) utilisant l'algorithme BFS pour cet exemple sont présentés dans TAB. 6.15, ainsi que les mesures pour les modèles SAN (taux fonctionnels) utilisant aussi l'algorithme BFS sont présentés dans TAB. 6.16.

	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)
R	Final	Pic	Final	Pic	RSS	PSS	RSS
N = 10							
1	21	179	5,344	91,532	1.10×10^{1}	2.05×10^{3}	1.05×10^{-2}
5	51	813	9,720	295,068	6.38×10^{2}	6.14×10^{3}	4.03×10^{-2}
9	65	899	11,856	340,596	1.02×10^{3}	1.02×10^{4}	4.08×10^{-2}
N = 20							
5	111	3,773	19,840	1,308,238	2.17×10^{4}	6.29×10^{6}	6.75×10^{-1}
10	176	5,723	29,360	1,986,768	6.17×10^{5}	1.15×10^{7}	8.13×10^{-1}
19	230	6,104	37,676	2,259,426	1.05×10^{6}	2.10×10^{7}	9.15×10^{-1}
N = 25							
5	141	6,078	24,900	2,087,823	6.84×10^{4}	2.01×10^{8}	9.11×10^{-1}
12	260	10,761	42,288	3,684,817	1.68×10^{7}	4.36×10^{8}	1.20×10^{0}
24	350	11,569	56,136	4,250,441	3.36×10^{7}	8.39×10^{8}	1.34×10^{0}
N = 100							
5	591	106,653	100,800	35,621,598	7.94×10^{7}	7.61×10^{30}	6.60×10^{0}
50	3,876	657,603	578,880	217,295,568	6.84×10^{29}	6.47×10^{31}	3.38×10^{1}
99	5,150	683,544	777,036	243,363,666	1.27×10^{30}	1.27×10^{32}	4.36×10^{1}

TAB. 6.15 – Partage de Ressources (taux constants) - Algorithme BFS

	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
R	Final	Pic	Final	Pic	RSS	PSS	Fonctions	RSS
N = 10								
1	19	139	4,908	51,801	1.10×10^{1}	1.02×10^{3}	1.85×10^{-4}	6.16×10^{-4}
5	35	2,777	7,228	607,513	6.38×10^{2}	1.02×10^{3}	1.05×10^{-2}	3.75×10^{-3}
9	19	532	4,940	139,337	1.02×10^{3}	1.02×10^{3}	5.76×10^{-4}	1.75×10^{-3}
N = 20								
5	95	64,711	17,384	12,932,283	2.17×10^{4}	1.05×10^{6}	7.73×10^{-1}	2.29×10^{-2}
10	120	2,783,330	20,968	565,087,339	6.17×10^{5}	1.05×10^{6}	6.17×10^{1}	1.33×10^{-1}
19	39	2,082	9,340	539,287	1.05×10^{6}	1.05×10^{6}	2.06×10^{-1}	7.42×10^{-3}
N = 25								
5	125	187,653	22,408	36,986,418	6.84×10^{4}	3.36×10^{7}	9.46×10^{0}	5.26×10^{-2}
24	49	3,232	11,540	835,862	3.36×10^{7}	3.36×10^{7}	7.41×10^{0}	1.19×10^{-2}

TAB. 6.16 – Partage de Ressources (taux fonctionnels) - Algorithme BFS

Dans les modèles SAN (taux fonctionnels) de cet exemple, l'automate $\mathcal{A}^{(N+1)}$ n'est pas représenté et N fonctions sont utilisées (f_{u_i}, où $i \in [1..N]$), et le domaine de chaque fonction est de 2^N états. En comparant les mesures de performance de TAB. 6.15 et TAB. 6.16, on remarque que les temps de la génération du RSS pour les modèles SAN (taux fonctionnels) sont plus rapides que les modèles SAN (taux constants). Toutefois, comme on l'a déjà remarqué dans l'exemple précédent, la génération des MDD $FuncMdd$ des fonctions est la partie plus lente de la méthode et elle limite beaucoup la génération du RSS de ces modèles (dans cet exemple, pour les $N > 25$).

Dans TAB. 6.17 et TAB. 6.18, on présente les mesures obtenues utilisant l'algorithme de Saturation pour les modèles SAN (taux constants) et modèles SAN (taux fonctionnels) respectivement.

	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)
R	Final	Pic	Final	Pic	RSS	PSS	RSS
N = 10							
1	21	42	5,344	30,096	1.10×10^{1}	2.05×10^{3}	1.04×10^{-3}
5	51	72	9,720	47,480	6.38×10^{2}	6.14×10^{3}	3.95×10^{-3}
9	65	86	11,856	56,992	1.02×10^{3}	1.02×10^{4}	5.28×10^{-3}
N = 20							
5	111	152	19,840	156,110	2.17×10^{4}	6.29×10^{6}	1.65×10^{-2}
10	176	217	29,360	223,890	6.17×10^{5}	1.15×10^{7}	3.04×10^{-2}
19	230	271	37,676	289,362	1.05×10^{6}	2.10×10^{7}	4.37×10^{-2}
N = 25							
5	141	192	24,900	232,775	6.84×10^{4}	2.01×10^{8}	2.65×10^{-2}
12	260	311	42,288	379,551	1.68×10^{7}	4.36×10^{8}	5.87×10^{-2}
24	350	401	56,136	511,847	3.36×10^{7}	8.39×10^{8}	8.84×10^{-2}
N = 100							
5	591	792	100,800	3,170,750	7.94×10^{7}	7.61×10^{30}	5.85×10^{-1}
50	3,876	4,077	578,880	16,893,170	6.84×10^{29}	6.47×10^{31}	7.54×10^{0}
99	5,150	5,351	777,036	23,873,122	1.27×10^{30}	1.27×10^{32}	1.16×10^{1}

TAB. 6.17 – Partage de Ressources (taux constants) - Algorithme de Saturation

	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
R	Final	Pic	Final	Pic	RSS	PSS	Fonctions	RSS
N = 10								
1	19	110	4,908	38,397	1.10×10^{1}	1.02×10^{3}	1.91×10^{-4}	8.66×10^{-4}
5	35	2,593	7,228	555,297	6.38×10^{2}	1.02×10^{3}	1.04×10^{-2}	3.53×10^{-3}
9	19	255	4,940	74,493	1.02×10^{3}	1.02×10^{3}	5.65×10^{-4}	2.09×10^{-3}
N = 20								
5	95	63,937	17,384	12,724,827	2.17×10^{4}	1.05×10^{6}	8.02×10^{-1}	3.07×10^{-2}
10	120	2,782,111	20,968	564,781,503	6.17×10^{5}	1.05×10^{6}	6.19×10^{1}	4.09×10^{-2}
19	39	925	9,340	277,423	1.05×10^{6}	1.05×10^{6}	2.07×10^{-1}	9.15×10^{-3}
N = 25								
5	125	186,434	22,408	36,664,142	6.84×10^{4}	3.36×10^{7}	9.13×10^{0}	5.79×10^{-2}
24	49	1,410	11,540	426,438	3.36×10^{7}	3.36×10^{7}	7.50×10^{0}	1.46×10^{-2}

TAB. 6.18 – Partage de Ressources (taux fonctionnels) - Algorithme de Saturation

La limitation de l'utilisation d'une fonction qui possède un grand domaine est aussi observée dans TAB. 6.17 et TAB. 6.18 pour les modèles SAN (taux constants et fonctionnels) utilisant l'algorithme de Saturation. Au contraire des mesures de performance présentées dans TAB. 6.15 et TAB. 6.16 (pour l'algorithme BFS), la différence entre les temps de la génération du RSS des modèles (taux constants et fonctionnels) utilisant l'algorithme de Saturation est assez petite.

Il est intéressant de remarquer que l'utilisation de la mémoire pour les modèles SAN (taux fonctionnels) augmente significativement en fonction de N et R du modèle, sachant que la génération des MDD $FuncMdd$ des fonctions du modèle dépendent de ces deux paramètres.

6.5 Conclusion

Dans ce chapitre, on a présenté les mesures de performance des méthodes de génération de l'espace d'états atteignables de modèles SAN à temps continu décrits dans le chapitre 3, utilisant les algorithmes BFS et de Saturation.

Les mesures présentées dans ce chapitre soulignent la différence entre les deux algorithmes (BFS et de Saturation), où **l'algorithme de Saturation gère l'utilisation de mémoire de façon plus performante**. Le pic de mémoire est plus proche de la mémoire finale utilisée et la génération du RSS des modèles est plus rapide. Comme le pic de mémoire de l'algorithme de Saturation reste petit par rapport au pic de mémoire de l'algorithme BFS, l'algorithme de Saturation permet la génération du RSS de modèles qui ne sont pas possibles utilisant l'algorithme BFS (*e.g.*, $N = 1,000$ pour le modèle de l'*Atelier avec Kanbans*).

Dans le formalisme SAN à temps continu, l'utilisation d'éléments fonctionnels (tels comme des taux et probabilités fonctionnels) permet une plus grande flexibilité pour l'expression de l'interaction entre les composants du système. On a montré également une comparaison entre les mesures de la génération du RSS de modèles SAN (taux constants) et modèles SAN (taux fonctionnels), utilisant les algorithmes BFS et de Saturation. **La génération du RSS de modèles SAN (taux fonctionnels) qui possèdent *petites* fonctions** (*i.e.*, le domaine de la fonction est inférieur à quelques millions d'états) **peut être plus performante que la génération du RSS de modèles SAN (taux constants)**, puisque le nombre de termes tensoriels du descripteur d'atteignabilité est plus petit et la génération des MDD $FuncMdd$ des fonctions est faite de façon rapide. Toutefois, pour les modèles SAN (taux fonctionnels) qui utilisent des fonctions à grand espace d'états (*e.g.*, le modèle de l'exemple de *Partage de Ressources*, où le domaine des fonctions est égal à l'espace d'états produit du modèle), la génération des MDD $FuncMdd$ de ces fonctions est la partie qui demande le plus de calcul, vu que le MDD $FuncMdd$ d'une fonction est obtenu *explicitement* en évaluant état-par-état de son espace d'états. On remarque que la génération des MDD $FuncMdd$ des fonctions à grand espace d'états de façon explicite est un travail à améliorer afin de permettre la génération du RSS de modèles SAN (taux fonctionnels) autant grands que les modèles SAN (taux constants), *e.g.*, $N = 100$ pour le modèle de *Partage de Ressources*.

Enfin, par rapport à l'état d'art, les méthodes présentées dans le chapitre 5 **sont nouvelles pour la génération de l'espace d'états atteignables de modèles qui utilisent des fonctions**. Comme on l'a observé dans les mesures présentées, ces nouvelles méthodes permettent la génération de l'espace d'états atteignables de modèles SAN de façon rapide et utilisant peu de mémoire. Elles permettent aussi la représentation de très grands espaces d'états atteignables par un MDD (*e.g.*, pour le modèle du *Dîner des Philosophes*, où $N = 1,000$, le RSS est d'ordre de 10^{382} états). La représentation du RSS de modèles SAN par un MDD constitue une amélioration importante lorsqu'on compare avec la représentation vectorielle courante du RSS de modèles SAN.

Deuxième partie

Réseaux d'Automates Stochastiques à temps discret

Chapitre 7

Formalisme des Réseaux d'Automates Stochastiques (SAN) à temps discret

Depuis la proposition du formalisme des Réseaux d'Automates Stochastiques (*Stochastic Automata Networks* - SAN) par Plateau dans [110, 111], la plupart des travaux ont porté sur une échelle de temps continue. Le formalisme à temps discret a reçu beaucoup moins d'attention et cela s'explique par la difficulté de modéliser des systèmes complexes avec des distributions discrètes.

Dans ce chapitre, on va donner les définitions de base du formalisme SAN à temps discret[1] (Section 7.2) qui nous permettrons de définir un algorithme pour la construction de la chaîne de Markov représentée (Section 7.3). Mais d'abord, pour mieux comprendre le problème, on va introduire dans la section 7.1 les motivations et travaux précédents sur les SAN et aussi sur d'autres formalismes structurés à temps discret.

7.1 Motivation et travaux précédents

Contrairement aux systèmes modélisés sur une échelle de temps continue, où *un seul événement* peut avoir lieu à chaque instant de temps, en temps discret, *plusieurs événements* peuvent avoir lieu dans une même unité de temps et donc, chaque combinaison d'événement possible doit être déterminée. Par exemple, pour deux événements e_1 et e_2, quatre combinaisons sont possibles : (1) e_1 a lieu et e_2 n'a pas lieu ; (2) e_1 n'a pas lieu et e_2 a lieu ; (3) e_1 a lieu et e_2 a lieu aussi ; et (4) e_1 n'a pas lieu et e_2 n'a pas lieu non plus.

Cependant, la grande difficulté de la modélisation de systèmes à temps discret n'est pas de déterminer toutes les combinaisons possibles, mais de résoudre les conflits, lorsque le tirage d'un événement amène à un état où l'autre n'est plus réalisable.

On trouve dans la littérature un certain nombre de propositions de formalismes pour modéliser des systèmes à temps discret : *Réseaux de Petri Stochastiques* [97, 29], *Réseaux de Files d'Attente* [68] et *Réseaux d'Automates Stochastiques* [112, 8].

[1] La présentation du formalisme SAN à temps discret donnée dans ce chapitre est commune dans cette thèse et dans [15].

Au niveau des Réseaux de Petri Stochastiques, différentes sémantiques ont été proposées pour traiter le problème des transitions concurrentes (deux transitions réalisables dans la même unité de temps) [97, 29, 136, 121]. Pour illustrer ceci, supposons que plusieurs transitions soient possibles au même instant de temps, par exemple les transitions t_1 et t_2 dans FIG. 7.1. Ces deux transitions sont dites *concurrentes*, car le déclenchement d'une transition empêche l'occurrence de l'autre.

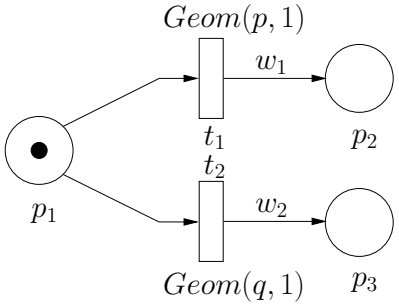


FIG. 7.1 – Exemple d'un réseau de Petri avec transitions concurrentes

Une des premières sémantiques pour traiter la concurrence a été proposée par Molloy dans [97]. Dans cette proposition, deux transitions concurrentes ne peuvent pas avoir lieu en même temps. Notons que, sur le graphe de marquage du réseau de Petri obtenu par cette sémantique, on insère des probabilités de choix, comme indiqué sur l'exemple de FIG. 7.2, où p est la probabilité d'occurrence de la transition t_1 et q la probabilité d'occurrence de t_2.

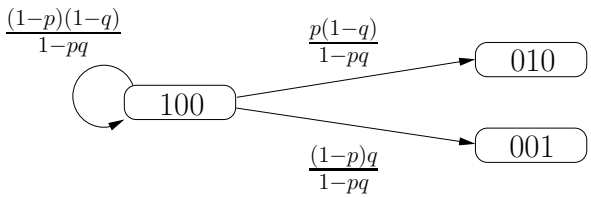


FIG. 7.2 – Graphe de marquage du modèle de FIG. 7.1 selon Molloy

Une autre sémantique a été proposée par Ciardo dans [29]. Dans cette sémantique, Ciardo propose d'associer un poids à chaque transition, *i.e.*, w_1 et w_2 pour le modèle de FIG. 7.1, et lorsque les deux transitions sont potentiellement réalisables en même temps, la probabilité de la transition effectivement réalisée est pondérée par la probabilité de chaque transition. FIG. 7.3 montre le graphe de marquage obtenu par cette sémantique.

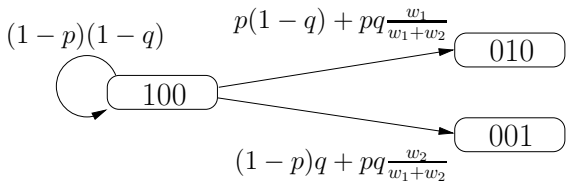


FIG. 7.3 – Graphe de marquage du modèle de FIG. 7.1 selon Ciardo

Dans ces deux approches, la simultanéité est écartée alors qu'elle peut avoir un sens physique dans certains systèmes.

Au niveau des Réseaux d'Automates Stochastiques (SAN), les travaux sont moins nombreux [112, 8]. La première proposition de SAN à temps discret a été faite par Plateau et Atif dans [112]. Dans cette approche, la notion d'*événements compatibles* a été introduite pour définir les ensembles d'événements qui peuvent se réaliser de façon simultanée. Cependant, la sémantique à adopter en cas de conflit n'est pas très explicite et le descripteur markovien obtenu devient souvent très complexe à cause du grand nombre de combinaisons possibles d'événements compatibles. Certains méthodes ont été proposées pour simplifier cette approche, soit par l'agrégation/désagrégation des automates [70], soit par les propriétés structurelles du modèle [59], mais ces méthodes s'appliquent à un sous-ensemble très réduit de modèles.

Une nouvelle proposition a été faite par Benoit dans [8]. Dans cette proposition, Benoit introduit la notion d'*ordre de priorité* entre les événements du modèle. L'ordre de priorité permet de définir le comportement à suivre en cas de conflit et rend possible la construction d'une sémantique de chaîne de Markov.

La définition formelle présentée dans la section 7.2 utilise la sémantique des SAN à temps discret proposée par Benoit dans [8]. Dans la section 7.3, nous définissons les règles de construction de l'automate global d'un modèle SAN. La section 7.4 présente la procédure d'obtention de la chaîne de Markov représentée par le modèle SAN. Enfin, la section 7.5 présente les conclusions de ce chapitre.

7.2 Description formelle des SAN à temps discret

Dans cette section, on va présenter les notations et les définitions de base qui seront utilisées pour la construction du Descripteur Markovien d'un modèle SAN à temps discret.

On va considérer un modèle SAN comprenant N automates et E événements.

✍ **Notation**

$\mathcal{A}$	l'ensemble des automates[2] ($\mid \mathcal{A} \mid = N$).

7.2.1 Définitions et notations de base pour un automate dans un réseau

On note $\mathcal{A}^{(i)}$, où $i \in [1..N]$, le i-ème automate d'un modèle SAN.

Pour les notations suivantes, on adopte $[i..j]$ pour le sous-ensemble de $\mathbb{N}$ contenant toutes les valeurs de i jusqu'à j (ces valeurs incluses) et $[i, j]$ pour le sous-ensemble de $\mathbb{R}$ (des valeurs continues) contenant toutes les valeurs de i jusqu'à j.

✍ **Notation**

$\mathcal{S}^{(i)}$	l'ensemble d'états de l'automate $\mathcal{A}^{(i)}$;
$x^{(i)}$	l'état local de l'automate $\mathcal{A}^{(i)}$, où $x^{(i)} \in \mathcal{S}^{(i)}$.

[2] On adopte la notation $\mid \mathcal{X} \mid$ afin de définir la cardinalité d'un ensemble $\mathcal{X}$.

Définition 7.2.1. *À tous les ensembles $\mathcal{S}^{(i)}$ on rajoute un état Φ, appelé état* **fantôme**.

Cet état sera utilisé comme état intermédiaire transitoire lorsque plusieurs événements ont lieu dans la même unité de temps. On verra l'utilité de l'état fantôme plus en détail sur l'automate $\mathcal{A}^{(1)}$ dans FIG. 7.4 (page 120).

✍ **Notation**

$\breve{\mathcal{S}}^{(i)}$	l'ensemble d'états d'un automate $\mathcal{A}^{(i)}$ d'un modèle SAN, $\breve{\mathcal{S}}^{(i)} = \mathcal{S}^{(i)} \cup \Phi$.

Définition 7.2.2. *L'espace d'états produit $\hat{\mathcal{S}}$ d'un modèle SAN est défini par le produit cartésien des espaces d'états $\mathcal{S}^{(i)}$, i.e., $\hat{\mathcal{S}} = \prod_{i=1}^{N} \mathcal{S}^{(i)}$.*

✍ **Notation**

$\tilde{x}$	l'état global d'un modèle SAN défini par la combinaison des états locaux des N automates, $\tilde{x} = (x^{(1)}, \ldots, x^{(N)})$, où $\tilde{x} \in \hat{\mathcal{S}}$;
ω	un ensemble d'indices d'automates, où $\omega \subseteq [1..N]$;
$\tilde{x}^{(\omega)}$	le vecteur des états locaux $x^{(i)}$ tel que $i \in \omega$.

On remarque que la définition d'un état local d'un automate ($x^{(i)}$) et la définition d'un état global ($\tilde{x}$) peuvent être vues comme des cas particuliers de $\tilde{x}^{(\omega)}$. Un état local $x^{(i)}$ est le cas où $\omega = \{i\}$, et l'état global $\tilde{x}$ est le cas où $\omega = \{1, 2, 3, \ldots, N\}$.

✍ **Notation**

$\hat{\mathcal{S}}^{(\omega)}$	l'espace d'états produit de l'ensemble des états locaux des automates $\mathcal{A}^{(i)}$, où $i \in \omega$;
$f(\hat{\mathcal{S}}^{(\omega)})$	une fonction de $\hat{\mathcal{S}}^{(\omega)} \to \mathbb{R}^{+}$, où l'ensemble d'indices d'automates $\omega \subseteq [1..N]$;
$f(\tilde{x}^{(\omega)})$	la fonction $f(\hat{\mathcal{S}}^{(\omega)})$ évaluée pour le vecteur d'états locaux $\tilde{x}^{(\omega)} \in \hat{\mathcal{S}}^{(\omega)}$.

Notons que les états $x^{(i)}$, où $i \in \omega$, sont les paramètres d'évaluation pour la fonction $f(\hat{\mathcal{S}}^{(\omega)})$, *i.e.*, l'espace d'états $\hat{\mathcal{S}}^{(\omega)}$ est le domaine de définition de la fonction f.

✍ **Notation**

$\mathcal{E}$	l'ensemble d'événements du modèle SAN considéré ;
e	l'identificateur d'un événement, où $e \in \mathcal{E}$.

Définition 7.2.3. *Pour $e \in \mathcal{E}$, on appelle tuple d'événement, le triplet (e, ρ_e, ϱ_e) composé de :*

1. *e, l'identificateur de l'événement ;*
2. *ρ_e, la fonction définie de $\hat{\mathcal{S}} \to [0, 1]$, la probabilité d'occurrence de l'événement e. L'occurrence de l'événement e est une variable aléatoire de loi de Bernoulli de paramètre ρ_e. Pour $e \neq e'$ les lois sont indépendantes ;*
3. *ϱ_e, la priorité de l'événement. La priorité est une valeur entière strictement positive, 1 est la priorité maximum et $+\infty$ est la priorité minimum. Deux événements différents ne peuvent pas avoir la même priorité.*

La définition 7.2.3 associe une probabilité d'occurrence ρ_e et une priorité ϱ_e à un événement e. La priorité détermine l'ordre d'occurrence lorsque plusieurs événements sont possibles dans la même unité de temps.

Définition 7.2.4. *Pour $x^{(i)} \in \breve{\mathcal{S}}^{(i)}$ et $y^{(i)} \in \breve{\mathcal{S}}^{(i)}$, on appelle tuple de transition locale un couple $(e, \pi_e(x^{(i)}, y^{(i)}))$:*

1. *e, l'identificateur d'un événement de $\mathcal{E}$;*
2. *$\pi_e(x^{(i)}, y^{(i)})$, une fonction définie de $\hat{\mathcal{S}} \rightarrow [0, 1]$, qui est appelée la* **probabilité de routage** *de ce tuple.*

✍ **Notation**

$\mathcal{T}$	l'ensemble des tuples de transition locale d'un modèle SAN ;
$2^{\mathcal{T}}$	l'ensemble de parties de $\mathcal{T}$. Une partie de $\mathcal{T}$ sera appelée **élément de transition**.

Définition 7.2.5. *$\mathcal{P}^{(i)}$ est la matrice de transition locale de $\mathcal{S}^{(i)} \times \breve{\mathcal{S}}^{(i)} \rightarrow 2^{\mathcal{T}}$, qui définit les éléments de transition de l'automate $\mathcal{A}^{(i)}$.*

✍ **Notation**

$\mathcal{P}^{(i)}(x^{(i)}, y^{(i)})$	l'élément de transition de l'état local $x^{(i)} \in \mathcal{S}^{(i)}$ vers l'état local $y^{(i)} \in \breve{\mathcal{S}}^{(i)}$, qui contient un sous-ensemble des tuples de transition locale dans $\mathcal{T}$.

La matrice de transition locale $\mathcal{P}^{(i)}$ d'un automate $\mathcal{A}^{(i)}$ (Définition 7.2.5) indique la relation entre les états de l'automate $\mathcal{A}^{(i)}$ et les événements qui peuvent déclencher une transition d'un état à l'autre dans cet automate. La transition peut être déclenchée par n'importe quel événement qui figure dans un tuple de l'élément de transition associé. On remarque que l'état fantôme Φ n'est *jamais l'état de départ* d'une transition.

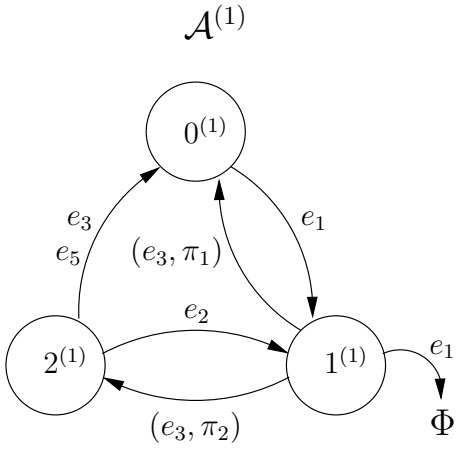
Définition 7.2.6. *Un automate $\mathcal{A}^{(i)}$ est défini par :*

1. *un ensemble d'états $\breve{\mathcal{S}}^{(i)}$;*
2. *un ensemble d'événements $\mathcal{E}$;*
3. *une matrice de transition locale $\mathcal{P}^{(i)}$.*

Pour représenter graphiquement un automate stochastique, on associe un graphe à l'automate. Les noeuds du graphe représentent les états de l'automate, et les arcs représentent les transitions entre ces états. Les arcs sont étiquetés par un élément de transition où figurent les événements qui peuvent déclencher la transition.

Dans FIG. 7.4, on décrit un automate avec 3 états plus l'état fantôme Φ ($\breve{\mathcal{S}}^{(1)} = \{0^{(1)}; 1^{(1)}; 2^{(1)}; \Phi\}$). On remarque que lorsque la probabilité de routage est égale à 1.0, alors on remplace sur le dessin $(e_1, 1.0(0^{(1)}, 1^{(1)}))$ par e_1 afin d'alléger la présentation, une fois qu'on connaît l'état de départ et d'arrivée de la transition par le dessin.

Notons qu'il se peut qu'un même événement, en partant du même état, puisse mener dans plusieurs états différents. Pour définir la probabilité d'aller dans chacun des états d'arrivée, on utilise la *probabilité de routage* du tuple de transition locale associé.



Événement	Prob.	Priorité
e_1	ρ_1	3
e_2	ρ_2	1
e_3	ρ_3	2
e_5	ρ_5	5

FIG. 7.4 – Représentation graphique de l'automate $\mathcal{A}^{(1)}$

À partir de cette représentation graphique, on peut déduire la matrice de transition locale $\mathcal{P}^{(1)}$ de l'automate $\mathcal{A}^{(1)}$:

$\mathcal{P}^{(1)}$	$0^{(1)}$	$1^{(1)}$	$2^{(1)}$	Φ
$0^{(1)}$	$\emptyset$	$\{(e_1, 1.0)\}$	$\emptyset$	$\emptyset$
$1^{(1)}$	$\{(e_3, \pi_1)\}$	$\emptyset$	$\{(e_3, \pi_2)\}$	$\{(e_1, 1.0)\}$
$2^{(1)}$	$\{(e_3, 1.0); (e_5, 1.0)\}$	$\{(e_2, 1.0)\}$	$\emptyset$	$\emptyset$

TAB. 7.1 – Matrice de transition locale $\mathcal{P}^{(1)}$ de l'automate $\mathcal{A}^{(1)}$ de FIG. 7.4

On remarque que les indices $x^{(i)}$ et $y^{(i)}$ associés aux probabilités de routages ont été enlevés de la matrices de transition locale pour également alléger la présentation.

Définition 7.2.7. *Soit un état local $x^{(i)} \in \mathcal{S}^{(i)}$ d'un automate $\mathcal{A}^{(i)}$ et un événement $e \in \mathcal{E}$, on définit $succ_e(x^{(i)})$ l'**ensemble d'états successeurs** $y^{(i)} \in \mathcal{S}^{(i)}$ de $x^{(i)}$. Ces états $y^{(i)}$ sont ceux tels que $\mathcal{P}^{(i)}(x^{(i)}, y^{(i)})$ possède au moins un tuple de transition locale avec l'identificateur e où : $\rho_e \neq 0$ et $\pi_e(x^{(i)}, y^{(i)}) \neq 0$.*

On remarque que l'ensemble d'états successeurs de $x^{(i)}$ grâce à l'événement e peut être vide (*i.e.*, $succ_e(x^{(i)}) = \emptyset$), cas où aucune transition ne peut avoir lieu dans $x^{(i)}$ par l'événement e.

Par exemple, l'ensemble d'états successeurs de chaque état de l'automate $\mathcal{A}^{(1)}$, présenté dans FIG. 7.4, pour chaque événement $e \in \mathcal{E}$ est :

Événement	Successeurs de $x^{(1)} = 0^{(1)}$	Successeurs de $x^{(1)} = 1^{(1)}$	Successeurs de $x^{(1)} = 2^{(1)}$
$succ_{e_1}(x^{(1)})$	$\{1^{(1)}\}$	$\emptyset$	$\emptyset$
$succ_{e_2}(x^{(1)})$	$\emptyset$	$\emptyset$	$\{1^{(1)}\}$
$succ_{e_3}(x^{(1)})$	$\emptyset$	$\{0^{(1)}, 2^{(1)}\}$	$\{0^{(1)}\}$
$succ_{e_5}(x^{(1)})$	$\emptyset$	$\emptyset$	$\{0^{(1)}\}$

TAB. 7.2 – Ensemble d'états successeurs de l'automate $\mathcal{A}^{(1)}$ de FIG. 7.4

Définition 7.2.8. *Soit un état local $x^{(i)} \in \mathcal{S}^{(i)}$ d'un automate $\mathcal{A}^{(i)}$, on défini $pot(x^{(i)})$ l'ensemble d'événements $e \in \mathcal{E}$ tel qu'il existe $y^{(i)} \in \breve{\mathcal{S}}^{(i)}$, où $\mathcal{P}^{(i)}(x^{(i)}, y^{(i)})$ possède un tuple de transition locale $(e, \pi_e(x^{(i)}, y^{(i)}))$. On appelle $pot(x^{(i)})$ l'**ensemble des événements possibles** à partir de $x^{(i)}$.*

La définition 7.2.8 définit l'ensemble d'événements possibles à partir de l'état $x^{(i)}$. On remarque aussi que si $succ_e(x^{(i)})$ n'est pas vide, alors $e \in pot(x^{(i)})$.

Dans TAB. 7.3, on présente les ensembles d'événements possibles pour chaque état de l'automate $\mathcal{A}^{(1)}$ de FIG. 7.4.

État local	Événements possibles
$pot(0^{(1)})$	$\{e_1\}$
$pot(1^{(1)})$	$\{e_1; e_3\}$
$pot(2^{(1)})$	$\{e_2; e_3; e_5\}$

TAB. 7.3 – Événements possibles à partir de chaque état de l'automate $\mathcal{A}^{(1)}$ de FIG. 7.4

Étant donné un état $x^{(i)} \in \mathcal{S}^{(i)}$ et l'ensemble $pot(x^{(i)})$ (Définition 7.2.8), la sémantique probabiliste donnée à cet automate est la suivante :

1. l'événement le plus prioritaire e_1 de $pot(x^{(i)})$ est déclenché à partir de $x^{(i)}$ avec probabilité ρ_{e_1}, ce qui mène à l'état $x_1^{(i)}$;
2. si l'événement suivant le plus prioritaire de $pot(x^{(i)})$ **est réalisable** dans $x_1^{(i)}$ (ce qui se voit car il doit être dans $pot(x_1^{(i)})$), alors son occurrence amène à $x_2^{(i)}$ et on répète les étapes 2 et 3 jusqu'à avoir examiné tous les événements de $pot(x^{(i)})$;
3. si l'événement suivant le plus prioritaire de $pot(x^{(i)})$ **n'est pas réalisable** dans $x_1^{(i)}$ (i.e. l'événement n'est pas dans $pot(x_1^{(i)})$), alors on l'ignore et on répète les étapes 2 et 3 jusqu'à avoir examiné tous les événements de $pot(x^{(i)})$.

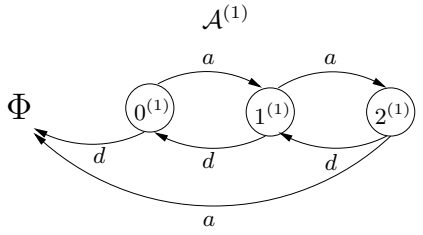
On remarque que cette sémantique autorise la réalisation de plusieurs événements dans la même unité de temps. Par exemple, pour une file d'attente à temps discret, dans un état où le nombre de client est $n > 0$, une arrivée et un départ de la file peuvent survenir dans la même unité de temps.

Dans l'exemple précédent et pour l'état $1^{(1)}$, l'ensemble d'événements possibles inclut l'événement e_1 associée à la transition vers l'état fantôme Φ. Ceci signifie que e_1 est dans $pot(1^{(1)})$. Cependant, cet événement sera *réalisable* uniquement si l'occurrence de l'événement e_3, *plus prioritaire*, amène à l'état

$0^{(1)}$, où e_1 *est réalisable* (donc dans $pot(0^{(1)})$) et amène vers un état "vrai" de l'automate (pas l'état fantôme Φ).

Prenons par exemple l'état $1^{(1)}$ et son ensemble $pot(1^{(1)}) = \{e_1; e_3\}$. Lorsque l'événement le plus prioritaire (e_3) a lieu, deux états sont possibles ($succ_{e_3}(1^{(1)})$) : $0^{(1)}$ avec probabilité π_1 et $2^{(1)}$ avec probabilité π_2. Pour l'état successeur $0^{(1)}$, l'événement suivant le plus prioritaire (e_1) est réalisable et mène à l'état $1^{(1)}$. Cette suite d'occurrence $((e_3, \pi_1), (e_1, 1.0))$ représente une transition de la chaîne de Markov de l'état $1^{(1)}$ vers lui-même. Pour l'état successeur $2^{(1)}$, l'événement suivant le plus prioritaire (e_1) n'est pas réalisable ($e_1 \notin pot(2^{(1)})$), alors on l'ignore. Étant donné qu'il n'y a plus d'événement possible, la suite d'occurrence (formée uniquement par (e_3, π_2)) provoque une transition de la chaîne de Markov de l'état $1^{(1)}$ vers l'état $2^{(1)}$.

Pour une file d'attente, l'état fantôme Φ peut, par exemple, autoriser l'arrivée et le départ d'un client dans la même unité de temps, si $n = 0$. C'est la modélisation qui autorise cet enchaînement. L'exemple de file d'attente de FIG. 7.5 présente cette modélisation.



Événement	Prob.	Priorité
a	ρ_a	ϱ_a
d	ρ_d	ϱ_d

FIG. 7.5 – File d'attente avec arrivée et départ pour $n = 0$

Dans cet exemple (FIG. 7.5), si $\varrho_d < \varrho_a$ (*i.e.*, si l'événement d est plus prioritaire que l'événement a) à partir de l'état $2^{(1)}$ la réalisation simultanée des événements d (départ tardif) et a est possible. Si $\varrho_a < \varrho_d$ (*i.e.*, si l'événement a est plus prioritaire que l'événement d), alors à partir de l'état $0^{(1)}$ la réalisation simultanée des événements a (arrivée tardive) et d est possible grâce à l'état fantôme Φ.

7.2.1.1 Automate complété

Dans la composition parallèle d'automates pour les SAN (voir dans la section suivante) on aura besoin d'exprimer le fait que, par exemple, si l'événement e_1 de l'automate $\mathcal{A}^{(1)}$ et l'événement e_4 de l'automate $\mathcal{A}^{(2)}$ sont réalisables au même instant, alors quatre possibilités existent : (1) e_1 a lieu et e_4 n'a pas lieu ; (2) e_1 n'a pas lieu et e_4 a lieu ; (3) e_1 a lieu et e_4 a lieu aussi ; et (4) e_1 n'a pas lieu et e_4 n'a pas lieu non plus.

Pour ce faire, nous avons besoin de représenter de façon explicite sur l'automate le résultat de la *non occurrence* d'un événement. A cet effet, on va introduire des événements *factices*, appelés "*événements complémentaires*" pour chaque événement présent dans un automate.

Définition 7.2.9. *À tous les événements e d'un modèle SAN, où $e \in \mathcal{E}$ dont le triplet est (e, ρ_e, ϱ_e), on associe un événement complémentaire défini par le triplet $(\bar{e}, \rho_{\bar{e}}, \varrho_{\bar{e}})$ où :*

1. *$\bar{e}$, l'identificateur de l'événement complémentaire associé à l'événement e ;*
2. *$\rho_{\bar{e}}$, la fonction définie par $1 - \rho_e$, représentant la probabilité d'occurrence de l'événement complémentaire $\bar{e}$;*
3. *$\varrho_{\bar{e}}$, la priorité de l'événement complémentaire. La priorité d'un événement complémentaire est égale à la priorité de l'événement auquel l'événement complémentaire est associé (i.e., $\varrho_{\bar{e}} = \varrho_e$).*

✍ **Soit**

$\bar{\mathcal{E}}$	l'ensemble d'événements complémentaires de $\mathcal{E}$;
$\breve{\mathcal{E}}$	l'union des ensembles d'événements $\mathcal{E}$ et $\bar{\mathcal{E}}$ ($\breve{\mathcal{E}} = \mathcal{E} \cup \bar{\mathcal{E}}$).

Définition 7.2.10. *À tous les événements complémentaires $\bar{e} \in \bar{\mathcal{E}}$, on associe des tuples de transition locale défini par :*

1. $\forall x^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $\exists (e, \pi_e(x^{(i)}, y^{(i)})) \in \mathcal{P}^{(i)}(x^{(i)}, y^{(i)})$ *et* $x^{(i)} \neq y^{(i)}$
   $(\bar{e}, \pi_{\bar{e}}(x^{(i)}, x^{(i)}))$
2. $\forall x^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $\exists (e, \pi_e(x^{(i)}, \Phi)) \in \mathcal{P}^{(i)}(x^{(i)}, \Phi)$
   $(\bar{e}, \pi_{\bar{e}}(x^{(i)}, \Phi))$

Tous les tuples de transition locale des événements complémentaires ont comme probabilité de routage 1.0. L'occurrence d'un événement complémentaire amène toujours vers l'état de départ.

✍ **Notation**

$\bar{\mathcal{T}}$	l'ensemble des tuples de transition locale des événements complémentaires de l'ensemble $\bar{\mathcal{E}}$;
$\breve{\mathcal{T}}$	l'union des ensembles des tuples de transition locale $\mathcal{T}$ et $\bar{\mathcal{T}}$ ($\breve{\mathcal{T}} = \mathcal{T} \cup \bar{\mathcal{T}}$) ;
$2^{\breve{\mathcal{T}}}$	l'ensemble des parties de $\breve{\mathcal{T}}$.

Ces événements complémentaires sont utiles pour la composition parallèle dans le SAN. On appelle *automate complété*, l'automate où apparaissent les tuples des événements complémentaires.

Définition 7.2.11. *L'automate complété, noté $\breve{\mathcal{A}}^{(i)}$, d'un automate $\mathcal{A}^{(i)}(\breve{\mathcal{S}}^{(i)}, \mathcal{E}, \mathcal{P}^{(i)})$ est défini par :*

1. *l'ensemble d'états $\breve{\mathcal{S}}^{(i)}$;*
2. *un ensemble d'événements $\breve{\mathcal{E}}$;*
3. *une matrice de transition locale $\breve{\mathcal{P}}^{(i)} : \mathcal{S}^{(i)} \times \breve{\mathcal{S}}^{(i)} \rightarrow 2^{\breve{\mathcal{T}}}$:*

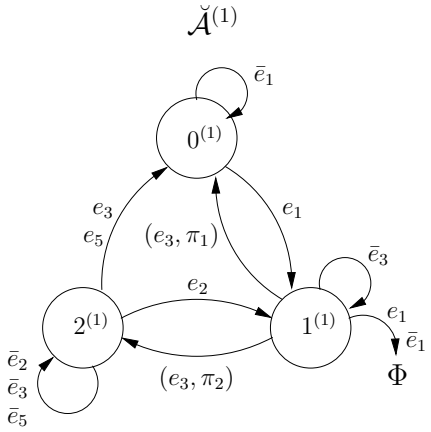
$$\forall x^{(i)} \in \mathcal{S}^{(i)}, \forall y^{(i)} \in \mathcal{S}^{(i)} \text{ tel que } x^{(i)} \neq y^{(i)}$$
$$\breve{\mathcal{P}}^{(i)}(x^{(i)}, y^{(i)}) = \mathcal{P}^{(i)}(x^{(i)}, y^{(i)})$$
$$\forall x^{(i)} \in \mathcal{S}^{(i)}$$
$$\breve{\mathcal{P}}^{(i)}(x^{(i)}, x^{(i)}) =$$
$$\mathcal{P}^{(i)}(x^{(i)}, x^{(i)}) \bigcup_{\forall y^{(i)} \in \mathcal{S}^{(i)}, \forall e \in \mathcal{E} \text{ tel que } \exists\, (e, \pi_e(x^{(i)}, y^{(i)})) \in \mathcal{P}^{(i)}(x^{(i)}, y^{(i)})} \{(\bar{e}, \pi_{\bar{e}}(x^{(i)}, x^{(i)}))\}$$
$$\forall x^{(i)} \in \mathcal{S}^{(i)}$$
$$\breve{\mathcal{P}}^{(i)}(x^{(i)}, \Phi) = \mathcal{P}^{(i)}(x^{(i)}, \Phi) \bigcup_{\forall e \in \mathcal{E} \text{ tel que } \exists\, (e, \pi_e(x^{(i)}, \Phi)) \in \mathcal{P}^{(i)}(x^{(i)}, \Phi)} \{(\bar{e}, \pi_{\bar{e}}(x^{(i)}, \Phi))\}$$

L'automate complété $\breve{\mathcal{A}}^{(i)}$ est défini sur le même ensemble d'états $\breve{\mathcal{S}}^{(i)}$ de son automate $\mathcal{A}^{(i)}$. L'ensemble d'événements $\breve{\mathcal{E}}$ de l'automate complété contient les événements du modèle SAN et aussi les événements complémentaires ajoutés au modèle. La définition de la matrice de transition locale $\breve{\mathcal{P}}^{(i)}$ de l'automate complété est présentée dans la troisième règle de la définition 7.2.11. Aux éléments diagonaux sont ajoutés les tuples de transition locale des événements complémentaires. Un tuple de transition

locale d'un événement complémentaire $\bar{e}$ est ajouté à $\breve{\mathcal{P}}^{(i)}(x^{(i)}, x^{(i)})$ uniquement s'il existe une transition d'un $x^{(i)}$ à un état $y^{(i)}$ (où $y^{(i)}$ différent de l'état fantôme Φ) qui peut être déclenchée par un tuple de transition locale avec un identificateur de l'événement e. De façon similaire, un tuple de transition locale d'un événement complémentaire $\bar{e}$ est ajouté à $\breve{\mathcal{P}}^{(i)}(x^{(i)}, \Phi)$, pour chaque tuple de transition locale associé à $\mathcal{P}^{(i)}(x^{(i)}, \Phi)$. Les tuples de transition locale des événements complémentaires associés à des tuples de transition locale qui amène à l'état fantôme Φ sont associés à la même transition, *i.e.*, à la transition vers l'état fantôme Φ. Cette représentation rend l'événement complémentaire possible sans que soit réalisable à partir de $x^{(i)}$.

Prenons par exemple l'automate $\mathcal{A}^{(1)}$ de FIG. 7.4 (page 120). À partir de l'état $1^{(1)}$, les événements e_1 et e_3 sont possibles, cependant uniquement l'événement e_3 est réalisable à partir de cet état. L'événement e_1 sera réalisable uniquement si l'occurrence de l'événement e_3 a amené à l'état $0^{(1)}$. Pour garder cette sémantique, l'événement complémentaire $\bar{e}_1$ peut avoir lieu uniquement si l'événement e_3 (plus prioritaire) a eu lieu d'abord et a amené à l'état $0^{(1)}$. Par conséquence, l'événement complémentaire $\bar{e}_1$ doit être associé à la transition de l'état $1^{(1)}$ vers l'état fantôme Φ. De cette façon, l'événement complémentaire $\bar{e}_1$ est possible à partir de l'état $1^{(1)}$ mais n'est pas réalisable.

Dans FIG. 7.6, on montre la représentation graphique de l'automate complété $\breve{\mathcal{P}}^{(1)}$ de l'automate $\mathcal{P}^{(1)}$ présenté dans FIG. 7.4 (page 120).



Événement	Prob.	Priorité
e_1	ρ_1	3
e_2	ρ_2	1
e_3	ρ_3	2
e_5	ρ_5	5

Événement	Prob.	Priorité
$\bar{e}_1$	$1-\rho_1$	3
$\bar{e}_2$	$1-\rho_2$	1
$\bar{e}_3$	$1-\rho_3$	2
$\bar{e}_5$	$1-\rho_5$	5

FIG. 7.6 – Représentation graphique de l'automate complété $\breve{\mathcal{A}}^{(1)}$ de l'automate $\mathcal{A}^{(1)}$ de FIG. 7.4

La matrice de transition locale $\breve{\mathcal{P}}^{(1)}$ de l'automate complété $\breve{\mathcal{A}}^{(1)}$ de FIG. 7.6 est :

$\breve{\mathcal{P}}^{(1)}$	$0^{(1)}$	$1^{(1)}$	$2^{(1)}$	Φ
$0^{(1)}$	$\{(\bar{e}_1, 1.0)\}$	$\{(e_1, 1.0)\}$	$\emptyset$	$\emptyset$
$1^{(1)}$	$\{(e_3, \pi_1)\}$	$\{(\bar{e}_3, 1.0)\}$	$\{(e_3, \pi_2)\}$	$\{(e_1, 1.0); (\bar{e}_1, 1.0)\}$
$2^{(1)}$	$\{(e_3, 1.0); (e_5, 1.0)\}$	$\{(e_2, 1.0)\}$	$\{(\bar{e}_2, 1.0); (\bar{e}_3, 1.0); (\bar{e}_5, 1.0)\}$	$\emptyset$

TAB. 7.4 – Matrice de transition locale $\breve{\mathcal{P}}^{(1)}$ de l'automate complété $\breve{\mathcal{A}}^{(1)}$ de FIG. 7.6

À partir de la matrice de transition locale de l'automate complété (Définition 7.2.11) et de l'ensemble d'événements possibles à partir de chaque état, on peut construire la chaîne de Markov représentée par l'automate complété.

Pour un seul automate ($i = 1$, réseau réduit à un automate), les états "vrai" de l'automate représentent les états de la chaîne de Markov[3] tandis que les probabilités de transition sont calculées par un calcul probabiliste des occurrences des événements possibles, sachant que l'hypothèse de base est que tous ces événements ont des lois de probabilités de Bernoulli représentant des variables aléatoires indépendantes (Définition 7.2.3, page 118). La définition de la chaîne de Markov pour un réseau d'automates est détaillée dans la section 7.3. Cependant, les principes de base présentés informellement ci-après, pour la construction de la chaîne de Markov d'un seul automate, sont les mêmes que pour un réseau d'automates.

Le calcul de la probabilité d'occurrence de chaque transition de la chaîne de Markov est fait de la façon suivante :

- Pour chaque état "vrai" de l'automate ($x^{(i)} \in \mathcal{S}^{(i)}$), on définit l'ensemble d'événements possibles ($pot(x^{(i)})$) ;
- Pour chaque événement e de $pot(x^{(i)})$, il existe dans $\mathcal{P}^{(i)}$ des tuples de transition locale. Ces tuples de transition locale peuvent être réécrit sous forme de *chaînons de transition globale* (voir définition formelle dans définition 7.3.1, page 132) de la forme suivante : si $\mathcal{P}^{(i)}(x^{(i)}, y^{(i)})$ contient le tuple $(e, \pi_e(x^{(i)}, y^{(i)}))$ alors $(x^{(i)}, y^{(i)}, e, \pi_e(x^{(i)}, y^{(i)}))$ est appelé *chaînon* de $\mathcal{A}^{(i)}$.
- La sémantique que l'on donne au comportement markovien de l'automate est la suivante : pour que les événements $\{e_1, e_2, \ldots, e_C\} \in pot(x^{(i)})$ avec $\varrho_1 < \varrho_2 < \ldots < \varrho_C$ soient réalisables en **1 unité de temps** il faut qu'il existe une suite de chaînons tel que :
  - Chaque événement n'apparaît qu'une fois ;
  - L'état d'arrivée d'un chaînon est égal à l'état de départ du chaînon suivant. Il faut que les chaînons d'une liste de chaînons s'enchaînent de façon à forme une chaîne. Ces listes de chaînons de transition globale sont appelées *chaînes de transitions globales*. Les règles qui définissent une chaîne de transitions globales sont définies formellement dans la définition 7.3.2 (page 132).

On va maintenant illustrer la construction de la chaîne de Markov de l'automate complété $\breve{\mathcal{A}}^{(1)}$ de FIG. 7.6.

Prenons par exemple l'état $0^{(1)}$. À partir de cet état, deux événements sont possibles ($pot(0^{(1)}) = \{e_1; \bar{e}_1\}$). Pour cet ensemble d'événements, on peut construire les chaînons de transition globale à partir de tuples de transition locale de l'automate. TAB. 7.5 présente les tuples de transition locale et les chaînons obtenus à partir de chaque tuple de transition locale.

Événement	Tuple de trans. locale	Chaînon de trans. globale	Événement	Tuple de trans. locale	Chaînon de trans. globale
e_1	$(e_1, 1.0)$	$(0^{(1)}, 1^{(1)}, e_1, 1.0)$	$\bar{e}_1$	$(\bar{e}_1, 1.0)$	$(0^{(1)}, 0^{(1)}, \bar{e}_1, 1.0)$

TAB. 7.5 – Ensemble de chaînons de transition globale pour l'ensemble d'événements de $pot(0^{(1)})$

À partir de ces chaînons, on peut construire deux chaînes de transitions globales : $\{(0^{(1)}, 1^{(1)}, e_1, 1.0)\}$ et $\{(0^{(1)}, 0^{(1)}, \bar{e}_1, 1.0)\}$. La chaîne de transitions globales $\{(0^{(1)}, 1^{(1)}, e_1, 1.0)\}$ contient un seul chaînon qui représente la transition de l'état $0^{(1)}$ vers l'état $1^{(1)}$ déclenchée par l'événement e_1. La probabilité d'occurrence de transition de l'état $0^{(1)}$ vers l'état $1^{(1)}$ de la chaîne de Markov est simplement la probabilité d'occurrence (ρ_1) de l'événement e_1. La chaîne de transitions globales $\{(0^{(1)}, 0^{(1)}, \bar{e}_1, 1.0)\}$ représente une transition de l'état $0^{(1)}$ vers lui-même. Cette chaîne aussi contient un seul chaînon et la

[3] Pour un réseau d'automates, les états de la chaîne de Markov est le produit cartésien des espaces d'états de chaque automate du modèle.

probabilité de transition de la chaîne de Markov est égale à $1 - \rho_1$, *i.e.*, la probabilité d'occurrence de l'événement $\bar{e}_1$. FIG. 7.7 présente les chemins de construction de ces deux chaînes.

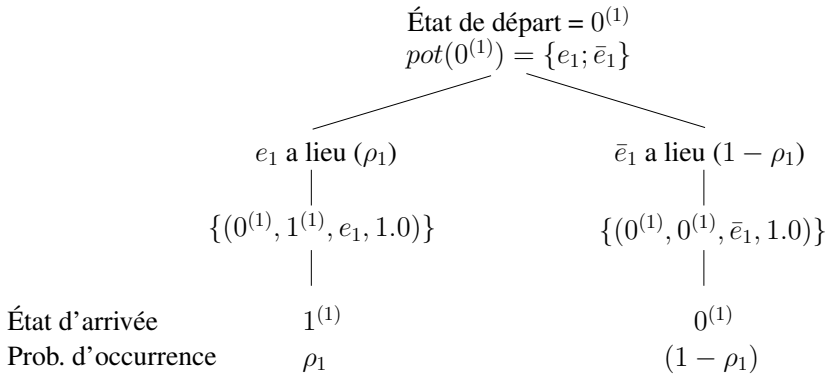


FIG. 7.7 – Construction des chaînes de transitions globales pour l'état $0^{(1)}$ de l'automate $\breve{\mathcal{A}}^{(1)}$ de FIG. 7.6

Prenons maintenant l'état $1^{(1)}$ de l'automate complété $\breve{\mathcal{A}}^{(1)}$ de FIG. 7.6. À partir de cet état, quatre événements sont possibles ($pot(1^{(1)}) = \{e_3; \bar{e}_3; e_1; \bar{e}_1\}$). Pour cet ensemble d'événements, on a l'ensemble de chaînons suivant :

Événement	Tuple de trans. locale	Chaînon de trans. globale	Événement	Tuple de trans. locale	Chaînon de trans. globale
e_3	(e_3, π_1)	$(1^{(1)}, 0^{(1)}, e_3, \pi_1)$	$\bar{e}_3$	$(\bar{e}_3, 1.0)$	$(1^{(1)}, 1^{(1)}, \bar{e}_3, 1.0)$
	(e_3, π_2)	$(1^{(1)}, 2^{(1)}, e_3, \pi_2)$		$(\bar{e}_3, 1.0)$	$(2^{(1)}, 2^{(1)}, \bar{e}_3, 1.0)$
	$(e_3, 1.0)$	$(2^{(1)}, 0^{(1)}, e_3, 1.0)$			
e_1	$(e_1, 1.0)$	$(0^{(1)}, 1^{(1)}, e_1, 1.0)$	$\bar{e}_1$	$(\bar{e}_1, 1.0)$	$(0^{(1)}, 0^{(1)}, \bar{e}_1, 1.0)$

TAB. 7.6 – Ensemble de chaînons de transition globale pour l'ensemble d'événements de $pot(1^{(1)})$

Pour cet ensemble de chaînons, quatre chaînes de transitions globales sont possibles :

1. $\{(1^{(1)}, 2^{(1)}, e_3, \pi_2)\}$;
2. $\{(1^{(1)}, 0^{(1)}, e_3, \pi_1); (0^{(1)}, 0^{(1)}, \bar{e}_1, 1.0)\}$;
3. $\{(1^{(1)}, 0^{(1)}, e_3, \pi_1); (0^{(1)}, 1^{(1)}, e_1, 1.0)\}$;
4. $\{(1^{(1)}, 1^{(1)}, \bar{e}_3, 1.0)\}$.

Ces chaînes de transitions globales sont des listes de chaînons de transition globale construites de façon à ce que chaque chaînon de transition globale soit enchaîné au chaînon suivant. Dans FIG. 7.8, on présente les chemins de construction de ces quatre chaînes de transitions globales.

On va regarder en détail chaque chaîne de transitions globales de cet état.

On va commencer par deuxième chaîne de FIG. 7.8 : $\{(1^{(1)}, 0^{(1)}, e_3, \pi_1); (0^{(1)}, 0^{(1)}, \bar{e}_1, 1.0)\}$. Cette chaîne de transitions globales va de l'état $1^{(1)}$ vers l'état $0^{(1)}$ de la chaîne de Markov. La probabilité d'occurrence calculée en multipliant la probabilité d'occurrence (ρ_3) de l'événement e_3 du premier chaînon

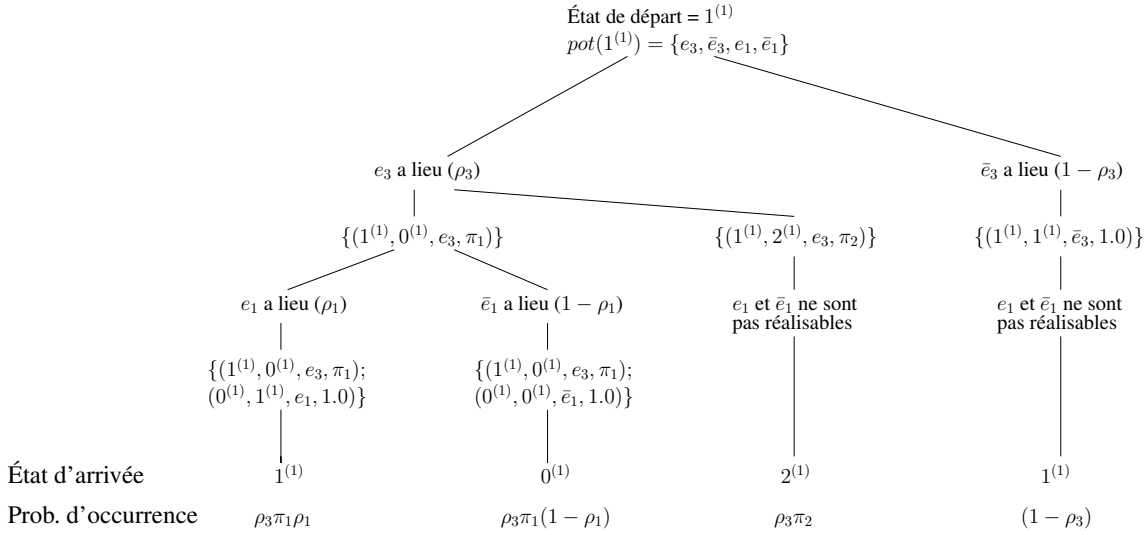


FIG. 7.8 – Construction des chaînes de transitions globales pour l'état $1^{(1)}$ de l'automate $\breve{\mathcal{A}}^{(1)}$ de FIG. 7.6

de la chaîne par la probabilité de routage du chaînon (π_1) et par la probabilité d'occurrence ($1-\rho_1$) de l'événement $\bar{e}_1$ du deuxième chaînon de la chaîne. La transition de l'état $1^{(1)}$ vers l'état $0^{(1)}$ de la chaîne de Markov a comme probabilité $\rho_3\pi_1(1-\rho_1)$. On remarque que pour cette chaîne de transitions globales, on a une séquence d'événements qui sont réalisables. La probabilité d'occurrence de la transition de la chaîne de Markov doit prendre en compte la probabilité d'occurrence de tous les événements de la chaîne de transitions globales.

La chaîne de transitions globales $\{(1^{(1)}, 2^{(1)}, e_3, \pi_2)\}$, troisième chaîne dans FIG. 7.8 contient un seul chaînon pour une transition de l'état $1^{(1)}$ vers l'état $2^{(1)}$ de la chaîne de Markov. La probabilité d'occurrence de cette transition est calculée en multipliant la probabilité d'occurrence (ρ_3) de l'événement e_3 qui déclenche la transition par probabilité de routage π_2 du chaînon. La probabilité d'occurrence est égale à $\rho_3\pi_2$.

Prenons maintenant la première et la dernière chaîne de transitions globales. On remarque que ces deux chaînes arrivent au même état. Cela représente deux possibilités pour une seule transition, et donc, la probabilité d'occurrence de cette transition dans la chaîne de Markov est la somme des probabilités d'occurrence de chaque chaîne de transitions globales. Calculons maintenant la probabilité d'occurrence de chaque chaîne de transitions globales. La probabilité d'occurrence de la chaîne de transitions globales $\{(1^{(1)}, 0^{(1)}, e_3, \pi_1); (0^{(1)}, 1^{(1)}, e_1, 1.0)\}$ est calculée en multipliant la probabilité d'occurrence ρ_3 de l'événement e_3 par la probabilité de routage π_1 du premier chaînon et par la probabilité d'occurrence ρ_1 de l'événement e_1 du deuxième chaînon. La deuxième chaîne de transitions globales $\{(1^{(1)}, 1^{(1)}, \bar{e}_3, 1.0)\}$ est composée d'un seul chaînon, donc la probabilité de transition est simplement la probabilité d'occurrence $(1-\rho_3)$ de l'événement $\bar{e}_3$. La probabilité d'occurrence de l'état $1^{(1)}$ vers lui-même de la chaîne de Markov est $\rho_3\pi_1\rho_1 + (1-\rho_3)$.

À partir de l'état $2^{(1)}$, six événements sont possibles ($pot(2^{(1)}) = \{e_2; \bar{e}_2; e_3; \bar{e}_3; e_5; \bar{e}_5\}$). Pour cet ensemble d'événements, on peut construire les chaînons de transitions globales à partir de tuples de transition locale de l'automate. Dans TAB. 7.7, on présente les tuples de transition locale et les chaînons obtenus à partir de chaque tuple de transition locale.

Événement	Tuple de trans. locale	Chaînon de trans. globale	Événement	Tuple de trans. locale	Chaînon de trans. globale
e_2	$(e_2, 1.0)$	$(2^{(1)}, 1^{(1)}, e_2, 1.0)$	$\bar{e}_2$	$(\bar{e}_2, 1.0)$	$(2^{(1)}, 2^{(1)}, \bar{e}_2, 1.0)$
e_3	(e_3, π_1)	$(1^{(1)}, 0^{(1)}, e_3, \pi_1)$	$\bar{e}_3$	$(\bar{e}_3, 1.0)$	$(1^{(1)}, 1^{(1)}, \bar{e}_3, 1.0)$
	(e_3, π_2)	$(1^{(1)}, 2^{(1)}, e_3, \pi_2)$		$(\bar{e}_3, 1.0)$	$(2^{(1)}, 2^{(1)}, \bar{e}_3, 1.0)$
	$(e_3, 1.0)$	$(2^{(1)}, 0^{(1)}, e_3, 1.0)$			
e_5	$(e_5, 1.0)$	$(2^{(1)}, 0^{(1)}, e_5, 1.0)$	$\bar{e}_5$	$(\bar{e}_5, 1.0)$	$(2^{(1)}, 2^{(1)}, \bar{e}_5, 1.0)$

TAB. 7.7 – Ensemble de chaînons de transition globale pour l'ensemble d'événements de $pot(2^{(1)})$

À partir de cet ensemble de chaînons, on peut construire sept chaînes de transitions globales sortant de l'état $2^{(1)}$. La construction de ces chaînes de transitions globales est présentée dans FIG. 7.9.

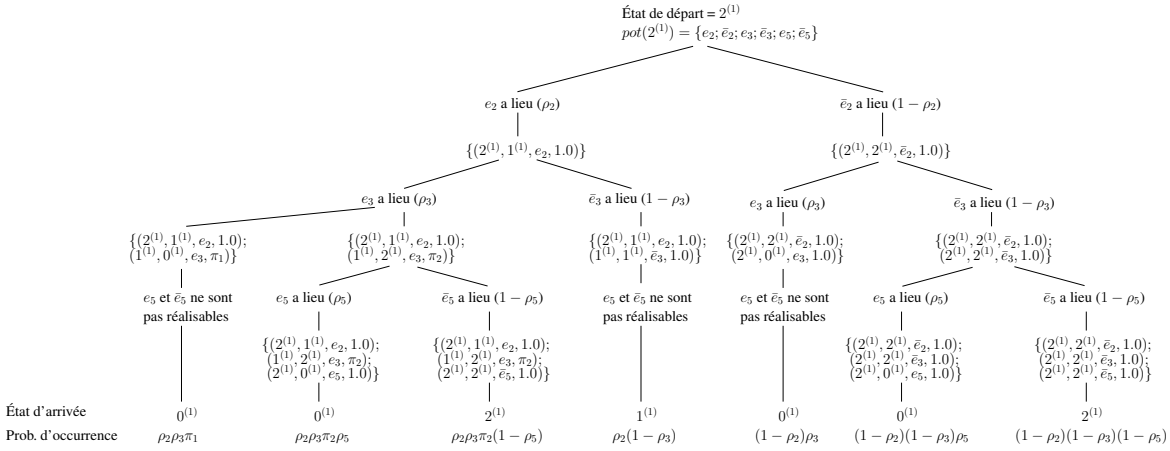


FIG. 7.9 – Construction des chaînes de transitions globales pour l'état $2^{(1)}$ de l'automate $\breve{\mathcal{A}}^{(1)}$ de FIG. 7.6

Le calcul des probabilités de transition pour l'état $2^{(1)}$ suit la même procédure. La définition formelle de la chaîne de Markov représentée par le modèle SAN sera présentée dans la section 7.4 (page 135).

Dans FIG. 7.10, on présente la chaîne de Markov représentée par l'automate complété $\breve{\mathcal{A}}^{(1)}$ de FIG. 7.6.

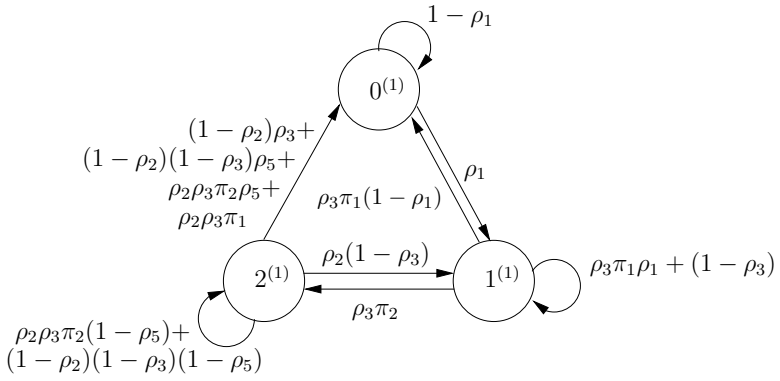
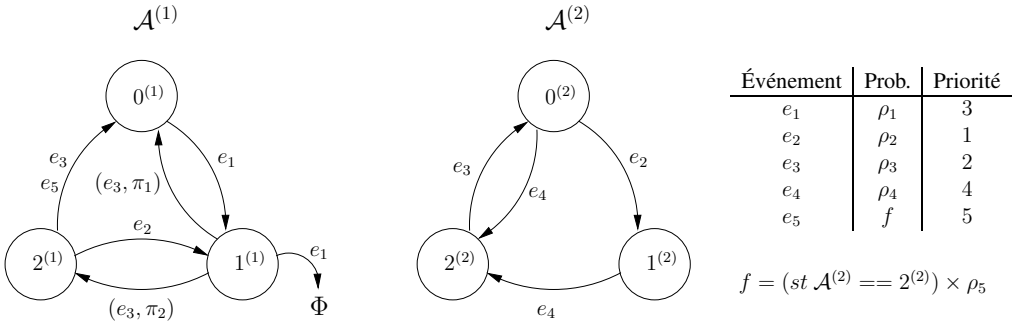


FIG. 7.10 – Chaîne de Markov représentée par l'automate $\mathcal{A}^{(1)}$

7.2.2 Définitions et notations de base pour un réseau d'automates

Soit un ensemble d'automates $\mathcal{A}^{(i)}$, où $i \in [1..N]$, selon la définition 7.2.6 (page 119). On va utiliser le modèle SAN décrit par FIG. 7.11 pour illustrer les définitions qui suivent.



Événement	Prob.	Priorité
e_1	ρ_1	3
e_2	ρ_2	1
e_3	ρ_3	2
e_4	ρ_4	4
e_5	f	5

FIG. 7.11 – Dessin d'un réseau d'automates

Définition 7.2.12. *Un réseau d'automates stochastiques (où modèle SAN) composé de N automates et E événements est défini par :*

1. *un ensemble d'événements $e \in \mathcal{E}$ commun à tous les automates, chaque événement avec son tuple d'événement (e, ρ_e, ϱ_e) et $|\mathcal{E}| = E$;*
2. *chacun des automates $\mathcal{A}^{(i)}$ ($i \in [1..N]$), son espace d'état $\breve{\mathcal{S}}^{(i)}$ et sa matrice de transition locale $\mathcal{P}^{(i)}$;*
3. *l'espace d'état produit $\hat{\mathcal{S}} = \prod \mathcal{S}^{(i)}$;*
4. *la fonction d'atteignabilité $\mathcal{F}$, qui définit l'ensemble d'états atteignables du modèle SAN dans $\mathcal{S}$ (voir Définition 7.2.13).*

Définition 7.2.13. *La fonction d'atteignabilité $\mathcal{F}$ est une fonction définie de $\hat{\mathcal{S}} \rightarrow [0..1]$. La fonction associe aux états globaux $\tilde{x} \in \hat{\mathcal{S}}$ la valeur 1 s'ils sont* **atteignables** *et la valeur 0 s'ils sont* **non-atteignables**.

Définition 7.2.14. *L'espace d'états atteignables $\mathcal{S}$ est le sous-ensemble de $\hat{\mathcal{S}}$ ($\mathcal{S} \subseteq \hat{\mathcal{S}}$) composé de tous les états globaux $\tilde{x} \in \hat{\mathcal{S}}$ tels que $\mathcal{F}(\tilde{x}) = 1$.*

La fonction d'atteignabilité $\mathcal{F}$ s'évalue pour tous les états globaux $\tilde{x} \in \hat{\mathcal{S}}$ d'un modèle SAN et ainsi détermine quels sont les états atteignables de ce modèle.

✍ **Soit** : Étant donné $e \in \mathcal{E}$,

$\mathcal{O}_e$ l'ensemble d'indices i ($i \in [1..N]$) tel que la matrice de transition locale $\mathcal{P}^{(i)}$ contienne au moins un tuple de transition locale avec l'identificateur de l'événement e.

Définition 7.2.15. *Un événement $e \in \mathcal{E}$ est appelé :*

1. *événement local, si $\mid \mathcal{O}_e \mid = 1$;*
2. *événement synchronisant, si $\mid \mathcal{O}_e \mid > 1$.*

La définition 7.2.15 classe chaque événement qui peut être un *événement local* ou un *événement synchronisant*. Par définition un événement synchronisant *est réalisable uniquement s'il est réalisable au même instant dans tous les automates concernés par l'événement*. Le déclenchement d'un événement synchronisant change, de façon synchronisée, l'état local de tous les automates concernés par l'événement au même instant.

Pour le modèle SAN décrit par FIG. 7.11, on peut classer les événements ainsi :

Événement	Type	$\mathcal{O}_e$
e_1	local	$\{1\}$
e_2	synchronisant	$\{1;2\}$
e_3	synchronisant	$\{1;2\}$
e_4	local	$\{2\}$
e_5	local	$\{1\}$

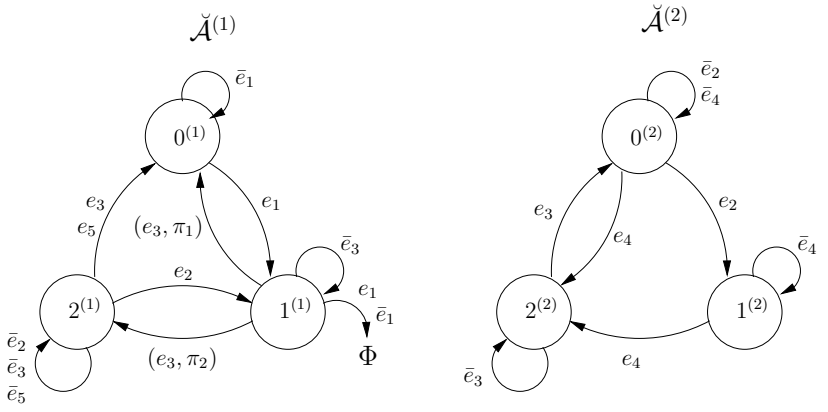
TAB. 7.8 – Classement des événements du modèle SAN de FIG. 7.11

7.2.2.1 Réseau d'automates complétés

Définition 7.2.16. *Un réseau d'automates stochastiques complétés composé de N automates complétés et $2E$ événements est défini par :*

1. *un ensemble d'événements $e \in \breve{\mathcal{E}}$ commun à tous les automates, chacun avec son tuple d'événement (e, ρ_e, ϱ_e), où $|\breve{\mathcal{E}}| = 2E$;*
2. *chacun des automates complétés $\breve{\mathcal{A}}^{(i)}$ ($i \in [1..N]$), son espace d'état $\breve{\mathcal{S}}^{(i)}$ et sa matrice de transition locale $\breve{\mathcal{P}}^{(i)}$;*
3. *la fonction d'atteignabilité $\mathcal{F}$, qui définit l'ensemble des états atteignables du SAN dans $\hat{\mathcal{S}}$.*

Dans FIG. 7.12, on présente le réseau d'automates complétés déduit du réseau d'automates présenté dans FIG. 7.11.



Événement	Prob.	Priorité
e_1	ρ_1	3
e_2	ρ_2	1
e_3	ρ_3	2
e_4	ρ_4	4
e_5	f	5

Événement	Prob.	Priorité
$\bar{e}_1$	$1-\rho_1$	3
$\bar{e}_2$	$1-\rho_2$	1
$\bar{e}_3$	$1-\rho_3$	2
$\bar{e}_4$	$1-\rho_4$	4
$\bar{e}_5$	$1-f$	5

$$f = (st\ \mathcal{A}^{(2)} == 2^{(2)}) \times \rho_5$$

FIG. 7.12 – Représentation graphique d'un réseau d'automates complétés du modèle SAN de FIG. 7.11

7.2.3 SAN *bien définis*

Quelques restrictions doivent être faites pour assurer le cadre Markovien d'un modèle SAN. Les modèles SAN respectant ces restrictions sont appelés SAN *bien définis*.

✍ **Notation**

$\pi_e(x^{(i)}, y^{(i)})(\tilde{x})$ la probabilité de routage associée au tuple de transition locale $(e, \pi_e(x^{(i)}, y^{(i)}))$ dans $\mathcal{P}^{(i)}(x^{(i)}, y^{(i)})$ évaluée pour l'état global $\tilde{x}$.

Restriction 5 *Un automate $\mathcal{A}^{(i)}$ est bien défini, si et seulement si pour tout $\tilde{x} \in \hat{\mathcal{S}}$, pour tout $x^{(i)} \in \mathcal{S}^{(i)}$, pour tout $e \in \mathcal{E}$ tel que $succ_e(x^{(i)})$ n'est pas vide et pour $y^{(i)} \in succ_e(x^{(i)})$:*

$$5.1. \qquad \left(\sum_{y^{(i)} \in succ_e(x^{(i)})} \pi_e(x^{(i)}, y^{(i)})(\tilde{x}) \right) = 1$$

La restriction 5.1 impose que la somme des probabilités de routage de toutes les transitions en sortant d'un même état doit être égale à 1.

Restriction 6 *Un événement $e \in \mathcal{E}$ est bien défini, si et seulement si :*

6.1. $\forall x^{(i)}, \forall y^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $y^{(i)} \in succ_e(x^{(i)})$ *et* $(e, \pi_e(x^{(i)}, y^{(i)})) \in \mathcal{P}^{(i)}(x^{(i)}, y^{(i)})$
si $\exists (e_1, \pi_{e_1}(x^{(i)}, y^{(i)})) \in \mathcal{P}^{(i)}(x^{(i)}, y^{(i)})$ *alors* e_1 *est un événement différent de* e.

6.2. $\forall x^{(i)}, \forall y^{(i)} \in \mathcal{S}^{(i)}$ *tel que* $y^{(i)} \in succ_e(x^{(i)})$ *et* $(e, \pi_e(x^{(i)}, y^{(i)})) \in \mathcal{P}^{(i)}(x^{(i)}, y^{(i)})$
si $\exists (e_1, \pi_{e_1}(x^{(i)}, y^{(i)})) \in \mathcal{P}^{(i)}(x^{(i)}, \Phi)$ *alors* e_1 *est un événement différent de* e.

La restriction 6.1 imposée aux événements exige que l'identificateur d'un événement apparaisse *une seule fois* dans le sous-ensemble de tuples de transition locale d'un état donné vers un autre. La restriction 6.2 impose qu'il ne doit pas exister un tuple de transition locale $(e, \pi_e(x^{(i)}, y^{(i)}))$ à partir d'un état donné $x^{(i)} \in \mathcal{S}^{(i)}$ vers l'état fantôme Φ, s'il existe déjà un tuple de transition locale de $x^{(i)}$ vers un état $y^{(i)} \in \mathcal{S}^{(i)}$ avec le même événement e.

Restriction 7 *Un modèle SAN est bien défini, si et seulement si :*

7.1. tous ses automates sont bien définis ;

7.2. tous ses événements sont bien définis.

7.3 L'automate global

L'objectif d'un modèle SAN est de décrire de façon modulaire un automate unique (dit *automate global*) qui décrit le comportement de l'ensemble du réseau d'automates. Dans cette section nous procédons à cette définition.

Dans cette section, on va considérer un modèle SAN tel que défini dans la définition 7.2.16 (page 130).

7.3.1 Transition globale

Définition 7.3.1. *Étant donné un SAN complété* $(\breve{\mathcal{E}}, \breve{\mathcal{A}}^{(i)}, \hat{\mathcal{S}}, \mathcal{F})$*, où* $i \in [1..N]$*, on associe pour tout* $\tilde{x} \in \hat{\mathcal{S}}$*, pour tout* $\tilde{y} \in \hat{\mathcal{S}}$ *et pour tout* $e \in \breve{\mathcal{E}}$ *tel que pour tout* $i \in \mathcal{O}_e$ *il existe* $(e, \pi_e(x^{(i)}, y^{(i)})) \in \breve{\mathcal{P}}(x^{(i)}, y^{(i)})$*, un* **chaînon de transition globale** $(\tilde{x}, \tilde{y}, e, \Pi_e(\tilde{x}, \tilde{y}))$ *où :*

1. $\tilde{x}$*, l'état global de départ ;*
2. $\tilde{y}$*, l'état global d'arrivée ;*
3. e*, l'identificateur d'un événement qui déclenche la transition ;*
4. $\Pi_e(\tilde{x}, \tilde{y})$*, la probabilité de routage globale définie par :*

$$\Pi_e(\tilde{x}, \tilde{y}) = \prod_{i \in \mathcal{O}_e, \exists (e, \pi_e(x^{(i)}, y^{(i)})) \in \mathcal{P}(x^{(i)}, y^{(i)})} \pi_e(x^{(i)}, y^{(i)}).$$

Un chaînon de transition globale représente une transition de l'état global $\tilde{x}$ vers l'état global $\tilde{y}$ par le déclenchement uniquement de l'événement e avec une probabilité de routage globale $\Pi_e(\tilde{x}, \tilde{y})$ calculée par le produit des probabilités de routage de tous les automates concernés par l'événement e vers l'état d'arrivée $y^{(i)}$ quand $i \in \mathcal{O}_e$.

7.3.2 Chaîne de transitions globales

✍ **Notation**

$\tilde{\mathcal{T}}_e$	l'ensemble de chaînons de transition globale déclenchés par l'événement e ;
ϵ	un sous-ensemble d'événements de $\breve{\mathcal{E}}$;
$\tilde{\mathcal{T}}_\epsilon$	l'union des $\tilde{\mathcal{T}}_e$, où $e \in \epsilon$ ($\tilde{\mathcal{T}}_\epsilon = \bigcup_{e \in \epsilon} \tilde{\mathcal{T}}_e$).

Définition 7.3.2. *Soit un ensemble d'évènements* ϵ*, on appelle* **chaîne de transitions globales***, une liste ordonnée* $\{(\tilde{x}_1, \tilde{y}_1, e_1, \Pi_{e_1}(\tilde{x}_1, \tilde{y}_1)), \ldots, (\tilde{x}_C, \tilde{y}_C, e_C, \Pi_{e_C}(\tilde{x}_C, \tilde{y}_C))\}$ *composée des* C *chaînons de transition globale de* $\tilde{\mathcal{T}}_\epsilon$*, qui respecte les règles suivantes :*

1. $\varrho_{e_1} < \varrho_{e_2} < \ldots < \varrho_{e_C}$
2. $\forall i \in [2..C]$
   $\tilde{x}_i = \tilde{y}_{i-1}$
3. $\nexists (\tilde{x}, \tilde{y}, e, \Pi_e(\tilde{x}, \tilde{y})) \in \tilde{\mathcal{T}}_\epsilon$ *tel que*
   $\varrho_e < \varrho_{e_1}$ *et* $\tilde{y} = \tilde{x}_1$
4. $\nexists (\tilde{x}, \tilde{y}, e, \Pi_e(\tilde{x}, \tilde{y})) \in \tilde{\mathcal{T}}_\epsilon$ *tel que*
   $\varrho_{e_C} < \varrho_e$ *et* $\tilde{x} = \tilde{y}_C$
5. $\forall i \in [1..C-1], \nexists (\tilde{x}, \tilde{y}, e, \Pi_e(\tilde{x}, \tilde{y})) \in \tilde{\mathcal{T}}_\epsilon$ *tel que*
   $\varrho_{e_i} < \varrho_e < \varrho_{e_{i+1}}$ *et* $\tilde{y}_i = \tilde{x}$

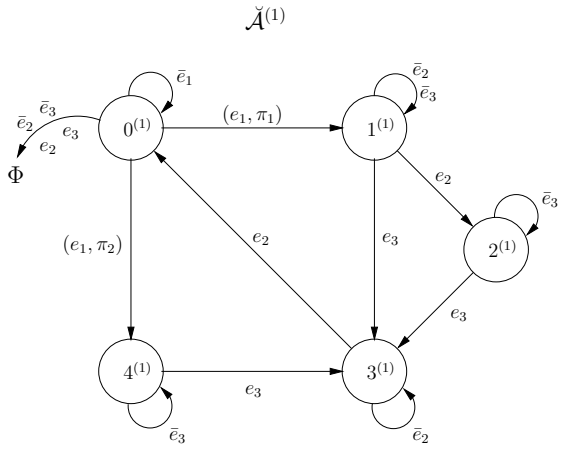
La définition 7.3.2 établit un ensemble de règles pour qu'une liste de chaînons de transition globale soit considérée une *chaîne de transitions globales*. La règle 1 définit l'ordre des chaînons de transition globale dans la chaîne, respectant l'ordre de priorité des événements dans la chaîne. La règle 2 garantit

l'enchaînement de tous les chaînons. Cette règle assure que l'état d'arrivée $\tilde{y}_i$ d'un chaînon d'indice i sera l'état de départ $\tilde{x}_{i+1}$ du chaînon suivant (d'indice $i+1$). **On se restreint uniquement aux chaînes les plus longues qui respectent les règles 1 et 2**. Autrement dit, aucun autre chaînon de transition globale ne peut être enchaîné, soit au début, soit au mieux, soit à la fin de la chaîne. Les 3 dernières règles assurent cette condition. La règle 3 garantit qu'aucun chaînon de transition globale ne peut être enchaîné au début de la chaîne. La règle 4 assure qu'aucun chaînon de transition globale ne peut être ajouté à la fin de la chaîne. La règle 5 garantit qu'aucun chaînon de transition globale ne peut être enchaîné entre deux chaînons qui se suivent.

✍ **Notation**

$\tilde{x}_{i_g}$	l'état de départ du i-ème chaînon de transition globale de la chaîne de transitions globales g ;
$\tilde{y}_{i_g}$	l'état d'arrivée du i-ème chaînon de transition globale de la chaîne de transitions globales g ;
$\mathcal{L}_\epsilon$	l'ensemble des chaînes de transitions globales de l'ensemble d'événements ϵ ;
$\mathcal{L}(\tilde{x}, \tilde{y})$	l'ensemble de chaînes de transitions globales $g \in \mathcal{L}_{pot(\tilde{x})}$ tel que $\tilde{x}_{1_g} = \tilde{x}$ et $\tilde{y}_{C_g} = \tilde{y}$ où C_g est le nombre de chaînons de transition globale de la chaîne de transitions globales g ;
$\mathcal{L}$	l'ensemble des chaînes de transitions globales du modèle SAN.

Pour illustrer les chaînes de transitions globales qui nous intéressent, on va considérer l'automate complété $\breve{\mathcal{A}}^{(1)}$ de FIG. 7.13 et un ensemble d'événements $\epsilon = \{e_1; \bar{e}_1; e_2; \bar{e}_2; e_3; \bar{e}_3\}$. Cet ensemble d'événements correspond au $\epsilon = pot(0^{(1)}) = \{e_1; \bar{e}_1; e_2; \bar{e}_2; e_3; \bar{e}_3\}$ (Définition 7.3.3, page 135).



Événement	Prob.	Priorité
e_1	ρ_1	1
e_2	ρ_2	2
e_3	ρ_3	3

Événement	Prob.	Priorité
$\bar{e}_1$	$1-\rho_1$	1
$\bar{e}_2$	$1-\rho_2$	2
$\bar{e}_3$	$1-\rho_3$	3

FIG. 7.13 – Exemple d'un automate complété $\breve{\mathcal{A}}^{(1)}$

Pour l'ensemble d'événement ϵ et l'automate complété $\breve{\mathcal{A}}^{(1)}$ de FIG. 7.13, dans TAB. 7.9, on présente l'ensemble des chaînons de transition globale $\tilde{\mathcal{T}}_\epsilon$ du modèle. Pour simplifier la lecture, on va représenter, par exemple, un chaînon $(0^{(1)}, 1^{(1)}, e_1, \pi_1)$ par $(0, 1, e_1, \pi_1)$.

Étant donné le grand nombre de chaînes possibles à partir des chaînons de TAB. 7.9, on va présenter uniquement quelques chaînes qui expliquent les règles de la définition 7.3.2.

Événement	Chaînon de trans. globale	Événement	Chaînon de trans. globale
e_1	$(0,1,e_1,\pi_1)$ $(0,4,e_1,\pi_2)$	$\bar{e}_1$	$(0,0,\bar{e}_1,1)$
e_2	$(1,2,e_2,1)$ $(3,0,e_2,1)$	$\bar{e}_2$	$(1,1,\bar{e}_2,1)$ $(3,3,\bar{e}_2,1)$
e_3	$(1,3,e_3,1)$ $(2,3,e_3,1)$ $(4,3,e_3,1)$	$\bar{e}_3$	$(1,1,\bar{e}_3,1)$ $(2,2,\bar{e}_3,1)$ $(4,4,\bar{e}_3,1)$

TAB. 7.9 – Ensemble de chaînons de transition globale $\tilde{\mathcal{T}}_\epsilon$ de FIG. 7.13

Étant donné $\epsilon = pot(0^{(1)}) = \{e_1;\bar{e}_1;e_2;\bar{e}_2;e_3;\bar{e}_3\}$. On considère la liste de chaînons suivante $\{(0,1,e_1,\pi_1);(1,2,e_2,1);(2,3,e_3,1)\}$. Dans cette liste, on peut voir que les chaînons sont ordonnés selon ordre de priorité des événements et l'état d'arrivée 1 du chaînon $(0,1,e_1,\pi_1)$ est égal à l'état de départ du chaînon $(1,2,e_2,1)$ et son état d'arrivée est égal à l'état de départ du chaînon $(2,3,e_3,1)$ qui le suit. On voit aussi qu'aucun autre chaînon de transition globale ne peut être introduit à la liste, soit au début (il n'y a pas d'événement plus prioritaire que e_1, premier chaînon de la chaîne), soit au milieux (aucun chaînon de l'ensemble des chaînons possibles $\tilde{\mathcal{T}}_\epsilon$ présenté dans TAB. 7.9 ne peut être inséré entre deux chaînons de cette chaîne), soit à la fin (aucun événement est moins prioritaire que e_3, dernier chaînon de la chaîne). Dans ce cas, en respectant les règles de la définition 7.3.2, cette liste est une *chaîne de transitions globales*.

Considérons maintenant la liste de chaînons suivant $\{(0,1,e_1,\pi_1);(1,2,e_2,1)\}$. Cette liste est une sous-liste de la liste précédente, et, les chaînons sont ordonnés selon l'ordre de priorité des événements et les états de départ et d'arrivée des chaînons s'enchaînent. Cependant, il existe un chaînon $(2,3,e_3,1)$ dans $\tilde{\mathcal{T}}_\epsilon$ qui peut être encore ajouté à la fin de la liste, en effet, $\varrho_{e_2} < \varrho_{e_3}$ et l'état d'arrivée du dernier chaînon de la chaîne $(1,2,e_2,1)$ est égal à l'état de départ du chaînon $(2,3,e_3,1)$ de l'ensemble des chaînons $\tilde{\mathcal{T}}_\epsilon$ (TAB. 7.9). Cette liste n'est pas une chaîne de transitions globales, car elle ne respecte pas la règle 4 de la définition 7.3.2, qui définit qu'une liste des chaînons de transition globale est une chaîne de transitions globales uniquement s'il n'existe pas d'autres chaînons de transition globale qui puissent être ajouté à la fin de la liste.

On va considérer maintenant la liste de transitions globales $\{(0,1,e_1,\pi_1);(1,3,e_3,1)\}$. Cette liste respecte l'ordre de priorité des événements, l'enchaînement des chaînons et aucun autre chaînon de transition globale dans $\tilde{\mathcal{T}}_\epsilon$ (TAB. 7.9) ne peut être introduit au début (e_1 est l'événement le plus prioritaire) ou à la fin (e_3 est l'événement le moins prioritaire) de la liste. Cependant, il existe un chaînon $(1,1,\bar{e}_2,1) \in \tilde{\mathcal{T}}_\epsilon$, qui peut être introduit entre les deux chaînons de la liste : $\varrho_{e_1} < \varrho_{e_2} < \varrho_{e_3}$ et l'état d'arrivée du premier chaînon de la liste $(0,1,e_1,\pi_1)$ est égal à l'état de départ du chaînon $(1,1,\bar{e}_2,1)$ et l'état d'arrivée est égal à l'état de départ du dernier chaînon de la liste $(1,3,e_3,1)$. Autrement dit, cette liste ne respecte pas la règle 5 de la définition 7.3.2 et elle n'est pas une chaîne de transitions globales, car il existe *une chaîne plus longue* ($\{(0,1,e_1,\pi_1);(1,1,\bar{e}_2,1);(1,3,e_3,1)\}$) qui contient tous les chaînons de la liste $\{(0,1,e_1,\pi_1);(1,3,e_3,1)\}$.

Prenons un exemple de chaînes de transitions globales qui ne contiennent pas tous les événements possibles de l'état $0^{(1)}$ de l'automate complété $\breve{\mathcal{A}}^{(1)}$. Prenons la liste de transitions globales $\{(0,4,e_1,\pi_2);$

$(4, 3, e_3, 1)\}$. Cette liste est une chaîne de transitions globales. Les chaînons sont ordonnés par l'ordre de priorité de événements, l'état d'arrivée du chaînon $(0, 4, e_1, \pi_2)$ est égal à l'état de départ du chaînon $(4, 3, e_3, 1)$ et aucun autre chaînon ne peut être introduit, soit au début, soit au mieux, soit à la fin de la liste. Il faut remarquer que cette chaîne de transitions globales ne contient pas des chaînons avec l'événement e_2, ni son événement complémentaire $\bar{e}_2$, qui sont dans l'ensemble $\epsilon = \{e_1; \bar{e}_1; e_2; \bar{e}_2; e_3; \bar{e}_3\}$, *i.e.*, dans $pot(0^{(1)})$. Les événements e_2 et $\bar{e}_2$ ne sont pas présents dans cette chaîne car le premier chaînon de la chaîne $(0, 4, e_1, \pi_2)$ amène à l'état 4, où les événements e_2 et $\bar{e}_2$ ne sont pas réalisables.

La définition suivante nomme l'ensemble des *événements possibles* à partir d'un état global $\tilde{x}$. Pour que l'événement soit un événement possible globalement, *il doit être réalisable dans tous les automates concernés par l'événement*.

Définition 7.3.3. *Soit un état global $\tilde{x} \in \hat{\mathcal{S}}$, on définit l'ensemble d'événements $pot(\tilde{x})$ comme : $e \in pot(\tilde{x}) \Leftrightarrow \forall i \in \mathcal{O}_e, e \in pot(x^{(i)})$*

7.3.3 Automate global

Définition 7.3.4. *Soit un SAN complété $(\check{\mathcal{E}}, \check{\mathcal{A}}^{(i)}, \hat{\mathcal{S}}, \mathcal{F})$, où $i \in [1..N]$, l'automate global $\mathcal{A}$ est défini par :*

1. *l'ensemble d'états $\hat{\mathcal{S}}$;*
2. *l'ensemble d'événements $\check{\mathcal{E}}$;*
3. *la fonction d'atteignabilité $\mathcal{F}$;*
4. *la matrice de transition globale $\mathcal{G} : \hat{\mathcal{S}} \times \hat{\mathcal{S}} \to \mathcal{L}$:*

$$\forall \tilde{x} \in \hat{\mathcal{S}}, \forall \tilde{y} \in \hat{\mathcal{S}}$$
$$\mathcal{G}(\tilde{x}, \tilde{y}) = \bigcup_{\forall g \in \mathcal{L}(\tilde{x}, \tilde{y})} \{g\}$$

Contrairement au formalisme SAN en temps continu, ici un automate local n'est pas le même objet mathématique que l'automate global. En effet, les éléments de transition de la matrice de transition locale d'un automate contiennent un ensemble des tuples de transition locale (e, π) où l'occurrence de n'importe quel événement de l'élément de transition déclenche la transition. Dans l'automate global, les éléments de transition de la matrice de transition globale contiennent un ensemble de chaînes de transitions globales $\{(\tilde{x}_1, \tilde{y}_1, e_1, \Pi_{(\tilde{x}_1, \tilde{y}_1)}), \ldots, (\tilde{x}_C, \tilde{y}_C, e_C, \Pi_{(\tilde{x}_C, \tilde{y}_C)})\}$ où tous les événements associés aux chaînons de cette chaîne *doivent avoir lieu pour que la transition se réalise*.

7.4 La chaîne de Markov

La probabilité d'occurrence d'une chaîne de transitions globales est le produit de toutes les probabilités d'occurrence ρ_e des événements de la chaîne de transitions globales et des probabilités de routage globales $\Pi_e(\tilde{x}, \tilde{y})$.

✍ **Notation**

$\rho_e(\tilde{x})$	la probabilité d'occurrence de l'événement e évaluée pour l'état global $\tilde{x}$;
$(e, \Pi_e(\tilde{x}, \tilde{y}))$	le tuple de transition globale de l'événement e avec la probabilité de routage globale $\Pi_e(\tilde{x}, \tilde{y})$ de l'état global $\tilde{x}$ vers l'état global $\tilde{y}$;
$\Pi_e(\tilde{x}_e, \tilde{y}_e)(\tilde{x})$	la probabilité de routage globale $\Pi_e(\tilde{x}_e, \tilde{y}_e)$ du chaînon de transition globale $(\tilde{x}_e, \tilde{y}_e, e, \Pi_e(\tilde{x}_e, \tilde{y}_e))$ évaluée pour l'état global $\tilde{x}$. On remarque de l'état global $\tilde{x}$ est égal à l'état de départ du chaînon.

Définition 7.4.1. *Étant donné une chaîne de transitions globales* $g = \{(\tilde{x}_1, \tilde{y}_1, e_1, \Pi_{e_1}(\tilde{x}_1, \tilde{y}_1)), \ldots, (\tilde{x}_C, \tilde{y}_C, e_C, \Pi_{e_C}(\tilde{x}_C, \tilde{y}_C))\}$*, la probabilité d'occurrence, notée* Π_g*, de cette chaîne est :*

$$\Pi_g = \Big(\rho_{e_1}(\tilde{x}_1) \times \Pi_{e_1}(\tilde{x}_1, \tilde{y}_1)(\tilde{x}_1)\Big) \times \ldots \times \Big(\rho_{e_C}(\tilde{x}_C) \times \Pi_{e_C}(\tilde{x}_C, \tilde{y}_C)(\tilde{x}_C)\Big),$$

où "$\times$*" note le produit de réels.*

La chaîne de Markov représentée par un modèle SAN est définie sur les états atteignables $\mathcal{S}$ du modèle SAN. À partir de l'automate global, on peut aisément définir la chaîne de Markov représentée par le modèle SAN. La suite des états pris par l'automate global est un processus aléatoire. Dans la construction qu'on vient de faire, ce processus est tel que la probabilité d'atteindre l'état suivant ne dépend que de l'état courant et de la probabilité d'occurrence d'un ensemble d'événements simultanés qui sont ceux des chaînes qui sont associées aux transitions dans $\mathcal{G}$. On peut donc lui associer une matrice de transition de Markov de la façon suivante :

Définition 7.4.2. *Soit un modèle SAN dont l'automate global* $\mathcal{A}$ $(\breve{\mathcal{E}}, \hat{\mathcal{S}}, \mathcal{G}, \mathcal{F})$*, la matrice de transition de la chaîne de Markov modélisé par ce modèle SAN :*

1. $P : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ *:*

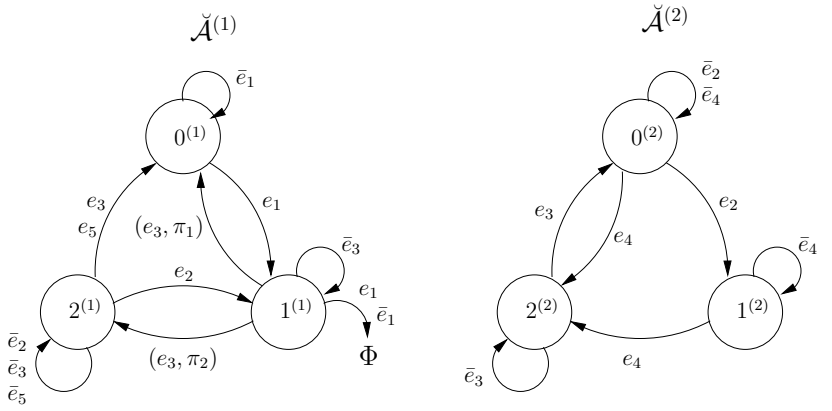
$$\forall \tilde{x} \in \mathcal{S}, \forall \tilde{y} \in \mathcal{S}$$
$$P(\tilde{x}, \tilde{y}) = \sum_{\forall g \in \mathcal{G}(\tilde{x}, \tilde{y})} \Pi_g$$

La probabilité de transition de l'état $\tilde{x}$ vers l'état $\tilde{y}$ de la chaîne de Markov est la somme des probabilités d'occurrence de chaque chaîne de transitions globales qui amène de l'état $\tilde{x}$ vers l'état $\tilde{y}$.

7.4.1 Exemple de construction de la chaîne de Markov

Pour clarifier la construction de la chaîne de Markov, on va présenter pour un exemple et pour chaque état global atteignable ($\tilde{x} \in \mathcal{S}$), l'ensemble d'événements possibles ($pot(\tilde{x})$) ainsi que l'ensemble de chaînons de transition globale pour chaque $pot(\tilde{x})$ et les chaînes de transitions globales sortant de chaque état.

Le modèle considéré est celui présenté dans FIG. 7.12 (page 130). On rappelle ce modèle.



Événement	Prob.	Priorité
e_1	ρ_1	3
e_2	ρ_2	1
e_3	ρ_3	2
e_4	ρ_4	4
e_5	f	5

Événement	Prob.	Priorité
$\bar{e}_1$	$1-\rho_1$	3
$\bar{e}_2$	$1-\rho_2$	1
$\bar{e}_3$	$1-\rho_3$	2
$\bar{e}_4$	$1-\rho_4$	4
$\bar{e}_5$	$1-f$	5

$f = (st\ \mathcal{A}^{(2)} == 2^{(2)}) \times \rho_5$

FIG. 7.14 – Réseau d'automates complétés

Pour l'état global $0^{(1)}0^{(2)}$, l'ensemble d'événements possibles est $pot(0^{(1)}0^{(2)}) = \{e_1, \bar{e}_1, e_4, \bar{e}_4\}$. L'ensemble de chaînons de transition globale pour cet ensemble $\tilde{\mathcal{T}}_{pot(0^{(1)}0^{(2)})}$ est présenté dans TAB. 7.10.

Événement	Chaînon de transition globale	Événement	Chaînon de transition globale
e_1	$(0^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, e_1, 1)$	$\bar{e}_1$	$(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_1, 1)$
	$(0^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, e_1, 1)$		$(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, \bar{e}_1, 1)$
	$(0^{(1)}2^{(2)}, 1^{(1)}2^{(2)}, e_1, 1)$		$(0^{(1)}2^{(2)}, 0^{(1)}2^{(2)}, \bar{e}_1, 1)$
e_4	$(0^{(1)}0^{(2)}, 0^{(1)}2^{(2)}, e_4, 1)$	$\bar{e}_4$	$(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_4, 1)$
	$(0^{(1)}1^{(2)}, 0^{(1)}2^{(2)}, e_4, 1)$		$(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, \bar{e}_4, 1)$
	$(1^{(1)}0^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)$		$(1^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, \bar{e}_4, 1)$
	$(1^{(1)}1^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)$		$(1^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, \bar{e}_4, 1)$
	$(2^{(1)}0^{(2)}, 2^{(1)}2^{(2)}, e_4, 1)$		$(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_4, 1)$
	$(2^{(1)}1^{(2)}, 2^{(1)}2^{(2)}, e_4, 1)$		$(2^{(1)}1^{(2)}, 2^{(1)}1^{(2)}, \bar{e}_4, 1)$

TAB. 7.10 – Ensemble de chaînons de transition globale $\tilde{\mathcal{T}}_{pot(0^{(1)}0^{(2)})}$ de FIG. 7.14

À partir de ces chaînons, on peut construire les chaînes de transitions globales (TAB. 7.11) pour l'état global $0^{(1)}0^{(2)}$ et calculer les probabilités d'occurrence des transitions.

État d'arrivée	Chaîne de transitions globales	Probabilité de transition
$0^{(1)}0^{(2)}$	$\{(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_1, 1); (0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_4, 1)\}$	$(1-\rho_1)(1-\rho_4)$
$0^{(1)}2^{(2)}$	$\{(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_1, 1); (0^{(1)}0^{(2)}, 0^{(1)}2^{(2)}, e_4, 1)\}$	$(1-\rho_1)\rho_4$
$1^{(1)}0^{(2)}$	$\{(0^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, e_1, 1); (1^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, \bar{e}_4, 1)\}$	$\rho_1(1-\rho_4)$
$1^{(1)}2^{(2)}$	$\{(0^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, e_1, 1); (1^{(1)}0^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)\}$	$\rho_1\rho_4$

TAB. 7.11 – Ensemble des chaînes de transitions globales de l'état global $0^{(1)}0^{(2)}$ de FIG. 7.14

Dans FIG. 7.15, on présente les chemins de construction des chaînes de transitions globales pour l'état $0^{(1)}0^{(2)}$.

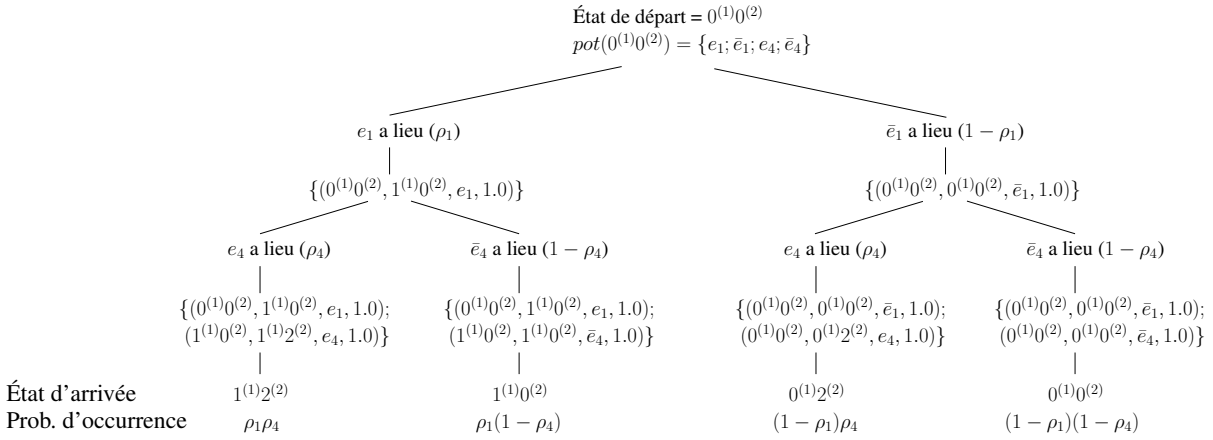


FIG. 7.15 – Construction des chaînes de transitions globales pour l'état $0^{(1)}0^{(2)}$

Pour l'état global $0^{(1)}1^{(2)}$, l'ensemble d'événements possibles est $pot(0^{(1)}1^{(2)}) = \{e_1, \bar{e}_1, e_4, \bar{e}_4\}$. L'ensemble de chaînons de transition globale pour cet ensemble $\tilde{\mathcal{T}}_{pot(0^{(1)}1^{(2)})}$ est présenté dans TAB. 7.12.

Événement	Chaînon de transition globale	Événement	Chaînon de transition globale
e_1	$(0^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, e_1, 1)$ $(0^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, e_1, 1)$ $(0^{(1)}2^{(2)}, 1^{(1)}2^{(2)}, e_1, 1)$	$\bar{e}_1$	$(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_1, 1)$ $(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, \bar{e}_1, 1)$ $(0^{(1)}2^{(2)}, 0^{(1)}2^{(2)}, \bar{e}_1, 1)$
e_4	$(0^{(1)}0^{(2)}, 0^{(1)}2^{(2)}, e_4, 1)$ $(0^{(1)}1^{(2)}, 0^{(1)}2^{(2)}, e_4, 1)$ $(1^{(1)}0^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)$ $(1^{(1)}1^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)$ $(2^{(1)}0^{(2)}, 2^{(1)}2^{(2)}, e_4, 1)$ $(2^{(1)}1^{(2)}, 2^{(1)}2^{(2)}, e_4, 1)$	$\bar{e}_4$	$(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_4, 1)$ $(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, \bar{e}_4, 1)$ $(1^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, \bar{e}_4, 1)$ $(1^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, \bar{e}_4, 1)$ $(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_4, 1)$ $(2^{(1)}1^{(2)}, 2^{(1)}1^{(2)}, \bar{e}_4, 1)$

TAB. 7.12 – Ensemble de chaînons de transition globale $\tilde{\mathcal{T}}_{pot(0^{(1)}1^{(2)})}$ de FIG. 7.14

À partir de ces chaînons, on peut construire les chaînes de transitions globales (TAB. 7.13) pour l'état global $0^{(1)}1^{(2)}$ et calculer les probabilités d'occurrence des transitions.

État d'arrivée	Chaîne de transitions globales	Probabilité de transition
$0^{(1)}1^{(2)}$	$\{(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, \bar{e}_1, 1); (0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, \bar{e}_4, 1)\}$	$(1-\rho_1)(1-\rho_4)$
$0^{(1)}2^{(2)}$	$\{(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, \bar{e}_1, 1); (0^{(1)}1^{(2)}, 0^{(1)}2^{(2)}, e_4, 1)\}$	$(1-\rho_1)\rho_4$
$1^{(1)}1^{(2)}$	$\{(0^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, e_1, 1); (1^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, \bar{e}_4, 1)\}$	$\rho_1(1-\rho_4)$
$1^{(1)}2^{(2)}$	$\{(0^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, e_1, 1); (1^{(1)}1^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)\}$	$\rho_1\rho_4$

TAB. 7.13 – Ensemble des chaînes de transitions globales de l'état global $0^{(1)}1^{(2)}$ de FIG. 7.14

Dans FIG. 7.16, on présente les chemins de construction des chaînes de transitions globales pour l'état global $0^{(1)}1^{(2)}$.

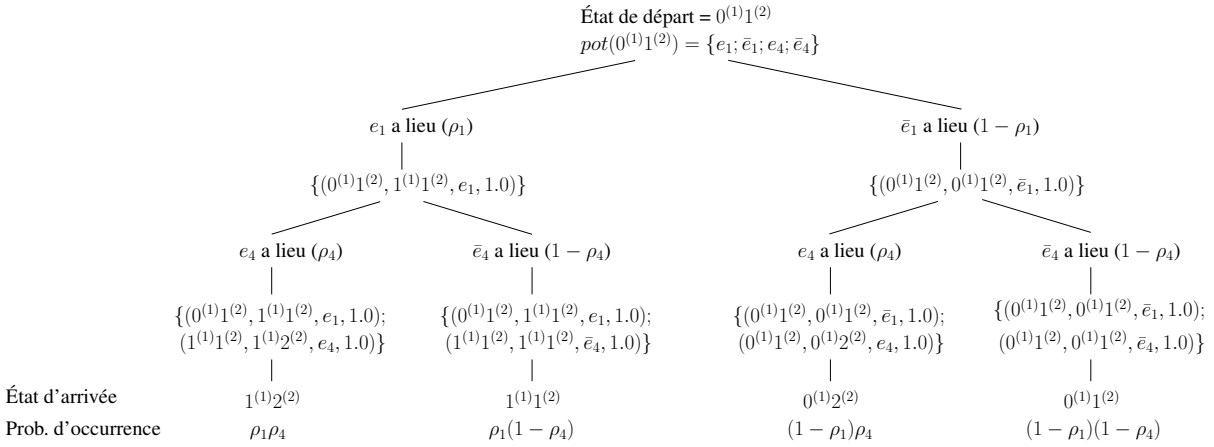


FIG. 7.16 – Construction des chaînes de transitions globales pour l'état $0^{(1)}1^{(2)}$

Pour l'état global $0^{(1)}2^{(2)}$, l'ensemble d'événements possibles est $pot(0^{(1)}2^{(2)}) = \{e_1, \bar{e}_1\}$. L'ensemble de chaînons de transition globale pour cet ensemble $\tilde{\mathcal{T}}_{pot(0^{(1)}0^{(2)})}$ est présenté dans TAB. 7.14.

Événement	Chaînon de transition globale	Événement	Chaînon de transition globale
e_1	$(0^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, e_1, 1)$	$\bar{e}_1$	$(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_1, 1)$
	$(0^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, e_1, 1)$		$(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, \bar{e}_1, 1)$
	$(0^{(1)}2^{(2)}, 1^{(1)}2^{(2)}, e_1, 1)$		$(0^{(1)}2^{(2)}, 0^{(1)}2^{(2)}, \bar{e}_1, 1)$

TAB. 7.14 – Ensemble de chaînons de transition globale $\tilde{\mathcal{T}}_{pot(0^{(1)}2^{(2)})}$ de FIG. 7.14

À partir de ces chaînons, on peut construire les chaînes de transitions globales suivantes (TAB. 7.15) pour l'état global $0^{(1)}2^{(2)}$, ainsi que les probabilités d'occurrence de chaque transition.

État d'arrivée	Chaîne de transitions globales	Probabilité de transition
$0^{(1)}2^{(2)}$	$\{(0^{(1)}2^{(2)}, 0^{(1)}2^{(2)}, \bar{e}_1, 1)\}$	$(1-\rho_1)$
$1^{(1)}2^{(2)}$	$\{(0^{(1)}2^{(2)}, 1^{(1)}2^{(2)}, e_1, 1)\}$	ρ_1

TAB. 7.15 – Ensemble des chaînes de transitions globales de l'état global $0^{(1)}2^{(2)}$ de FIG. 7.14

Dans FIG. 7.17, on présente les chemins de construction des chaînes de transitions globales pour l'état global $0^{(1)}2^{(2)}$.

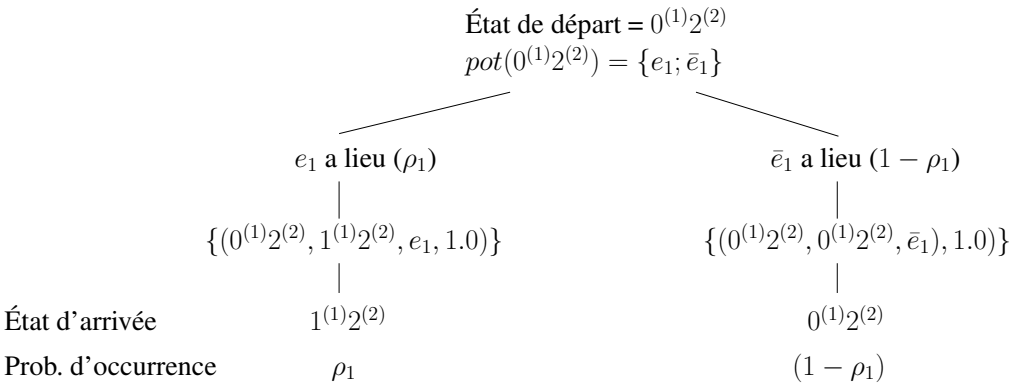


FIG. 7.17 – Construction des chaînes de transitions globales pour l'état $0^{(1)}2^{(2)}$

Pour l'état global $1^{(1)}0^{(2)}$, l'ensemble d'événements possibles est $pot(1^{(1)}0^{(2)}) = \{e_1, \bar{e}_1, e_4, \bar{e}_4\}$. L'ensemble de chaînons de transition globale pour cet ensemble $\tilde{\mathcal{T}}_{pot(1^{(1)}0^{(2)})}$ est présenté dans TAB. 7.16.

Événement	Chaînon de transition globale	Événement	Chaînon de transition globale
e_1	$(0^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, e_1, 1)$	$\bar{e}_1$	$(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_1, 1)$
	$(0^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, e_1, 1)$		$(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, \bar{e}_1, 1)$
	$(0^{(1)}2^{(2)}, 1^{(1)}2^{(2)}, e_1, 1)$		$(0^{(1)}2^{(2)}, 0^{(1)}2^{(2)}, \bar{e}_1, 1)$
e_4	$(0^{(1)}0^{(2)}, 0^{(1)}2^{(2)}, e_4, 1)$	$\bar{e}_4$	$(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_4, 1)$
	$(0^{(1)}1^{(2)}, 0^{(1)}2^{(2)}, e_4, 1)$		$(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, \bar{e}_4, 1)$
	$(1^{(1)}0^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)$		$(1^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, \bar{e}_4, 1)$
	$(1^{(1)}1^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)$		$(1^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, \bar{e}_4, 1)$
	$(2^{(1)}0^{(2)}, 2^{(1)}2^{(2)}, e_4, 1)$		$(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_4, 1)$
	$(2^{(1)}1^{(2)}, 2^{(1)}2^{(2)}, e_4, 1)$		$(2^{(1)}1^{(2)}, 2^{(1)}1^{(2)}, \bar{e}_4, 1)$

TAB. 7.16 – Ensemble de chaînons de transition globale $\tilde{\mathcal{T}}_{pot(1^{(1)}0^{(2)})}$ de FIG. 7.14

On remarque que les événements e_1 et $\bar{e}_1$ sont des événements possibles à partir de cet état, mais ils ne sont pas réalisables. En sachant que les événements e_1 et $\bar{e}_1$ sont les événements plus prioritaires de $pot(1^{(1)}0^{(2)})$ et donc, aucun autre événement peut avoir lieu avant eux, ils ne apparaissent dans aucune chaîne de transitions globales à partir de cet état.

À partir de ces chaînons, on peut construire les chaînes de transitions globales suivantes (TAB. 7.17) pour l'état global $1^{(1)}0^{(2)}$.

État d'arrivée	Chaîne de transitions globales	Probabilité de transition
$1^{(1)}0^{(2)}$	$\{(1^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, \bar{e}_4, 1)\}$	$(1-\rho_4)$
$1^{(1)}2^{(2)}$	$\{(1^{(1)}0^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)\}$	ρ_4

TAB. 7.17 – Ensemble des chaînes de transitions globales de l'état global $1^{(1)}0^{(2)}$ de FIG. 7.14

Dans FIG. 7.18, on présente les chemins de construction des chaînes de transitions globales pour l'état global $1^{(1)}0^{(2)}$.

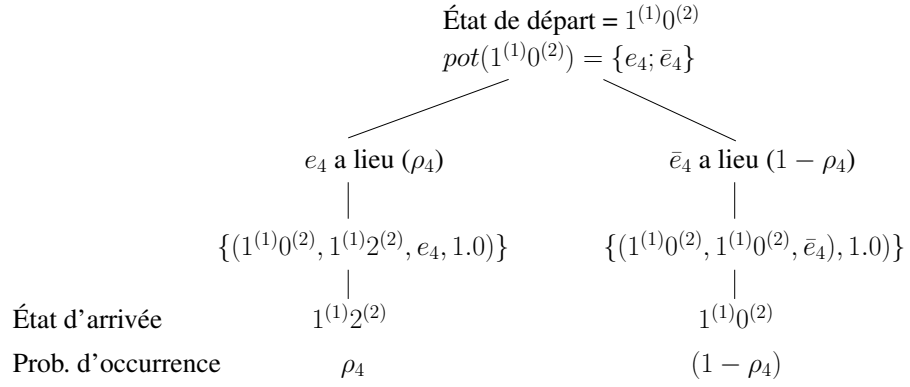


FIG. 7.18 – Construction des chaînes de transitions globales pour l'état $1^{(1)}0^{(2)}$

De façon similaire à l'état global précédente, pour l'état global $1^{(1)}1^{(2)}$, l'ensemble d'événements possibles est composé uniquement de l'événement e_4 et de son événement complémentaire $\bar{e}_4$. L'ensemble de chaînons de transition globale pour cet ensemble $\tilde{\mathcal{T}}_{pot(1^{(1)}1^{(2)})}$ est le même présenté dans TAB. 7.16.

Cependant, les chaînes de transitions globales sont différentes, car l'état de départ n'est pas le même. Dans TAB. 7.18, on présente les chaînes de transitions globales pour l'état global $1^{(1)}1^{(2)}$.

État d'arrivée	Chaîne de transitions globales	Probabilité de transition
$1^{(1)}1^{(2)}$	$\{(1^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, \bar{e}_4, 1)\}$	$(1-\rho_4)$
$1^{(1)}2^{(2)}$	$\{(1^{(1)}1^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)\}$	ρ_4

TAB. 7.18 – Ensemble des chaînes de transitions globales de l'état global $1^{(1)}1^{(2)}$ de FIG. 7.14

Dans FIG. 7.19, on présente les chemins de construction des chaînes de transitions globales pour l'état global $1^{(1)}1^{(2)}$.

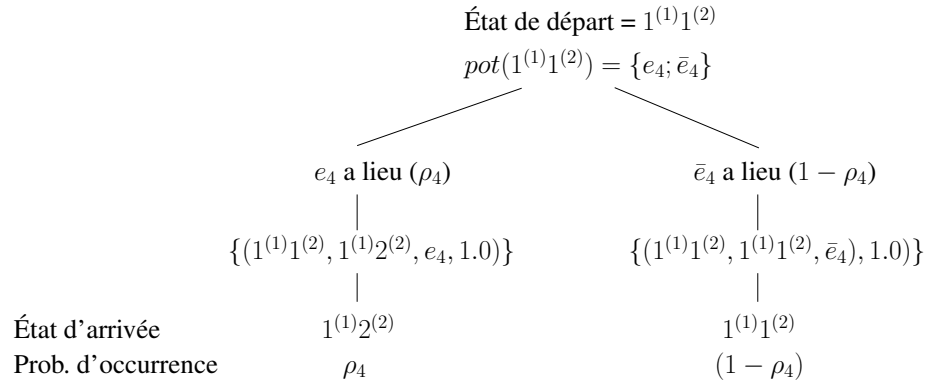


FIG. 7.19 – Construction des chaînes de transitions globales pour l'état $1^{(1)}1^{(2)}$

Pour l'état global $1^{(1)}2^{(2)}$, l'ensemble d'événements possibles est $pot(1^{(1)}2^{(2)}) = \{e_3, \bar{e}_3, e_1, \bar{e}_1\}$. L'ensemble de chaînons de transition globale pour cet ensemble $\tilde{\mathcal{T}}_{pot(1^{(1)}2^{(2)})}$ est présenté dans TAB. 7.19.

Événement	Chaînon de transition globale	Événement	Chaînon de transition globale
e_3	$(1^{(1)}2^{(2)}, 0^{(1)}0^{(2)}, e_3, \pi_1)$ $(1^{(1)}2^{(2)}, 2^{(1)}0^{(2)}, e_3, \pi_2)$ $(2^{(1)}2^{(2)}, 0^{(1)}0^{(2)}, e_3, 1)$	$\bar{e}_3$	$(1^{(1)}2^{(2)}, 1^{(1)}2^{(2)}, \bar{e}_3, 1)$ $(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}, \bar{e}_3, 1)$
e_1	$(0^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, e_1, 1)$ $(0^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, e_1, 1)$ $(0^{(1)}2^{(2)}, 1^{(1)}2^{(2)}, e_1, 1)$	$\bar{e}_1$	$(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_1, 1)$ $(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, \bar{e}_1, 1)$ $(0^{(1)}2^{(2)}, 0^{(1)}2^{(2)}, \bar{e}_1, 1)$

TAB. 7.19 – Ensemble de chaînons de transition globale $\tilde{\mathcal{T}}_{pot(1^{(1)}2^{(2)})}$ de FIG. 7.14

À partir de ces chaînons, on peut construire les chaînes de transitions globales suivantes (TAB. 7.20) pour l'état global $1^{(1)}2^{(2)}$.

État d'arrivée	Chaîne de transitions globales	Probabilité de transition
$0^{(1)}0^{(2)}$	$\{(1^{(1)}2^{(2)}, 0^{(1)}0^{(2)}, e_3, \pi_1); (0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_1, 1)\}$	$\rho_3\pi_1(1-\rho_1)$
$1^{(1)}0^{(2)}$	$\{(1^{(1)}2^{(2)}, 0^{(1)}0^{(2)}, e_3, \pi_1); (0^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, e_1, 1)\}$	$\rho_3\pi_1\rho_1$
$1^{(1)}2^{(2)}$	$\{(1^{(1)}2^{(2)}, 1^{(1)}2^{(2)}, \bar{e}_3, 1)\}$	$(1-\rho_3)$
$2^{(1)}0^{(2)}$	$\{(1^{(1)}2^{(2)}, 2^{(1)}0^{(2)}, e_3, \pi_2)\}$	$\rho_3\pi_2$

TAB. 7.20 – Ensemble des chaînes de transitions globales de l'état global $1^{(1)}2^{(2)}$ de FIG. 7.14

Dans FIG. 7.20, on présente les chemins de construction des chaînes de transitions globales pour l'état global $1^{(1)}2^{(2)}$.

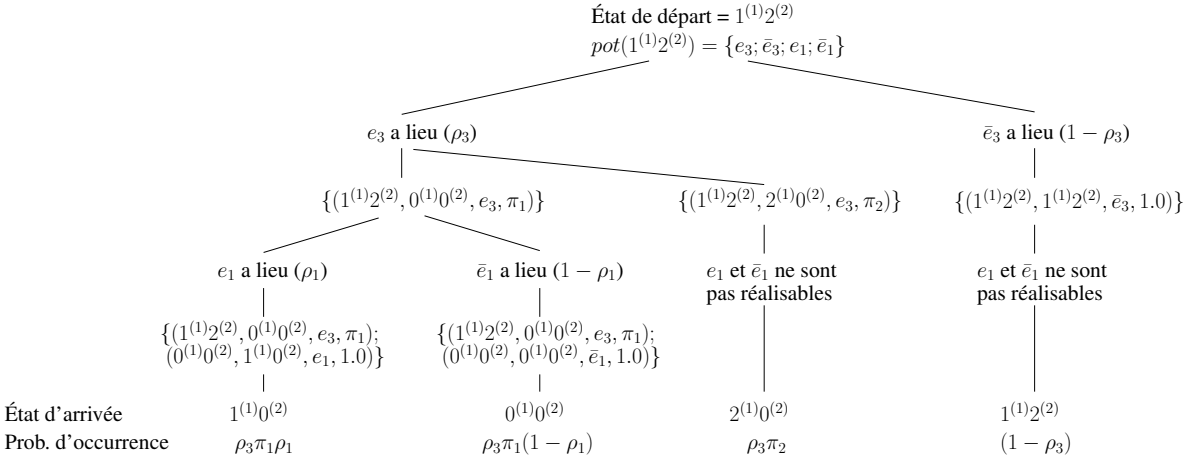


FIG. 7.20 – Construction des chaînes de transitions globales pour l'état $1^{(1)}2^{(2)}$

Pour l'état global $2^{(1)}0^{(2)}$, l'ensemble d'événements possibles est $pot(2^{(1)}0^{(2)}) = \{e_2, \bar{e}_2, e_4, \bar{e}_4, e_5, \bar{e}_5\}$. L'ensemble de chaînons de transition globale pour cet ensemble $\tilde{\mathcal{T}}_{pot(2^{(1)}0^{(2)})}$ est présenté dans TAB. 7.21.

Événement	Chaînon de transition globale	Événement	Chaînon de transition globale
e_2	$(2^{(1)}0^{(2)}, 1^{(1)}1^{(2)}, e_2, 1)$	$\bar{e}_2$	$(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_2, 1)$
e_4	$(0^{(1)}0^{(2)}, 0^{(1)}2^{(2)}, e_4, 1)$	$\bar{e}_4$	$(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_4, 1)$
	$(0^{(1)}1^{(2)}, 0^{(1)}2^{(2)}, e_4, 1)$		$(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, \bar{e}_4, 1)$
	$(1^{(1)}0^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)$		$(1^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, \bar{e}_4, 1)$
	$(1^{(1)}1^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)$		$(1^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, \bar{e}_4, 1)$
	$(2^{(1)}0^{(2)}, 2^{(1)}2^{(2)}, e_4, 1)$		$(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_4, 1)$
	$(2^{(1)}1^{(2)}, 2^{(1)}2^{(2)}, e_4, 1)$		$(2^{(1)}1^{(2)}, 2^{(1)}1^{(2)}, \bar{e}_4, 1)$
e_5	$(2^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, e_5, 1)$	$\bar{e}_5$	$(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_5, 1)$
	$(2^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, e_5, 1)$		$(2^{(1)}1^{(2)}, 2^{(1)}1^{(2)}, \bar{e}_5, 1)$
	$(2^{(1)}2^{(2)}, 0^{(1)}2^{(2)}, e_5, 1)$		$(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}, \bar{e}_5, 1)$

TAB. 7.21 – Ensemble de chaînons de transition globale $\tilde{\mathcal{T}}_{pot(2^{(1)}0^{(2)})}$ de FIG. 7.14

À partir de ces chaînons, on peut construire les chaînes de transitions globales suivantes (TAB. 7.22) pour l'état global $2^{(1)}0^{(2)}$.

État d'arrivée	Chaîne de transitions globales	Probabilité de transition
$0^{(1)}0^{(2)}$	$\{(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_2, 1); (2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_4, 1); (2^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, e_5, 1)\}$	0
$0^{(1)}2^{(2)}$	$\{(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_2, 1); (2^{(1)}0^{(2)}, 2^{(1)}2^{(2)}, e_4, 1); (2^{(1)}2^{(2)}, 0^{(1)}2^{(2)}, e_5, 1)\}$	$(1-\rho_2)\rho_4\rho_5$
$1^{(1)}1^{(2)}$	$\{(2^{(1)}0^{(2)}, 1^{(1)}1^{(2)}, e_2, 1); (1^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, \bar{e}_4, 1)\}$	$\rho_2(1-\rho_4)$
$1^{(1)}2^{(2)}$	$\{(2^{(1)}0^{(2)}, 1^{(1)}1^{(2)}, e_2, 1); (1^{(1)}1^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)\}$	$\rho_2\rho_4$
$2^{(1)}0^{(2)}$	$\{(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_2, 1); (2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_4, 1); (2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_5, 1)\}$	$(1-\rho_2)(1-\rho_4)$
$2^{(1)}2^{(2)}$	$\{(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_2, 1); (2^{(1)}0^{(2)}, 2^{(1)}2^{(2)}, e_4, 1); (2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}, \bar{e}_5, 1)\}$	$(1-\rho_2)\rho_4(1-\rho_5)$

TAB. 7.22 – Ensemble des chaînes de transitions globales de l'état global $2^{(1)}0^{(2)}$ de FIG. 7.14

Dans FIG. 7.21, on présente les chemins de construction des chaînes de transitions globales pour l'état global $2^{(1)}0^{(2)}$.

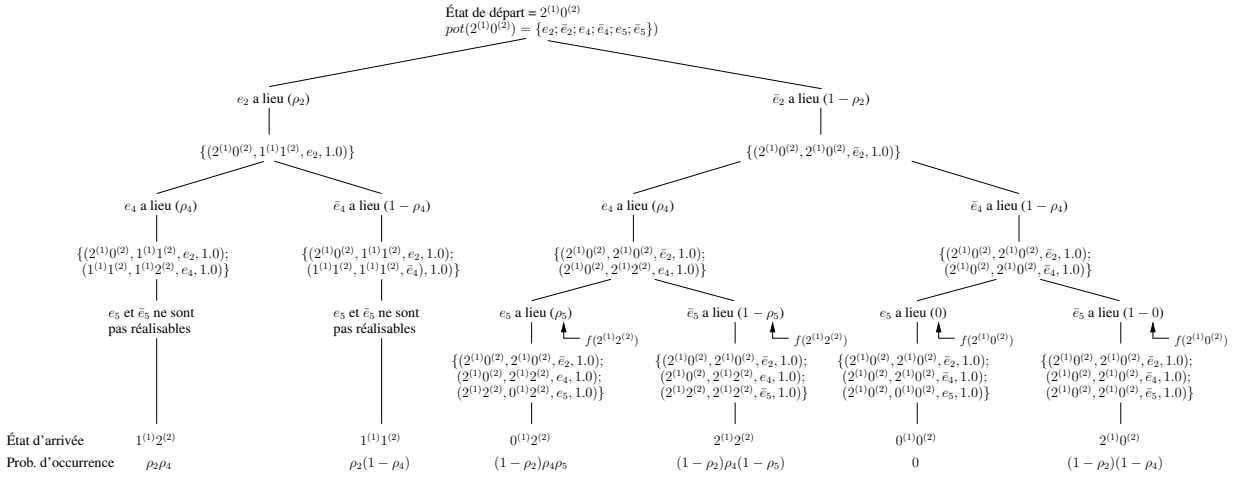


FIG. 7.21 – Construction des chaînes de transitions globales pour l'état $2^{(1)}0^{(2)}$

Pour l'état global $2^{(1)}1^{(2)}$, l'ensemble d'événements possibles est $pot(2^{(1)}1^{(2)}) = \{e_4, \bar{e}_4, e_5, \bar{e}_5\}$. L'ensemble de chaînons de transition globale pour cet ensemble $\tilde{\mathcal{T}}_{pot(2^{(1)}1^{(2)})}$ est présenté dans TAB. 7.23.

Événement	Chaînon de transition globale	Événement	Chaînon de transition globale
e_4	$(0^{(1)}0^{(2)}, 0^{(1)}2^{(2)}, e_4, 1)$ $(0^{(1)}1^{(2)}, 0^{(1)}2^{(2)}, e_4, 1)$ $(1^{(1)}0^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)$ $(1^{(1)}1^{(2)}, 1^{(1)}2^{(2)}, e_4, 1)$ $(2^{(1)}0^{(2)}, 2^{(1)}2^{(2)}, e_4, 1)$ $(2^{(1)}1^{(2)}, 2^{(1)}2^{(2)}, e_4, 1)$	$\bar{e}_4$	$(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, \bar{e}_4, 1)$ $(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, \bar{e}_4, 1)$ $(1^{(1)}0^{(2)}, 1^{(1)}0^{(2)}, \bar{e}_4, 1)$ $(1^{(1)}1^{(2)}, 1^{(1)}1^{(2)}, \bar{e}_4, 1)$ $(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_4, 1)$ $(2^{(1)}1^{(2)}, 2^{(1)}1^{(2)}, \bar{e}_4, 1)$
e_5	$(2^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, e_5, 1)$ $(2^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, e_5, 1)$ $(2^{(1)}2^{(2)}, 0^{(1)}2^{(2)}, e_5, 1)$	$\bar{e}_5$	$(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_5, 1)$ $(2^{(1)}1^{(2)}, 2^{(1)}1^{(2)}, \bar{e}_5, 1)$ $(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}, \bar{e}_5, 1)$

TAB. 7.23 – Ensemble de chaînons de transition globale $\tilde{\mathcal{T}}_{pot(2^{(1)}1^{(2)})}$ de FIG. 7.14

À partir de ces chaînons, on peut construire les chaînes de transitions globales suivantes (TAB. 7.24) pour l'état global $2^{(1)}1^{(2)}$.

État d'arrivée	Chaîne de transitions globales	Probabilité de transition
$0^{(1)}1^{(2)}$	$\{(2^{(1)}1^{(2)}, 2^{(1)}1^{(2)}, \bar{e}_4, 1); (2^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, e_5, 1)\}$	0
$0^{(1)}2^{(2)}$	$\{(2^{(1)}1^{(2)}, 2^{(1)}2^{(2)}, e_4, 1); (2^{(1)}2^{(2)}, 0^{(1)}2^{(2)}, e_5, 1)\}$	$\rho_4\rho_5$
$2^{(1)}1^{(2)}$	$\{(2^{(1)}1^{(2)}, 2^{(1)}1^{(2)}, \bar{e}_4, 1); (2^{(1)}1^{(2)}, 2^{(1)}1^{(2)}, \bar{e}_5, 1)\}$	$(1-\rho_4)$
$2^{(1)}2^{(2)}$	$\{(2^{(1)}1^{(2)}, 2^{(1)}2^{(2)}, e_4, 1); (2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}, \bar{e}_5, 1)\}$	$\rho_4(1-\rho_5)$

TAB. 7.24 – Ensemble des chaînes de transitions globales de l'état global $2^{(1)}1^{(2)}$ de FIG. 7.14

Dans FIG. 7.22, on présente les chemins de construction des chaînes de transitions globales pour l'état global $2^{(1)}1^{(2)}$.

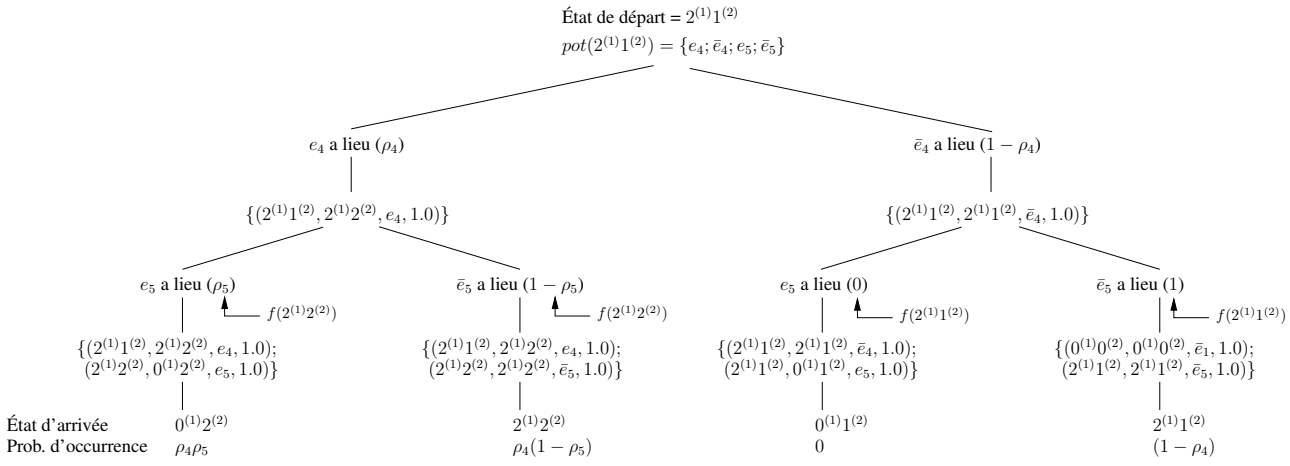


FIG. 7.22 – Construction des chaînes de transitions globales pour l'état $2^{(1)}1^{(2)}$

Pour l'état global $2^{(1)}2^{(2)}$ l'ensemble d'événements possibles est $pot(2^{(1)}2^{(2)}) = \{e_3, \bar{e}_3, e_5, \bar{e}_5\}$. L'ensemble de chaînons de transition globale pour cet ensemble $\tilde{\mathcal{T}}_{pot(2^{(1)}2^{(2)})}$ est présenté dans TAB. 7.25.

Événement	Chaînon de transition globale	Événement	Chaînon de transition globale
e_3	$(1^{(1)}2^{(2)}, 0^{(1)}0^{(2)}, e_3, \pi_1)$ $(1^{(1)}2^{(2)}, 2^{(1)}0^{(2)}, e_3, \pi_2)$ $(2^{(1)}2^{(2)}, 0^{(1)}0^{(2)}, e_3, 1)$	$\bar{e}_3$	$(1^{(1)}2^{(2)}, 1^{(1)}2^{(2)}, \bar{e}_3, 1)$ $(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}, \bar{e}_3, 1)$
e_5	$(2^{(1)}0^{(2)}, 0^{(1)}0^{(2)}, e_5, 1)$ $(2^{(1)}1^{(2)}, 0^{(1)}1^{(2)}, e_5, 1)$ $(2^{(1)}2^{(2)}, 0^{(1)}2^{(2)}, e_5, 1)$	$\bar{e}_5$	$(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}, \bar{e}_5, 1)$ $(2^{(1)}1^{(2)}, 2^{(1)}1^{(2)}, \bar{e}_5, 1)$ $(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}, \bar{e}_5, 1)$

TAB. 7.25 – Ensemble de chaînons de transition globale $\tilde{\mathcal{T}}_{pot(2^{(1)}2^{(2)})}$ de FIG. 7.14

À partir de ces chaînons, on peut construire les chaînes de transitions globales suivantes (TAB. 7.26) pour l'état global $2^{(1)}2^{(2)}$.

État d'arrivée	Chaîne de transitions globales	Probabilité de transition
$0^{(1)}0^{(2)}$	$\{(2^{(1)}2^{(2)}, 0^{(1)}0^{(2)}, e_3, 1)\}$	ρ_3
$0^{(1)}2^{(2)}$	$\{(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}, \bar{e}_3, 1); (2^{(1)}2^{(2)}, 0^{(1)}2^{(2)}, e_5, 1)\}$	$(1-\rho_3)\rho_5$
$2^{(1)}2^{(2)}$	$\{(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}, \bar{e}_3, 1); (2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}, \bar{e}_5, 1)\}$	$(1-\rho_3)(1-\rho_5)$

TAB. 7.26 – Ensemble des chaînes de transitions globales de l'état global $2^{(1)}2^{(2)}$ de FIG. 7.14

Dans FIG. 7.23, on présente les chemins de construction des chaînes de transitions globales pour l'état global $2^{(1)}2^{(2)}$.

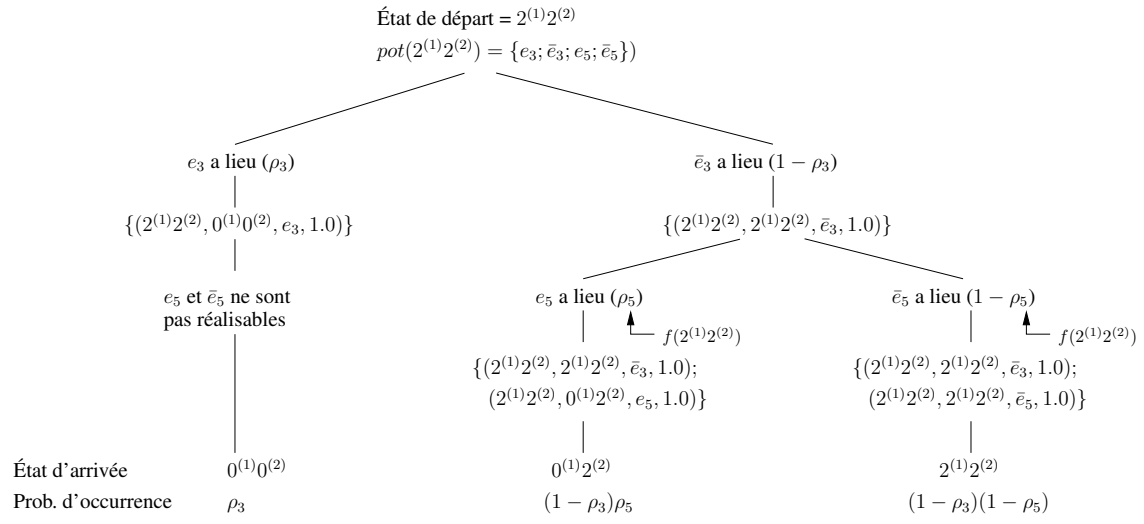


FIG. 7.23 – Construction des chaînes de transitions globales pour l'état $2^{(1)}2^{(2)}$

Avec les probabilités de transitions calculées à partir des chaînes de transitions globales, on peut en déduire la chaîne de Markov représentée par le modèle SAN (FIG. 7.24).

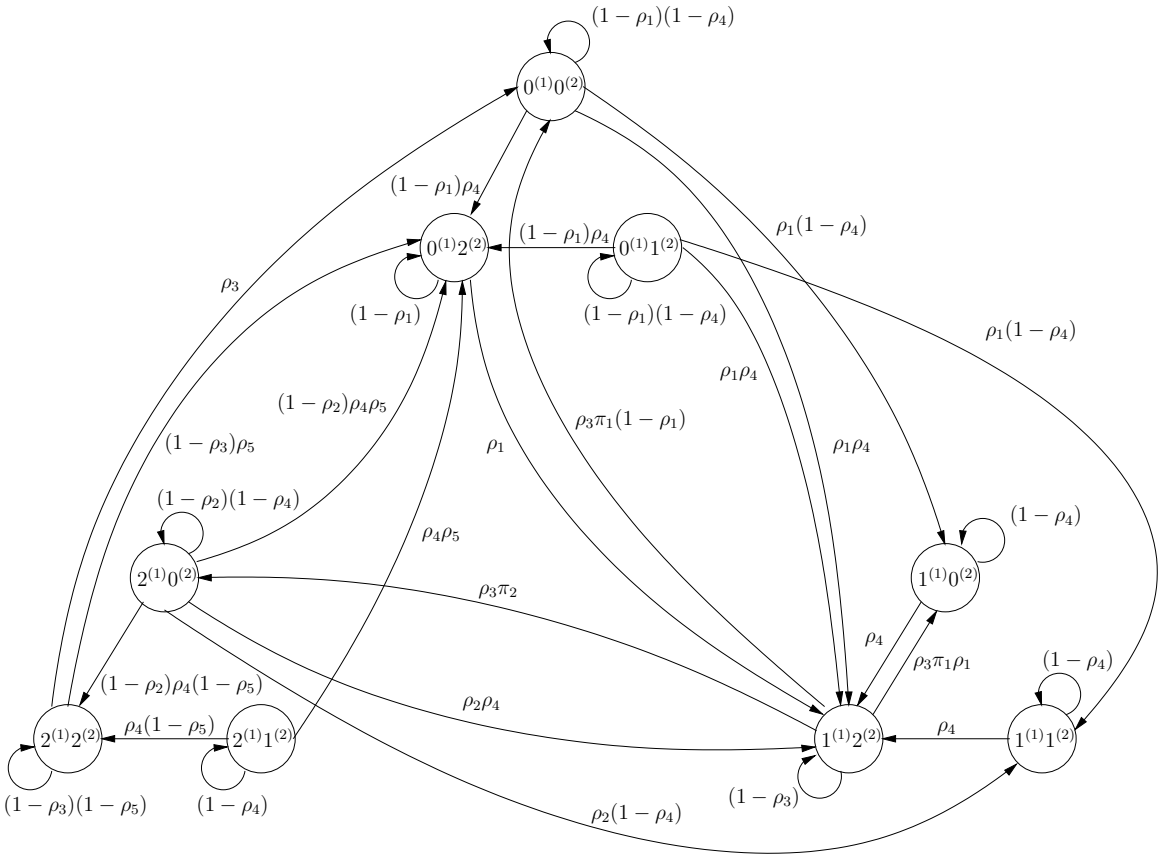


FIG. 7.24 – Chaîne de Markov équivalente au modèle de FIG. 7.11.

Et sa matrice de transition (les états étant dans l'ordre lexicographique) :

$$P = \left(\begin{array}{ccc|ccc|ccc}
(1-\rho_1)(1-\rho_4) & 0 & (1-\rho_1)\rho_4 & \rho_1(1-\rho_4) & 0 & \rho_1\rho_4 & 0 & 0 & 0 \\
0 & (1-\rho_1)(1-\rho_4) & (1-\rho_1)\rho_4 & 0 & \rho_1(1-\rho_4) & \rho_1\rho_4 & 0 & 0 & 0 \\
0 & 0 & (1-\rho_1) & 0 & 0 & \rho_1 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & (1-\rho_4) & 0 & \rho_4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & (1-\rho_4) & \rho_4 & 0 & 0 & 0 \\
\rho_3\pi_1(1-\rho_1) & 0 & 0 & \rho_3\pi_1\rho_1 & 0 & (1-\rho_3) & \rho_3\pi_2 & 0 & 0 \\
\hline
0 & 0 & (1-\rho_2)\rho_4\rho_5 & 0 & \rho_2(1-\rho_4) & \rho_2\rho_4 & (1-\rho_2)(1-\rho_4) & 0 & (1-\rho_2)\rho_4(1-\rho_5) \\
0 & 0 & \rho_4\rho_5 & 0 & 0 & 0 & 0 & (1-\rho_4) & \rho_4(1-\rho_5) \\
\rho_3 & 0 & (1-\rho_3)\rho_5 & 0 & 0 & 0 & 0 & 0 & (1-\rho_3)(1-\rho_5)
\end{array}\right)$$

7.5 Conclusion

On a présenté dans ce chapitre une définition formelle pour les réseaux d'automates stochastiques à temps discret.

Tout d'abord, on a défini les notations de base d'un automate et d'un réseau d'automates. À partir de ces définitions de base et de la matrice de transition locale de chaque automate, on a introduit la définition d'automate complété et de réseau d'automates complétés. Un automate complété ou un réseau

d'automates complétés inclut tous les transition définies pour un automate et aussi les transitions qui représentent la *non-occurrence* de chaque événement. La *non-occurrence* d'un événement a été représentée par l'ajout d'un événement complémentaire.

À partir d'un réseau d'automates complétés, on a formalisé l'automate global représenté par le réseau d'automates. L'automate global nous a permis par la suite de définir la matrice de transition de la chaîne de Markov représentée par le modèle SAN. Ceci nous donne la sémantique, en terme de processus aléatoire de l'évolution d'un modèle SAN à temps discret.

Dans le chapitre suivant, on va présenter quelques exemples décrits par le formalisme SAN à temps discret.

Chapitre 8

Exemples de modélisation à temps discret

Dans ce chapitre, on présente quelques exemples de modélisation afin d'illustrer l'utilisation du formalisme des Réseaux d'Automates Stochastiques (SAN - *Stochastic Automata Networks*) à temps discret présenté dans le chapitre 7.

Ces exemples ont pour but d'illustrer les caractéristiques de modélisation du formalisme SAN à temps discret.

Les systèmes informatiques présentés dans ce chapitre sont mêmes que ceux des exemples présentés dans le chapitre 3 :

- *Dîner des Philosophes* (Section 8.1) : un problème classique qui concerne l'ordonnancement de processus et l'allocation de ressources à ces derniers ;
- *Patron de Service Alterné* (Section 8.2) : un exemple d'un réseau de files d'attente ouvert, où ces files d'attente ont des capacités finies (avec blocage ou perte) et peuvent servir les clients suivant différentes probabilités ;
- *Atelier avec kanbans* (Section 8.3) : un exemple d'un modèle d'un atelier avec quatre postes de travail où des *kanbans* (*tickets*) sont utilisés pour contrôler les flux de pièces ;
- *Partage de Ressources* (Section 8.4) : un exemple de modélisation du problème classique de partage des ressources.

On va insister dans ces exemples sur la combinatoire possible des événements dans la même unité de temps : caractéristique particulière de la modélisation de systèmes en temps discret.

8.1 Dîner des Philosophes

La description du problème présenté pour ce modèle SAN à temps discret est la même que celle présentée pour le modèle SAN à temps continu, *i.e.*, un nombre N de philosophes se trouve autour

d'une table ; chaque philosophe a devant lui un plat de spaghetti et une fourchette se trouve à droite de chaque assiette. Lorsque un philosophe a faim, il va essayer de prendre des fourchettes pour manger. Un philosophe a besoin de deux fourchettes pour manger : celle qui se trouve à droite de sa propre assiette, et celle qui se trouve à droite de celle de son voisin à gauche, *i.e.*, les fourchettes qui se trouvent autour de sa propre assiette.

Le modèle SAN à temps discret (présenté dans FIG. 8.1) représente chaque philosophe par un automate $\mathcal{P}^{(i)}$ tel que $i \in [1..N]$, où chaque automate possède trois états : T (*thinking*) qui indique que le philosophe pense pendant un temps ; R (*right*) s'il possède la fourchette qui se trouve à droite de son assiette ; et L (*left*) s'il possède la fourchette qui se trouve à gauche de son assiette.

Le problème consiste à trouver un ordonnancement des philosophes tel qu'ils puissent tous manger, chacun à leur tour. On présente le comportement des philosophes pour ce modèle SAN à temps discret.

Le philosophe $\mathcal{P}^{(i)}$ reste pensif (l'état $T^{(i)}$) pendant un temps et lorsqu'il a faim, il va essayer de prendre la fourchette qui se trouve à droite de son assiette (l'état $R^{(i)}$). Alors, pour prendre sa fourchette droite, il faut que la fourchette gauche de son voisin à droite[1] (le philosophe $\mathcal{P}^{(i+1)}$) soit disponible, *i.e.*, que son voisin de droite doit se trouver soit dans l'état $T^{(i+1)}$, soit dans l'état $R^{(i+1)}$. Le philosophe $\mathcal{P}^{(i)}$ change de l'état $T^{(i)}$ vers l'état $R^{(i)}$ (*i.e.*, il prend sa fourchette droite) par l'occurrence de l'événement r_i.

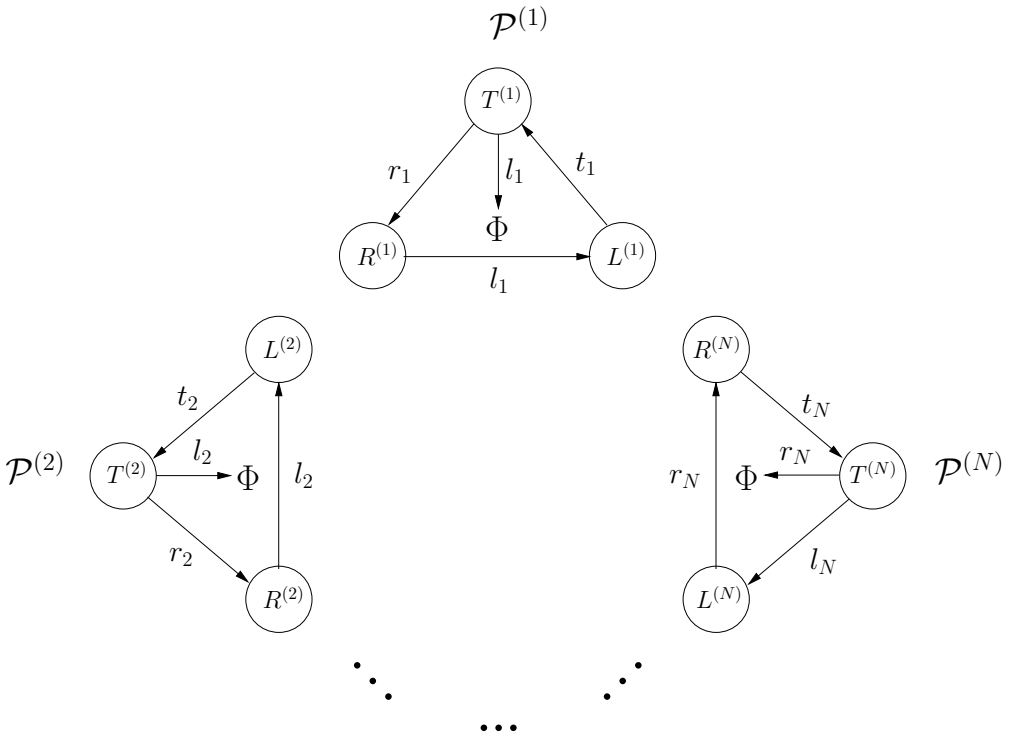
Pour manger effectivement, le philosophe $\mathcal{P}^{(i)}$ prend la deuxième fourchette, *i.e.*, la fourchette qui se trouve à gauche de son assiette, en changeant de l'état $R^{(i)}$ vers l'état $L^{(i)}$ par l'occurrence de l'événement l_i. Toutefois, pour prendre sa fourchette gauche, il faut que son voisin à gauche (le philosophe $\mathcal{P}^{(i-1)}$) se trouve dans l'état $T^{(i-1)}$, *i.e.*, que le philosophe $\mathcal{P}^{(i-1)}$ ne mange pas et par conséquence qu'il n'utilise pas sa fourchette droite (l'état $R^{(i-1)}$). Une fois que le philosophe a mangé, il revient à l'état $T^{(i)}$, *i.e.*, il change de l'état $L^{(i)}$ vers l'état $T^{(i)}$ par l'occurrence de l'événement t_i. Les événement r_i et l_i dans ce modèle (FIG. 8.1) se produisent avec une probabilité exprimée par les fonctions fr_i et fl_i respectivement.

Remarquons que le dernier philosophe ($\mathcal{P}^{(N)}$) est gaucher. Dans ce cas, il va essayer de prendre sa fourchette gauche avant sa fourchette droite, afin d'éviter une situation d'interblocage.

Dans le contexte des SAN à temps discret, plusieurs événements peuvent avoir lieu dans la même unité de temps, *e.g.*, entre philosophes non voisins, tous les événements peuvent avoir lieu dans une même unité de temps. Alors, on définit pour ce modèle qu'un philosophe $\mathcal{P}^{(i)}$ peut prendre les deux fourchettes dès qu'elles soient disponibles, *i.e.*, il peut passer dans une même unité de temps de l'état $T^{(i)}$ vers l'état $L^{(i)}$ par l'occurrence des événements r_i et l_i. Afin de permettre cette possibilité, on modélise une transition de l'état $T^{(i)}$ vers l'état fantôme Φ par l'occurrence de l'événement l_i, sachant qu'après l'occurrence de l'événement r_i, l'événement l_i sera autorisé à être tiré dans la même unité de temps. Sachant aussi que le dernier philosophe ($\mathcal{P}^{(N)}$) est gaucher, on modélise la transition de l'état $T^{(N)}$ vers l'état fantôme Φ par l'événement r_N. Dans ce cas, l'événement r_N pourra être tiré dans la même unité de temps que l'événement l_N, permettant que le philosophe $\mathcal{P}^{(N)}$ passe de l'état $T^{(N)}$ vers l'état $R^{(N)}$.

Dans le modèle SAN présenté dans FIG. 8.1, le changement d'état du philosophe $\mathcal{P}^{(i)}$ nécessite l'évaluation des fonctions fr_i et fl_i associées aux probabilité d'occurrence des événements r_i et l_i res-

[1] Le philosophe $\mathcal{P}^{(N)}$ se trouve à gauche de $\mathcal{P}^{(1)}$, ainsi que le voisin à droite du philosophe $\mathcal{P}^{(N)}$ est le philosophe $\mathcal{P}^{(1)}$.



Type	Even.	Prob.	Priorité
loc	r_1	fr_1	1
loc	l_1	fl_1	2
loc	t_1	ρ_{t_1}	3
loc	r_2	fr_2	4
loc	l_2	fl_2	5
loc	t_2	ρ_{t_2}	6
⋮	⋮	⋮	⋮
loc	l_N	fl_N	$((N-1)\times 3)+1$
loc	r_N	fr_N	$((N-1)\times 3)+2$
loc	t_N	ρ_{t_N}	$((N-1)\times 3)+3$

$$fr_1 = (st(\mathcal{P}^{(2)})! = L^{(2)}) \times \rho_{r_1}$$
$$fr_2 = (st(\mathcal{P}^{(3)})! = L^{(3)}) \times \rho_{r_2}$$
$$\vdots$$
$$fr_N = (st(\mathcal{P}^{(1)})! = L^{(1)}) \times \rho_{r_N}$$
$$fl_1 = (st(\mathcal{P}^{(N)})! = R^{(N)}) \times \rho_{l_1}$$
$$fl_2 = (st(\mathcal{P}^{(1)}) == T^{(1)}) \times \rho_{l_2}$$
$$\vdots$$
$$fl_N = (st(\mathcal{P}^{(N-1)}) == T^{(N-1)}) \times \rho_{l_N}$$

FIG. 8.1 – Dîner des Philosophes - modèle SAN à temps discret

pectivement, où les arguments de ces fonctions[2] sont les états des philosophes $\mathcal{P}^{(i+1)}$ et $\mathcal{P}^{(i-1)}$, *i.e.*, les états des philosophes voisins.

Une autre caractéristique des modèles SAN à temps discret est l'ordre de priorité des événements qui permet de définir le comportement à suivre en cas de conflit. Dans FIG. 8.1, on a modélisé le philosophe $\mathcal{P}^{(1)}$ plus prioritaire que le philosophe $\mathcal{P}^{(2)}$, qui est plus prioritaire que le philosophe $\mathcal{P}^{(3)}$ et ainsi de suite jusqu'au dernier philosophe $\mathcal{P}^{(N)}$. Et, pour les événements de chaque philosophe $\mathcal{P}^{(i)}$, l'événement r_i est plus prioritaire que l'événement l_i qui est plus prioritaire que l'événement t_i. On remarque que pour les événements du dernier philosophe $\mathcal{P}^{(N)}$, l'événement l_N est plus prioritaire que l'événement r_N qui est plus prioritaire que l'événement t_N.

[2]Dans cette thèse, les fonctions sont définies utilisant la notation adoptée par le logiciel PEPS [17]. L'interprétation d'une fonction est l'évaluation d'une expression de langage de programmation non-typé, *e.g.*, langage C. Chaque comparaison d'une fonction est évaluée à 1 (*vrai*) ou 0 (*faux*).

On remarque que, avec le jeu de priorités des événements, il est possible que le philosophe $\mathcal{P}^{(1)}$ prenne ses fourchettes (par l'occurrence des événements r_1 et l_1) et que le philosophe $\mathcal{P}^{(3)}$ prenne aussi ses fourchettes (par l'occurrence des événements r_3 et l_3) dans la même unité de temps. Par contre, le philosophe $\mathcal{P}^{(4)}$ ne peut pas prendre ses fourchettes dans la même unité de temps que les philosophes $\mathcal{P}^{(1)}$ et $\mathcal{P}^{(3)}$, car les événements de $\mathcal{P}^{(3)}$ (les événements r_3 et l_3) sont plus prioritaires que les événements r_4 et l_4 de $\mathcal{P}^{(4)}$ et l'occurrence des ces événements (r_3 et l_3) amènent vers un état dans lequel les événements r_4 et l_4 ne sont plus réalisables.

8.2 Patron de Service Alterné

On va présenter dans cette section un modèle d'un réseau de files d'attente ouvert avec quatre files (notées Q_1, Q_2, Q_3 et Q_4) de capacité (finie) K_1, K_2, K_3 et K_4 respectivement. Les clients arrivent dans les files $Q1$ et Q_2 avec des probabilités ρ_{λ_1} et ρ_{λ_2} respectivement. Le départ d'un client de la file Q_1 vers la file Q_3 est possible si et seulement il y a de la place disponible dans Q_3 (mécanisme de *blocage*). Le départ d'un client de Q_2 vers Q_3 est possible s'il y a de place disponible dans Q_3 ou, autrement, le client est perdu si la file est pleine (mécanisme de *perte*). Le mécanisme de blocage est aussi utilisé pour les clients qui sortent de Q_3 vers Q_4. Les clients sortent de Q_4 vers l'extérieur du système.

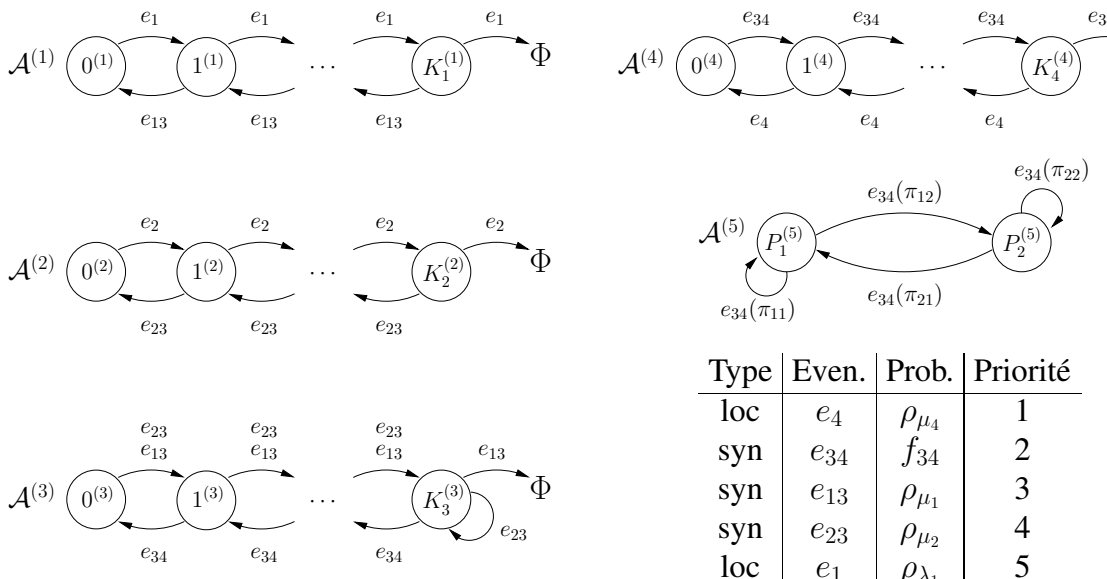
Les stations des files Q_1, Q_2 et Q_4 terminent le service des clients avec une probabilité ρ_{μ_1}, ρ_{μ_2} et ρ_{μ_4} respectivement. Toutefois, le serveur de la file Q_3 sert les clients avec un *patron de service alterné* (ASP - *Alternate Service Pattern*), *i.e.*, la probabilité de service de la file Q_3 change en fonction de P différents patrons de service ($\rho_{\mu_{31}}, \ldots, \rho_{\mu_{3P}}$). Quand un client est servi par la file Q_3, cette file peut changer son patron de service. De cette façon, quand un client est servi par le patron de service P_i, ce patron de service peut rester pour le client suivant avec une probabilité π_{ii}, ou il peut alterner pour un autre patron de service P_j avec une probabilité π_{ij} (pour tous les patrons de service P_i : $\sum_{j=1}^{P} \pi_{ij} = 1$).

Dans FIG. 8.2, on présente le modèle SAN à temps discret pour cet exemple avec un nombre de patrons de service $P = 2$. Dans ce modèle, chaque file Q_i est représentée par un automate $\mathcal{A}^{(i)}$, tel que $i \in [1..4]$. On représente par l'automate $\mathcal{A}^{(5)}$ le patron de service courant de la file Q_3. Les événements e_1 et e_2 représentent les arrivées de clients de l'extérieur vers les files Q_1 et Q_2 respectivement, et l'événement e_4 représente le départ de clients de Q_4 vers l'extérieur. Les événements e_{13} et e_{34} représentent le transfert de clients de Q_1 vers Q_3 et Q_3 vers Q_4 respectivement, et l'événement e_{23} représente le transfert de clients de Q_2 vers Q_3, et aussi le départ de clients de Q_2 vers l'extérieur (*perte*), lorsque la file Q_3 est pleine.

On remarque que l'événement e_{34} dans FIG. 8.2 se produit avec une probabilité exprimée par la fonction f_{34}, qui possède comme arguments les états de l'automate $\mathcal{A}^{(5)}$.

Pour un modèle SAN à temps discret, il faut déterminer l'ordre de priorité des événements du modèle afin de définir le comportement à suivre en cas de conflit de l'occurrence simultanée des événements. Dans cet exemple, on a déterminé que les événements qui amènent les clients vers l'extérieur du système sont les plus prioritaires. Dans ce cas, les événements de la file Q_4 sont plus prioritaires que les événements de la file Q_3 qui sont plus prioritaires que les événements des files Q_1 et Q_2, *i.e.*, l'ordre des événements du plus prioritaire au moins prioritaire est : e_4, e_{34}, e_{13}, e_{23}, e_1 et e_2.

De plus, une file d'attente à temps discret permet le départ et l'arrivée d'un client dans la même unité de temps. Pour cette raison, on modélise une transition du dernier état de l'automate $\mathcal{A}^{(i)}$ où $i \in [1..4]$



Type	Even.	Prob.	Priorité
loc	e_4	ρ_{μ_4}	1
syn	e_{34}	f_{34}	2
syn	e_{13}	ρ_{μ_1}	3
syn	e_{23}	ρ_{μ_2}	4
loc	e_1	ρ_{λ_1}	5
loc	e_2	ρ_{λ_2}	6

$$f_{34} = ((st(\mathcal{A}^{(5)}) == P_1^{(5)}) \times \rho_{\mu_{31}}) + ((st(\mathcal{A}^{(5)}) == P_2^{(5)}) \times \rho_{\mu_{32}})$$

FIG. 8.2 – ASP - modèle SAN à temps discret

(*i.e.*, l'automate qui représente la file Q_i) vers l'état fantôme Φ. Par exemple, pour la file Q_1, on a une transition de $K_1^{(1)}$ vers l'état Φ par l'occurrence de l'événement e_1. Cette modélisation permet le départ d'un client (*i.e.*, le changement de l'état $K_i^{(i)}$ vers l'état $K_i^{(i)} - 1$) et l'arrivée d'un autre client (*i.e.*, le changement de l'état $K_i^{(i)} - 1$ vers l'état $K_i^{(i)}$) dans la file Q_i dans la même unité de temps.

Par exemple, dans ce modèle, imaginons que chaque file Q_i possède un client (*i.e.*, les automates $\mathcal{A}^{(i)}$ sont dans l'état $1^{(i)}$ tel que $i \in [1..4]$). Dans une même unité de temps, selon l'ordre de priorité donné, les événements e_4, e_{34}, e_{13}, $\bar{e}_{23}$, e_1 et $\bar{e}_2$ peuvent avoir lieu, ainsi que les événements e_4, e_{34}, $\bar{e}_{13}$, e_{23}, $\bar{e}_1$ et e_2, ou bien les événements $\bar{e}_4$, $\bar{e}_{34}$, $\bar{e}_{13}$, $\bar{e}_{23}$, $\bar{e}_1$ et $\bar{e}_2$.

Cette combinatoire possible d'événements dans une même unité de temps dans un système en temps discret rend plus complexe le calcul effectif de la formulation matricielle de la chaîne de Markov sous-jacente.

8.3 Atelier avec kanbans

Le modèle présenté dans cette section est un exemple d'un atelier avec quatre postes de travail où des *kanbans* (*tickets*) sont utilisés pour contrôler le flux de pièces. Le flux de pièces dans cet atelier est le suivant : une pièce entre dans le poste de travail 1, après elle est divisée en deux autres pièces qui entrent dans les postes de travail 2 et 3, et ensuite ces deux pièces sont à nouveau assemblées en une seule pièce qui entre dans le poste de travail 4, d'où la pièce sort de l'atelier. Une pièce peut seulement entrer dans un poste de travail s'il y a un *kanban* disponible dans ce poste et, après qu'elle est traitée, elle est examinée

afin de savoir si elle doit être retraitée par ce poste. Notamment, une pièce peut seulement sortir du poste de travail 1 s'il y a un *kanban* dans les postes de travail 2 et 3, et une pièce peut seulement sortir du poste de travail 2 (poste de travail 3) s'il y a aussi une pièce prête dans le poste de travail 3 (poste de travail 2) et un *kanban* dans le poste de travail 4.

On présente dans FIG. 8.3 le modèle SAN à temps discret pour cet exemple. Dans FIG. 8.3, chaque poste de travail i (où $i \in [1..4]$) de cet atelier est représenté par trois automates :

– Pm_i : indique le nombre de pièces qui sont traitées par le poste de travail i simultanément ;

– $Pback_i$: indique le nombre de pièces qui doivent être retraitées par le poste de travail i ;

– $Pout_i$: indique le nombre de pièces prêtes dans le poste de travail i.

Chaque automate d'un poste de travail possède $N+1$ états énumérés de 0 à N, où N est le nombre de *kanbans* (*i.e.*, la *capacité*) dans le poste de travail. Dans le modèle dans FIG. 8.3, les événements in, s_{123} et s_{234} se produisent avec des probabilités exprimées par les fonctions f_{in}, $f_{s_{123}}$ et $f_{s_{234}}$ respectivement. Ces fonctions sont utilisées pour assurer que le nombre de *kanbans* dans chaque poste de travail n'est pas supérieur à N. Par exemple, la fonction f_{in} possède comme arguments les états des automates Pm_1, $Pback_1$ et $Pout_1$, et pour que l'événement in puisse avoir lieu avec une probabilité ρ_{in}, la somme des indices des états des automates Pm_1, $Pback_1$ et $Pout_1$ ne doit pas être supérieure à N.

L'état initial du système pour le modèle présenté dans FIG. 8.3 est : pour tout $i \in [1..4]$, les automates Pm_i, $Pback_i$ et $Pout_i$ sont dans l'état 0.

Pour ce modèle, on a déterminé que les événements du poste de travail qui amène les pièces vers l'extérieur de l'atelier sont les plus prioritaires, *i.e.*, on a précisé que le départ de pièces de l'atelier est plus prioritaire que l'arrivée de pièces. Dans ce cas, les événements du poste de travail 4 sont plus prioritaires que les événements des postes de travail 3 et 2 qui sont plus prioritaires que les événements du poste de travail 1. On peut observer dans le tableau dans FIG. 8.3 l'ordre de priorité de tous événements de ce modèle du plus prioritaire au moins prioritaire : out, ok_4, b_4, r_4, s_{234}, ok_3, b_3, r_3, ok_2, b_2, r_2, s_{123}, ok_1, b_1, r_1 et in.

Dans ce modèle, par exemple, on peut avoir l'occurrence simultanée des événements out, ok_4, $\bar{b}_4$, s_{234}, ok_3, $\bar{b}_3$, ok_2, $\bar{b}_2$, s_{123}, ok_1, $\bar{b}_1$, et in dans la même unité de temps. Cette occurrence simultanée des événements représente :

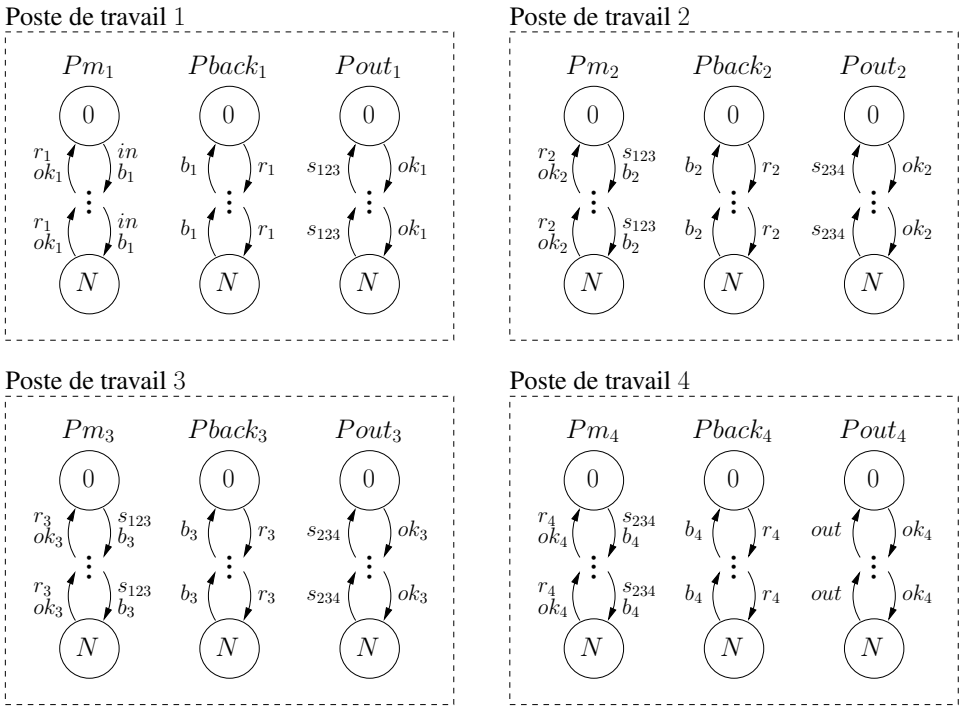
1. une pièce sort de l'atelier, *i.e.*, une pièce sort du poste de travail 4 (out) ;
2. une pièce est prête dans le poste de travail 4 (ok_4) ;
3. aucune pièce ne doit être retraitée par le poste de travail 4 ($\bar{b}_4$) ;
4. une pièce du poste de travail 2 et une autre pièce du poste de travail 3 sont assemblées en une seule pièce qui entre dans le poste de travail 4 (s_{234}) ;
5. une pièce est prête dans le poste de travail 3 (ok_3) ;
6. aucune pièce doit être retraitée par le poste de travail 3 ($\bar{b}_3$) ;
7. une pièce est prête dans le poste de travail 2 (ok_2) ;
8. aucune pièce doit être retraitée par le poste de travail 2 ($\bar{b}_2$) ;
9. une pièce du poste de travail 1 est divisée en deux autres pièces qui entrent dans les postes de travail 2 et 3 (s_{123}) ;

10. une pièce est prête dans le poste de travail 1 (ok_1) ;

11. aucune pièce doit être retraitée par le poste de travail 1 ($\bar{b}_1$) ;

12. une pièce entre dans l'atelier, *i.e.*, une pièce entre dans le poste de travail 1 (in).

Dans cet exemple, on peut aussi constater la complexité de la combinatoire possible d'événements dans la même unité de temps.



Type	Even.	Prob.	Priorité	Type	Even.	Prob.	Priorité
loc	out	ρ_{out}	1	syn	ok_2	ρ_{ok_2}	9
syn	ok_4	ρ_{ok_4}	2	syn	b_2	ρ_{b_2}	10
syn	b_4	ρ_{b_4}	3	syn	r_2	ρ_{r_2}	11
syn	r_4	ρ_{r_4}	4	syn	s_{123}	$f_{s_{123}}$	12
syn	s_{234}	$f_{s_{234}}$	5	syn	ok_1	ρ_{ok_1}	13
syn	ok_3	ρ_{ok_3}	6	syn	b_1	ρ_{b_1}	14
syn	b_3	ρ_{b_3}	7	syn	r_1	ρ_{r_1}	15
syn	r_3	ρ_{r_3}	8	loc	in	f_{in}	16

$$f_{in} = ((st(Pm_1) + st(Pback_1) + st(Pout_1)) < N) \times \rho_{in}$$

$$f_{s_{234}} = ((st(Pm_4) + st(Pback_4) + st(Pout_4)) < N) \times \rho_{s_{234}}$$

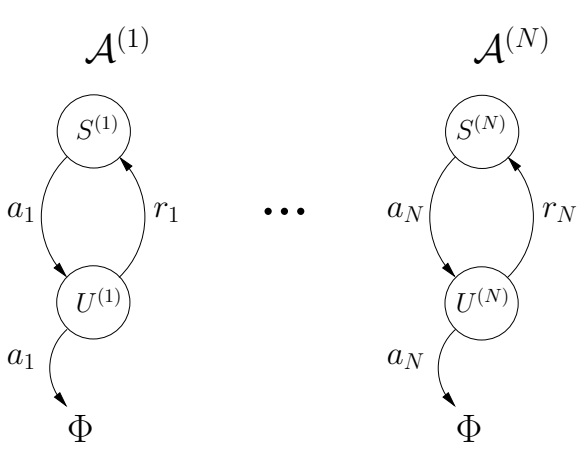
$$f_{s_{123}} = (((st(Pm_2) + st(Pback_2) + st(Pout_2)) < N) \;\&\&\; ((st(Pm_3) + st(Pback_3) + st(Pout_3)) < N)) \times \rho_{s_{123}}$$

FIG. 8.3 – Atelier avec kanbans - modèle SAN à temps discret

8.4 Partage de Ressources

Dans cette section, on présente un modèle SAN à temps discret d'un système de *partage de ressources* (RS - *Resource Sharing*).

Dans ce modèle, on a N clients distincts qui se partagent l'utilisation des R ressources communes identiques entre elles. On représente un client d'indice i par un automate $\mathcal{A}^{(i)}$ (où $i \in [1..N]$), qui possède deux états : $S^{(i)}$ (*sleeping*) état de *repos* ; et $U^{(i)}$ (*using*) état *actif*. Chaque automate $\mathcal{A}^{(i)}$ possède deux événements : a_i, l'événement de prise de ressource ; et r_i, l'événement de relâche de ressource. Chaque événement a_i se produit avec une probabilité exprimée par une fonction f_{u_i} qui contrôle le nombre de ressources utilisées actuellement. L'événement a_i se produit avec une probabilité ρ_{a_i} seulement s'il y a des ressources disponibles, *i.e.*, si $nb\ [\mathcal{A}^{(1)}..\mathcal{A}^{(N)}]\ U < R$. Dans FIG. 8.4, on présente le modèle SAN à temps discret de cet exemple.



Type	Even.	Prob.	Priorité
loc	r_1	ρ_{r_1}	1
loc	r_2	ρ_{r_2}	2
⋮	⋮	⋮	⋮
loc	r_N	ρ_{r_N}	N
loc	a_1	f_{u_1}	N + 1
loc	a_2	f_{u_2}	N + 2
⋮	⋮	⋮	⋮
loc	a_N	f_{u_N}	N + N

$$f_{u_i} = (nb\ [\mathcal{A}^{(1)}..\mathcal{A}^{(N)}]\ U < R) \times \rho_{a_i}$$

FIG. 8.4 – RS - modèle SAN à temps discret

Pour ce modèle SAN, on a déterminé que le relâchement d'une ressource par un client est plus prioritaire que la prise d'une ressource. Dans ce cas, l'événement r_i est plus prioritaire que l'événement a_i dans l'automate $\mathcal{A}^{(i)}$. D'ailleurs, on a modélisé que le premier client (l'automate $\mathcal{A}^{(1)}$) est plus prioritaire que le deuxième client (l'automate $\mathcal{A}^{(2)}$) et ainsi de suite jusqu'au dernier client (l'automate $\mathcal{A}^{(N)}$).

Dans cet exemple, on a modélisé qu'un client peut prendre une ressource dans la même unité de temps qu'il a relâché une autre ressource. Cette modélisation est possible grâce à la transition de l'état $U^{(i)}$ vers l'état fantôme Φ par l'occurrence de l'événement a_i dans l'automate $\mathcal{A}^{(i)}$. Dans ce cas, dans une même unité de temps, un client représenté par l'automate $\mathcal{A}^{(i)}$ peut passer de l'état $U^{(i)}$ vers l'état $S^{(i)}$ (par l'occurrence de l'événement r_i) et revenir à l'état $U^{(i)}$ par l'occurrence de l'événement a_i.

On a aussi modélisé le fait qu'un client puisse prendre une ressource dans la même unité de temps qu'un autre client (*e.g.*, le premier et le deuxième client peuvent prendre une ressource dans la même unité de temps par l'occurrence des événements a_1 et a_2).

De plus, il est possible que dans une même unité de temps plusieurs clients puissent relâcher une ressource et en prendre encore une autre simultanément. Par exemple, le premier et le deuxième client

peuvent relâcher une ressource chacun (par l'occurrence simultanée des événements r_1 et r_2) et, dans la même unité de temps, ils peuvent aussi prendre une autre ressource (par l'occurrence simultanée des événements a_1 et a_2).

On voit ainsi la complexité du système modélisé par ce modèle SAN dont le graphe est relativement simple.

8.5 Conclusion

On a présenté dans ce chapitre différents exemples de modélisation utilisant le formalisme des Réseaux d'Automates Stochastiques (SAN) à temps discret présenté dans le chapitre 7. Ces exemples ont illustrés une grande variété des concepts de modélisation du formalisme SAN à temps discret. En se basant dans ces exemples, nous illustrons des différents types d'interaction entre automates et nous présentons les principales caractéristiques de modélisation du formalisme.

En fait, la grande différence entre élaborer un modèle d'un exemple utilisant le formalisme SAN à temps discret et le formalisme SAN à temps continu est de déterminer les cas possibles de conflit de l'occurrence simultanée des événements dans le formalisme SAN à temps discret, *i.e.*, les cas où les événements peuvent avoir lieu dans une même unité de temps et amènent à un conflit. Le choix de l'ordre de priorité des événements a alors un rôle important dans la modélisation d'un exemple utilisant les SAN à temps discret, car cet ordre de priorité est primordial dans la description de la dynamique du modèle.

Il faut garder à l'esprit que les SAN à temps discret construisent non seulement des espaces d'états complexes - comme les SAN à temps continu - mais aussi des transitions très complexes (combinatoire possible des événements dans une même unité de temps) entre ces états.

Chapitre 9

Algèbre Tensorielle Complexe (ATX)

Dans ce chapitre, on va introduire l'*Algèbre Tensorielle compleXe* (ATX). Notre objectif est de présenter les notations et propriétés de l'ATX afin de proposer une formule tensorielle (voir Chapitre 10) pour le descripteur d'un Réseau d'Automates Stochastiques (SAN) à temps discret.

L'Algèbre Tensorielle Complexe est définie par un opérateur matriciel nommé *produit tensoriel complexe*.

On remarque que les éléments et les couples d'éléments dans le contexte de ce chapitre sont inspirés des notions d'événements et tuples de transitions définis pour les Réseaux d'Automates Stochastiques (SAN) à temps discret (Chapitre 7). Par la suite nous ferons les identifications nécessaires.

Dans ce qui suit, on va donner les notations et propriétés des opérateurs relatifs aux éléments de matrice d'un produit tensoriel complexe (Section 9.1). Dans la section 9.2, on étend les définitions de ces opérateurs et, finalement, dans la section 9.3, on présente[1] les définitions et propriétés du *produit tensoriel complexe* de matrices.

9.1 Opérateurs

Dans cette section, on présente les opérateurs relatifs aux éléments des matrices d'un produit tensoriel complexe. Typiquement, ces éléments sont les événements d'un modèle SAN, et dans la suite on les appellera aussi événements.

Définition 9.1.1. *Soit un ensemble $\breve{\mathcal{E}}$ composé de deux types d'événements :* ***actifs****, notés $e \in \mathcal{E}$, et à tout événement actif est associé un événement* ***complémentaire****, noté $\bar{e} \in \bar{\mathcal{E}}$, où $\breve{\mathcal{E}} = \mathcal{E} \cup \bar{\mathcal{E}}$.*

On remarque que les événements *actifs* sont tous les événements e de l'ensemble d'*événements* $\mathcal{E}$ d'un modèle SAN et les événements *complémentaires* sont tous les *événements complémentaires* $\bar{e}$ de l'ensemble d'événements $\bar{\mathcal{E}}$.

[1] Les définitions et extensions des opérateurs présentées dans ce chapitre sont communes dans cette thèse et dans [15].

On va introduire les définitions de deux événements particuliers de $\breve{\mathcal{E}}$.

Définition 9.1.2. *On définit un événement* ***nul****, noté* $O_{\breve{\mathcal{E}}}$.

Définition 9.1.3. *On définit un événement* ***neutre****, noté* $I_{\breve{\mathcal{E}}}$.

À tout événement *actif* et *complémentaire*, on donne une notion de priorité. Cette notion de priorité adoptée pour les événements actifs e complémentaires est la même que celle employée pour les événements d'un modèle SAN.

Définition 9.1.4. *Soit un événement* $e \in \mathcal{E}$*, on définit une priorité* $\varrho_e \in [1..+\infty[$ *associée à cet événement telle que : quelque soit* $e_1, e_2 \in \mathcal{E}$*,* $e_1 \neq e_2 \Rightarrow \varrho_{e_1} \neq \varrho_{e_2}$.

Définition 9.1.5. *Un événement* $\bar{e} \in \bar{\mathcal{E}}$ *possède la même priorité* ϱ_e *que son événement actif correspondant* $e \in \mathcal{E}$*, i.e., la priorité* $\varrho_{\bar{e}} = \varrho_e$.

Dans la suite, on définit trois opérateurs "$\cdot$", "$+$" et "$*$" sur l'ensemble d'événements $\breve{\mathcal{E}}$. On note $\mathcal{E}^+$ l'ensemble obtenu par toutes les expressions légales de ces opérateurs. L'ensemble $\mathcal{E}^+$ est l'ensemble sur lequel opère ces trois opérateurs.

Dans les sections suivantes, on présente les notations et propriétés de ces opérateurs qui sont attribuées aux événements d'ensemble $\breve{\mathcal{E}}$.

9.1.1 Opérateur de simultanéité

Définition 9.1.6. *On définit sur l'ensemble* $\breve{\mathcal{E}}$ *d'événements un opérateur, noté "*$\cdot$*". On appelle cet opérateur de* ***produit de simultanéité*** *et on le définit par :*

1. *Soit* $e_1 \in \breve{\mathcal{E}}, e_2 \in \breve{\mathcal{E}}$*,* $e_1 \cdot e_2$ *est* irréductible *et n'a un sens que si* $\varrho_{e_1} < \varrho_{e_2}$*. avec cette définition,* $e_1 \cdot e_1$ *n'a pas de sens, et* $e_1 \cdot \bar{e}_1$ *n'a pas de sens non plus ;*
2. *Pour l'événement nul :*
   $$\forall e \in \breve{\mathcal{E}}, \qquad e \cdot O_{\breve{\mathcal{E}}} = O_{\breve{\mathcal{E}}}$$
3. *Pour l'événement neutre :*
   $$\forall e \in \breve{\mathcal{E}}, \qquad e \cdot I_{\breve{\mathcal{E}}} = e$$
4. *L'opérateur de simultanéité est un opérateur* **associatif**. *Quels que soient les événements* $e_1, e_2, e_3 \in \breve{\mathcal{E}}$ *tel que* $\varrho_{e_1} < \varrho_{e_2} < \varrho_{e_3}$*, alors*
   $$e_1 \cdot (e_2 \cdot e_3) = (e_1 \cdot e_2) \cdot e_3$$

Dans un modèle SAN, le produit de simultanéité est la représentation de l'occurrence simultanée de plusieurs événements dans une seule unité de temps. $e_1 \cdot e_1$ n'a pas de sens, car l'événement e_1 ne peut pas avoir lieu simultanément avec lui même. De même, $e_1 \cdot \bar{e}_1$ n'a pas de sens, car l'événement e_1 et son événement complémentaire $\bar{e}_1$ ne peuvent pas avoir lieu simultanément non plus.

Définition 9.1.7. *Soient s événements distincts de $\breve{\mathcal{E}}$, on écrit la forme itérée de l'opérateur de simultanéité, appelé* ***combinaison simultanée*** *(cs), par la notation suivante :*

$$cs = \dot{\sum_{i=1}^{s}} e_i = e_1 \cdot e_2 \cdot \ldots \cdot e_s, \tag{9.1}$$

où les priorités sont telles que $\varrho_{e_1} < \varrho_{e_2} < \ldots < \varrho_{e_s}$. On note $\breve{\mathcal{E}}_{cs} \subseteq \breve{\mathcal{E}}$ l'ensemble de facteurs qui apparaissent dans la combinaison simultanée cs.

Définition 9.1.8. *Soit cs la combinaison simultanée $cs = \dot{\sum_{i=1}^{s}} e_i = e_1 \cdot e_2 \cdot \ldots \cdot e_s$, on définit cs^{-e_j} la combinaison simultanée cs où l'événement e_j est remplacé par l'événement neutre $I_{\breve{\mathcal{E}}}$, i.e., $cs^{-e_j} = e_1 \cdot \ldots \cdot e_{j-1} \cdot I_{\breve{\mathcal{E}}} \cdot e_{j+1} \cdot \ldots \cdot e_s$.*

9.1.2 Opérateur de choix

Définition 9.1.9. *On définit sur l'ensemble $\breve{\mathcal{E}}$ un opérateur, noté "$+$". On appelle cet opérateur de* ***somme de choix*** *et on le définit par :*

1. $\forall e_1, e_2, \qquad e_1 + e_2$ *a un sens et est irréductible ;*
2. *Pour l'événement nul :*
   $\forall e \in \breve{\mathcal{E}}, \qquad e + O_{\breve{\mathcal{E}}} = e$
3. *Pour l'événement neutre :*
   $\forall e \in \breve{\mathcal{E}}, \qquad e + I_{\breve{\mathcal{E}}}$ *est irréductible*
4. *L'opérateur de choix est un opérateur associatif et quels que soient les événements $e_1, e_2, e_3 \in \breve{\mathcal{E}}$, on a :*
   $e_1 + (e_2 + e_3) = (e_1 + e_2) + e_3$
5. *L'opérateur de choix est un opérateur commutatif et quels que soient les événements $e_1, e_2 \in \breve{\mathcal{E}}$, on a :*
   $e_1 + e_2 = e_2 + e_1$

Dans un modèle SAN, la somme de choix est la représentation de l'occurrence de plusieurs événements qui peuvent déclencher une même transition. Par exemple, soit une transition de l'état x vers l'état y dans un modèle SAN, transition qui peut être déclenchée par l'événement e_1 **ou** par l'événement e_2. Ceci en est représentée par $e_1 + e_2$.

Définition 9.1.10. *Soient c événements de $\breve{\mathcal{E}}$, on écrit la forme itérée de l'opérateur de choix avec la notation suivante :*

$$\dot{\sum_{i=1}^{c}} e_i = e_1 + e_2 + \ldots + e_c \tag{9.2}$$

9.1.3 Opérateur de concurrence

Définition 9.1.11. *On définit sur l'ensemble $\breve{\mathcal{E}}$ un opérateur, noté "$*$". On appelle cet opérateur de* ***produit de concurrence*** *et on le définit par :*

1. $\forall e_1, e_2 \in \breve{\mathcal{E}}, \quad e_1 * e_2$ *existe et si* $e_1 \neq e_2$*, alors* $e_1 * e_2$ *est* irréductible *;*
2. *Pour l'événement nul :*
   $\forall e \in \breve{\mathcal{E}}, \quad e * O_{\breve{\mathcal{E}}} = O_{\breve{\mathcal{E}}}$
3. *Pour l'événement neutre :*
   $\forall e \in \breve{\mathcal{E}}, \quad e * I_{\breve{\mathcal{E}}} = e$
4. *Soit un événement $e \in \breve{\mathcal{E}}$, on associe un degré d'idempotence entier, noté $\eta_e \in [1.. + \infty[$, tel que l'on ait les propriétés suivantes :*

$$\forall e \in \mathcal{E}, \forall i, 1 \leq i < \eta_e \quad \underbrace{e * e * \ldots * e}_{i \text{ fois}} = O_{\breve{\mathcal{E}}} \tag{9.3}$$

$$\forall e \in \bar{\mathcal{E}}, \forall i, 1 \leq i < \eta_e \quad \underbrace{e * e * \ldots * e}_{i \text{ fois}} = I_{\breve{\mathcal{E}}} \tag{9.4}$$

$$\forall e \in \breve{\mathcal{E}}, i = \eta_e \quad \underbrace{e * e * \ldots * e}_{i \text{ fois}} = e \tag{9.5}$$

$$\forall i, i > \eta_e \quad \underbrace{e * e * \ldots * e}_{i \text{ fois}} \quad \textit{n'a pas de sens} \tag{9.6}$$

5. *L'opérateur de concurrence est un opérateur associatif et quels que soient les événements $e_1, e_2, e_3 \in \breve{\mathcal{E}}$,*
   $e_1 * (e_2 * e_3) = (e_1 * e_2) * e_3$
6. *L'opérateur de concurrence est un opérateur commutatif et quels que soient les événements $e_1, e_2 \in \breve{\mathcal{E}}$,*
   $e_1 * e_2 = e_2 * e_1$
7. *L'opérateur de* concurrence *"$*$" est distributif sur l'opérateur de* choix *"$+$" et quels que soient les événements $e_1, e_2, e_3 \in \breve{\mathcal{E}}$,*
   $e_1 * (e_2 + e_3) = (e_1 * e_2) + (e_1 * e_3)$

Dans un modèle SAN, le produit de concurrence est la représentation de l'occurrence dans la même unité de temps d'événements dans différents automates. Par exemple, $e_1 * e_2$ représente l'occurrence de l'événement e_1 dans un automate **et** l'occurrence de l'événement e_2 dans un autre automate dans la même unité de temps.

Dans la propriété (4) de la définition 9.1.11 sont présentées les propriétés d'un produit de concurrence relatif au degré d'idempotence d'un événement. Le degré d'idempotence sera mis en relation avec le nombre d'automates concernés par un événement dans un modèle SAN. Pour un événement synchronisant concernant η_e automates, celui-ci ne pourra avoir lieu que si $\underbrace{e * e * \ldots * e}_{\eta_e \text{ fois}}$.

Définition 9.1.12. *Soient p événements de $\breve{\mathcal{E}}$, on écrit la forme itérée de l'opérateur de concurrence avec la notation suivante :*

$$\overset{p}{\underset{i=1}{*}} e_i = e_1 * e_2 * \ldots * e_p \tag{9.7}$$

Dans un modèle SAN à temps discret, où plusieurs événements peuvent avoir lieu simultanément dans un même automate, les facteurs d'un produit de concurrence peuvent être des combinaisons simultanées au lieu de "simples" événements de $\breve{\mathcal{E}}$.

Définition 9.1.13. *Soient p combinaisons simultanées $(cs_i)_{i=1}^{p}$, on appelle cc la* ***combinaison concurrente*** *des $(cs_i)_{i=1}^{p}$:*

$$cc = \overset{p}{\underset{i=1}{*}} cs_i = cs_1 * cs_2 * \ldots * cs_p \tag{9.8}$$

Les facteurs d'une combinaison concurrente sont des combinaisons simultanées qui représentent le produit de simultanéité d'événements dans un même automate. On remarque qu'un événement $e_1 \in \breve{\mathcal{E}}$ peut être interprété comme une combinaison simultanée cs d'un seul événement ($s = 1$), *i.e.*, $\breve{\mathcal{E}}_{cs} = \{e_1\}$.

Définition 9.1.14. *Soit un événement $e \in \breve{\mathcal{E}}$ et une combinaison concurrente cc, on définit $O_e(cc)$ le nombre de fois que l'événement e apparaît dans les combinaisons simultanées (cs) de cc.*

Dans le contexte des SAN, soit une combinaison concurrente et un événement, on cherche le nombre de fois que l'identificateur de cet événement apparaît dans les combinaisons simultanées de la combinaison concurrente. On remarque que e apparaît au plus une fois dans chaque cc.

Propriété 9.1.1. *Soit un événement $e \in \breve{\mathcal{E}}$ de degré d'idempotence η_e, une combinaison concurrente $cc = cs_1 * \ldots * cs_p$ avec p combinaisons simultanées et un $O_e(cc)$ donné, alors le produit de concurrence "$*$" a les propriétés suivantes :*

1. $$cc = \underbrace{cs_1 * \ldots * cs_p}_{O_e(cc)=\eta_e} = e \cdot (cs'_1 * \ldots * cs'_p) \qquad cs'_i = \begin{cases} cs_i & \text{si } e \notin \breve{\mathcal{E}}_{cs_i} \\ cs_i^{-e} & \text{si } e \in \breve{\mathcal{E}}_{cs_i} \end{cases} \tag{9.9}$$

2. $$cc = \underbrace{cs_1 * \ldots * cs_p}_{O_e(cc)<\eta_e} = cs'_1 * \ldots * cs'_p \qquad \text{si } e \in \bar{\mathcal{E}} \text{ et } cs'_i = \begin{cases} cs_i & \text{si } e \notin \breve{\mathcal{E}}_{cs_i} \\ cs_i^{-e} & \text{si } e \in \breve{\mathcal{E}}_{cs_i} \end{cases} \tag{9.10}$$

3. $$cc = \underbrace{cs_1 * \ldots * cs_p}_{O_e(cc)<\eta_e} = O_{\breve{\mathcal{E}}} \qquad \text{si } e \in \mathcal{E} \tag{9.11}$$

La propriété 9.1.1 est une extension de la propriété (4) de la définition de l'opérateur "$*$" (Définition 9.1.11), où les facteurs d'un produit de concurrence sont des combinaisons simultanées au lieu de simples événements de $\breve{\mathcal{E}}$.

Dans le contexte des SAN, un événement est réalisable uniquement s'il est réalisable au même instant dans **tous les automates concernés** par cet événement. On rappelle que l'ensemble d'indices d'automates concernés par un événement e est noté par $\mathcal{O}_e$ et $|\mathcal{O}_e| = \eta_e$. Avec la vision de cet aspect positionnel des événements d'un modèle SAN à temps discret, le produit de concurrence "$*$" possède la propriété suivante.

Propriété 9.1.2. *Soit un événement $e \in \breve{\mathcal{E}}$, une combinaison concurrente $cc = cs_1 * \ldots * cs_N$ avec N combinaisons simultanées, où une combinaison simultanée cs_i représente l'occurrence simultanée d'événements dans l'automate complété $\breve{\mathcal{A}}^{(i)}$, et l'ensemble d'indices d'automates concernés $\mathcal{O}_e$ par cet événement e, alors le produit de concurrence "$*$" a les propriétés :*

• *Si pour tout* cs_i*, où* $i \in \mathcal{O}_e$*,* $e \in \breve{\mathcal{E}}_{cs_i}$ *alors*

$$cc = cs_1 * \ldots * cs_N = e \cdot (cs'_1 * \ldots * cs'_N) \qquad cs'_i = \begin{cases} cs_i & \text{si } e \notin \breve{\mathcal{E}}_{cs_i} \\ cs_i^{-e} & \text{si } e \in \breve{\mathcal{E}}_{cs_i} \end{cases} \tag{9.12}$$

• *S'il existe* cs_i*, où* $i \in \mathcal{O}_e$*, tel que* $e \notin \breve{\mathcal{E}}_{cs_i}$ *alors*

$$\text{si } e \in \bar{\mathcal{E}},\ cc = cs_1 * \ldots * cs_N = cs'_1 * \ldots * cs'_N \qquad cs'_i = \begin{cases} cs_i & \text{si } e \notin \breve{\mathcal{E}}_{cs_i} \\ cs_i^{-e} & \text{si } e \in \breve{\mathcal{E}}_{cs_i} \end{cases} \tag{9.13}$$

$$\text{si } e \in \mathcal{E},\ cc = cs_1 * \ldots * cs_N = O_{\breve{\mathcal{E}}} \tag{9.14}$$

La propriété 9.1.2 est un cas particulier de la propriété 9.1.1, où les combinaisons simultanées cs d'une combinaison concurrente cc possède une information positionnelle en fonction des indices d'automates d'un modèle SAN.

Cette vision permet d'obtenir une optimisation algorithmique. Par exemple, si on évalue la combinaison concurrente cc de gauche à droite, dès le premier indice $i \in \mathcal{O}_e$, tel que $e \notin \breve{\mathcal{E}}_{cs_i}$, alors on constate par la propriété 9.1.2 que cc est évaluée à $O_{\breve{\mathcal{E}}}$. Cette constatation est présentée par le corollaire suivant.

Corollaire 9.1.1. *Soit un événement* $e \in \mathcal{E}$ *et une combinaison concurrente* $cc = (cc_A) * (cc_B)$*, où* cc_A *et* cc_B *sont des combinaisons concurrentes tel que* $cc_A = cs_1 * \ldots * cs_p$ *et* $cc_B = cs_{p+1} * \ldots * cs_N$*,*

s'il existe $i \in \mathcal{O}_e$*, où* $i \leq p$*, tel que* $e \notin \breve{\mathcal{E}}_{cs_i}$*, alors*

l'évaluation de $cc_A = O_{\breve{\mathcal{E}}}$ *et par le produit de concurrence avec l'événement nul (propriété 2 de la définition 9.1.11) ce qui donne* $cc = O_{\breve{\mathcal{E}}}$*.*

Sachant qu'une combinaison simultanée cs_i représente l'occurrence simultanée d'événements dans l'automate complété $\breve{\mathcal{A}}^{(i)}$ d'un modèle SAN à temps discret (où $i \in [1..N]$) et l'opérateur de concurrence est un opérateur commutatif, on peut définir un "*attribut*" pour chaque combinaison simultanée cs dans une combinaison concurrente $cc = cs_1 * \ldots * cs_N$ afin de garder l'information relative à l'automate d'origine de cette combinaison simultanée.

Définition 9.1.15. *Soit un modèle SAN* $(\breve{\mathcal{E}}, \breve{\mathcal{A}}^{(i)}, \hat{\mathcal{S}}, \mathcal{F})$ *à temps discret (où* $i \in [1..N]$*) et une combinaison concurrente* cc *relative à l'occurrence globale d'événements du modèle SAN telle que* $cc = cs_1 * \ldots * cs_N$*, où* cs_i *représente l'occurrence simultanée d'événements dans l'automate complété* $\breve{\mathcal{A}}^{(i)}$*, on définit l'attribut de position "*a*" pour la combinaison simultanée* cs_i *tel que :*

$$a(cs_i) = i$$

étant i *l'indice de l'automate complété* $\breve{\mathcal{A}}^{(i)}$ *du modèle SAN (où* $i \in [1..N]$*).*

L'utilisation d'un attribut de position pour les combinaisons simultanées dans une combinaison concurrente a un rôle important dans la méthode de résolution d'un modèle SAN à temps discret qui sera présentée dans le chapitre 11.

9.2 Extension de ces opérateurs

Dans cette section, on va étendre les définitions et propriétés des opérateurs présentés dans la section précédente.

Les opérateurs présentés dans la section 9.1 sont appliqués aux éléments de l'ensemble $\breve{\mathcal{E}}$. Dans cette section, au lieu d'utiliser les éléments de $\breve{\mathcal{E}}$, on présente l'extension des opérateurs pour les *couples d'éléments*. Les couples d'éléments sont typiquement les couples de transitions d'un modèle SAN, comprenant un événement et une probabilité de routage. Désormais, on va utiliser *tuples de transitions* pour faire référence aux *couples d'éléments*.

Définition 9.2.1. *Soit $\breve{\mathcal{T}}$ l'ensemble des tuples de transitions $(e, \pi_e(x, y))$, où $e \in \breve{\mathcal{E}}$ et $\pi_e(x, y) \in [0, 1]$ est une probabilité, fonction de x et y, où x et y appartiennent à un espace $\mathcal{S}$.*

Un tuple de transition $(e, \pi_e(x, y))$ représente typiquement une transition de l'état x vers l'état y déclenchée par l'événement e avec la probabilité de routage $\pi_e(x, y)$ dans un automate d'un modèle SAN, x et y appartenant à l'espace d'état $\mathcal{S}$ de l'automate. Il est intéressant de remarquer que la probabilité de routage $\pi_e(x, y)$, ainsi que la probabilité d'occurrence ρ_e de l'événement e peuvent être exprimées par des *fonctions* définies dans l'espace (ou sous-espace) d'états du modèle SAN. Dans ce chapitre, on se restreint à la présentation des définitions et propriétés des opérateurs de l'algèbre tensorielle complexe. Le traitement de fonctions des probabilités d'occurrence et probabilités de routage des événements est reprise au chapitre 11 lors de la résolution du modèle.

Notons qu'un même événement $e \in \breve{\mathcal{E}}$ peut être associé à plusieurs tuples de transitions de $\breve{\mathcal{T}}$, *i.e.*, soit un événement $e \in \breve{\mathcal{E}}$, $(e, \pi_e(x, y)) \in \breve{\mathcal{T}}$ et $(e, \pi_e(z, w)) \in \breve{\mathcal{T}}$ sont deux tuples de transitions distincts de l'événement e si $x \neq z$ ou $y \neq w$.

Définition 9.2.2. *On définit un tuple de transition **nul** $(O_{\breve{\mathcal{E}}}, 0)$, noté $O_{\breve{\mathcal{T}}}$.*

Définition 9.2.3. *On définit un tuple de transition **neutre** $(I_{\breve{\mathcal{E}}}, 1)$, noté $I_{\breve{\mathcal{T}}}$.*

Dans la suite, on définit trois opérateurs "$\cdot$", "$+$" et "$*$" sur l'ensemble de tuples de transitions $\breve{\mathcal{T}}$. On note $\mathcal{T}^+$ l'ensemble obtenu par toutes les expressions légales de ces opérateurs. L'ensemble $\mathcal{T}^+$ est l'ensemble sur lequel opère ces trois opérateurs.

Dans les sections suivantes, on va présenter les notations et propriétés des opérateurs "$\cdot$", "$+$" et "$*$" pour les tuples de transitions de l'ensemble $\breve{\mathcal{T}}$. Ces notations et propriétés sont des extensions des notations et propriétés présentées dans les sections précédentes.

9.2.1 Extension de l'opérateur de simultanéité

Définition 9.2.4. *On définit sur l'ensemble de tuples de transitions $\breve{\mathcal{T}}$ l'opérateur de **simultanéité**, noté "$\cdot$". Cet opérateur est défini par :*

1. *Soit $(e_1, \pi_{e_1}(x_1, y_1)) \in \breve{\mathcal{T}}, (e_2, \pi_{e_2}(x_2, y_2)) \in \breve{\mathcal{T}}$,*
   $(e_1, \pi_{e_1}(x_1, y_1)) \cdot (e_2, \pi_{e_2}(x_2, y_2))$ est irréductible et n'a un sens que si $\varrho_{e_1} < \varrho_{e_2}$ et $y_1 = x_2$. On remarque que $(e_1, \pi_{e_1}(x_1, y_1)) \cdot (e_1, \pi_{e_1}(x_1, y_1))$ n'a pas de sens, et $(e_1, \pi_{e_1}(x_1, y_1)) \cdot (\bar{e}_1, \pi_{\bar{e}_1}(x_1, y_1))$ n'a pas de sens non plus ;

2. *Pour le tuple de transition nul :*
$$\forall(e,\pi_e(x,y)) \in \breve{\mathcal{T}}, \qquad (e,\pi_e(x,y)) \cdot O_{\breve{\mathcal{T}}} = O_{\breve{\mathcal{T}}}$$

3. *Pour le tuple de transition neutre :*
$$\forall(e,\pi_e(x,y)) \in \breve{\mathcal{T}}, \qquad (e,\pi_e(x,y)) \cdot I_{\breve{\mathcal{T}}} = (e,\pi_e(x,y))$$

4. *L'opérateur de simultanéité est un opérateur* **associatif**. *Quels que soient les tuples de transitions* $(e_1,\pi_{e_1}(x_1,y_1)),(e_2,\pi_{e_2}(x_2,y_2)),(e_3,\pi_{e_3}(x_3,y_3)) \in \breve{\mathcal{T}}$ *tel que* $\varrho_{e_1} < \varrho_{e_2} < \varrho_{e_3}$ *et* $y_1 = x_2$ *et* $y_2 = x_3$*, alors*
$$(e_1,\pi_{e_1}(x_1,y_1)) \cdot \Big[(e_2,\pi_{e_2}(x_2,y_2)) \cdot (e_3,\pi_{e_3}(x_3,y_3))\Big] =$$
$$\Big[(e_1,\pi_{e_1}(x_1,y_1)) \cdot (e_2,\pi_{e_2}(x_2,y_2))\Big] \cdot (e_3,\pi_{e_3}(x_3,y_3))$$

Définition 9.2.5. *Soient s tuples de transitions de* $\breve{\mathcal{T}}$*, on écrit la forme itérée de l'opérateur de simultanéité, appelé* ***combinaison simultanée de couples*** *(csc), avec la notation suivante :*

$$csc = \overset{s}{\underset{i=1}{\cdot}}(e_i,\pi_{e_i}(x_i,y_i)) = (e_1,\pi_{e_1}(x_1,y_1)) \cdot (e_2,\pi_{e_2}(x_2,y_2)) \cdot \ldots \cdot (e_s,\pi_{e_s}(x_s,y_s)), \tag{9.15}$$

où les priorités sont telle que $\varrho_{e_1} < \varrho_{e_2} < \ldots < \varrho_{e_s}$ *et* $\forall i \in [1..s-1], y_i = x_{i+1}$*. On note* $\breve{\mathcal{T}}_{csc} \subseteq \breve{\mathcal{T}}$ *l'ensemble des facteurs qui apparaissent dans csc et* $\breve{\mathcal{E}}_{csc} \subseteq \breve{\mathcal{E}}$ *l'ensemble des événements qui apparaissent dans csc.*

Définition 9.2.6. *Soit csc la combinaison simultanée de couples* $csc = \overset{s}{\underset{i=1}{\cdot}}(e_i,\pi_{e_i}(x_i,y_i))$ = $(e_1,\pi_{e_1}(x_1,y_1))$ $\cdot (e_2,\pi_{e_2}(x_2,y_2)) \cdot \ldots \cdot (e_s,\pi_{e_s}(x_s,y_s))$*, on définit* csc^{-e_j} *la combinaison simultanée de couples csc où le tuple de la transition* $(e_j,\pi_{e_j}(x_j,y_j))$ *est remplacé par le tuple de transition neutre* $I_{\breve{\mathcal{T}}}$*, i.e.,* $csc^{-e_j} = (e_1,\pi_{e_1}(x_1,y_1)) \cdot \ldots \cdot (e_{j-1},\pi_{e_{j-1}}(x_{j-1},y_{j-1})) \cdot I_{\breve{\mathcal{T}}} \cdot (e_{j+1},\pi_{e_{j+1}}(x_{j+1},y_{j+1})) \cdot \ldots \cdot (e_s,\pi_{e_s}(x_s,y_s))$.

9.2.2 Extension de l'opérateur de choix

Définition 9.2.7. *On définit sur l'ensemble de tuples de transitions* $\breve{\mathcal{T}}$ *l'opérateur de* ***choix****, noté "+". Cet opérateur est défini par :*

1. $\forall(e_1,\pi_{e_1}(x_1,y_1)),(e_2,\pi_{e_2}(x_2,y_2)) \in \breve{\mathcal{T}}$,
$(e_1,\pi_{e_1}(x_1,y_1)) + (e_2,\pi_{e_2}(x_2,y_2))$ *a un sens et est irréductible ;*
2. *Pour le tuple de transition nul :*
$$\forall(e,\pi_e(x,y)) \in \breve{\mathcal{T}}, \qquad (e,\pi_e(x,y)) + O_{\breve{\mathcal{T}}} = (e,\pi_e(x,y))$$
3. *Pour le tuple de transition neutre :*
$$\forall(e,\pi_e(x,y)) \in \breve{\mathcal{T}}, \qquad (e,\pi_e(x,y)) + I_{\breve{\mathcal{T}}} \text{ est irréductible}$$
4. *L'opérateur de choix est un opérateur* **associatif**. *Quels que soient les tuples de transitions* $(e_1,\pi_{e_1}(x_1,y_1))$, $(e_2,\pi_{e_2}(x_2,y_2)),(e_3,\pi_{e_3}(x_3,y_3)) \in \breve{\mathcal{T}}$*, alors*
$$(e_1,\pi_{e_1}(x_1,y_1)) + \Big[(e_2,\pi_{e_2}(x_2,y_2)) + (e_3,\pi_{e_3}(x_3,y_3))\Big] =$$
$$\Big[(e_1,\pi_{e_1}(x_1,y_1)) + (e_2,\pi_{e_2}(x_2,y_2))\Big] + (e_3,\pi_{e_3}(x_3,y_3))$$
5. *L'opérateur de choix est un opérateur* **commutatif**, *quels que soient les tuples de transitions* $(e_1,\pi_{e_1}(x_1,y_1)),(e_2,\pi_{e_2}(x_2,y_2)) \in \breve{\mathcal{T}}$,
$$(e_1,\pi_{e_1}(x_1,y_1)) + (e_2,\pi_{e_2}(x_2,y_2)) = (e_2,\pi_{e_2}(x_2,y_2)) + (e_1,\pi_{e_1}(x_1,y_1))$$

Définition 9.2.8. *Soient c tuples de transitions de $\breve{\mathcal{T}}$, on définit la forme itérée de l'opérateur de choix avec la notation suivante :*

$$\overset{c}{\underset{i=1}{+}}(e_i, \pi_{e_i}(x_i, y_i)) = (e_1, \pi_{e_1}(x_1, y_1)) + (e_2, \pi_{e_2}(x_2, y_2)) + \ldots + (e_c, \pi_{e_c}(x_c, y_c)) \tag{9.16}$$

9.2.3 Extension de l'opérateur de concurrence

Définition 9.2.9. *On définit sur l'ensemble de tuple de transitions $\breve{\mathcal{T}}$ l'opérateur de* ***concurrence****, noté "$*$". Cet opérateur est défini par :*

1. $\forall (e_1, \pi_{e_1}(x_1, y_1)), (e_2, \pi_{e_2}(x_2, y_2)) \in \breve{\mathcal{T}}, \qquad (e_1, \pi_{e_1}(x_1, y_1)) * (e_2, \pi_{e_2}(x_2, y_2))$ *existe et si* $e_1 \neq e_2$*, alors* $(e_1, \pi_{e_1}(x_1, y_1)) * (e_2, \pi_{e_2}(x_2, y_2))$ *est* irréductible *;*
2. *Pour le tuple de transition nul :*
   $\forall (e, \pi_e(x, y)) \in \breve{\mathcal{T}}, \qquad (e, \pi_e(x, y)) * O_{\breve{\mathcal{T}}} = O_{\breve{\mathcal{T}}}$
3. *Pour le tuple de transition neutre :*
   $\forall (e, \pi_e(x, y)) \in \breve{\mathcal{T}}, \qquad (e, \pi_e(x, y)) * I_{\breve{\mathcal{T}}} = (e, \pi_e(x, y))$
4. *Soit un événement $e \in \breve{\mathcal{E}}$ et n tuples de transitions $(e, \pi_e(x^{(j)}, y^{(j)})) \in \breve{\mathcal{T}}$ (où $j \in [1..n]$ et $x^{(j)}, y^{(j)}$ sont les paramètres de la probabilité du j-ème tuple de transition), on associe un degré d'idempotence entier, noté $\eta_e \in [1.. + \infty[$, tel que on ait les propriétés suivantes :*

$$\forall i, 1 \leq i < \eta_e \quad \underbrace{(e, \pi_e(x^{(1)}, y^{(1)})) * \ldots * (e, \pi_e(x^{(n)}, y^{(n)}))}_{i \text{ fois}} = O_{\breve{\mathcal{T}}} \qquad \text{si } e \in \mathcal{E} \tag{9.17}$$

$$\forall i, 1 \leq i < \eta_e \quad \underbrace{(e, \pi_e(x^{(1)}, y^{(1)})) * \ldots * (e, \pi_e(x^{(n)}, y^{(n)}))}_{i \text{ fois}} = I_{\breve{\mathcal{T}}} \qquad \text{si } e \in \bar{\mathcal{E}} \tag{9.18}$$

$$i = \eta_e \quad \underbrace{(e, \pi_e(x^{(1)}, y^{(1)})) * \ldots * (e, \pi_e(x^{(n)}, y^{(n)}))}_{i \text{ fois}} = (e, \overset{n}{\underset{j=1}{\amalg}} \pi_e(x^{(j)}, y^{(j)})) \tag{9.19}$$

$$\forall i > \eta_e \quad \underbrace{(e, \pi_(x^{(1)}, y^{(1)})) * \ldots * (e, \pi_e(x^{(n)}, y^{(n)}))}_{i \text{ fois}} \qquad \text{n'a pas de sens} \tag{9.20}$$

5. *L'opérateur de concurrence est un opérateur* **associatif**. *Quels que soient les tuples de transitions $(e_1, \pi_{e_1}(x_1, y_1)), (e_2, \pi_{e_2}(x_2, y_2)), (e_3, \pi_{e_3}(x_3, y_3)) \in \breve{\mathcal{T}}$, alors*
   $$(e_1, \pi_{e_1}(x_1, y_1)) * \Big[(e_2, \pi_{e_2}(x_2, y_2)) * (e_3, \pi_{e_3}(x_3, y_3))\Big] =$$
   $$\Big[(e_1, \pi_{e_1}(x_1, y_1)) * (e_2, \pi_{e_2}(x_2, y_2))\Big] * (e_3, \pi_{e_3}(x_3, y_3))$$
6. *L'opérateur de concurrence est un opérateur commutatif, quels que soient les tuples de transitions $(e_1, \pi_{e_1}(x_1, y_1)), (e_2, \pi_{e_2}(x_2, y_2)) \in \breve{\mathcal{T}}$,*
   $(e_1, \pi_{e_1}(x_1, y_1)) * (e_2, \pi_{e_2}(x_2, y_2)) = (e_2, \pi_{e_2}(x_2, y_2)) * (e_1, \pi_{e_1}(x_1, y_1))$
7. *L'opérateur de* concurrence *"$*$" est distributif sur l'opérateur de* choix *"$+$", i.e., soient les tuples de transitions $(e_1, \pi_{e_1}(x_1, y_1)), (e_2, \pi_{e_2}(x_2, y_2)), (e_3, \pi_{e_3}(x_3, y_3)) \in \breve{\mathcal{T}}$,*
   $$(e_1, \pi_{e_1}(x_1, y_1)) * \Big[(e_2, \pi_{e_2}(x_2, y_2)) + (e_3, \pi_{e_3}(x_3, y_3))\Big] =$$
   $$\Big[(e_1, \pi_{e_1}(x_1, y_1)) * (e_2, \pi_{e_2}(x_2, y_2))\Big] + \Big[(e_1, \pi_{e_1}(x_1, y_1)) * (e_3, \pi_{e_3}(x_3, y_3))\Big]$$

Définition 9.2.10. *Soient p tuples de transitions de $\breve{\mathcal{T}}$, on définit la forme itérée de l'opérateur de concurrence avec la notation suivante :*

$$\overset{p}{\underset{i=1}{*}}(e_i, \pi_{e_i}(x_i, y_i)) = (e_1, \pi_{e_1}(x_1, y_1)) * (e_2, \pi_{e_2}(x_2, y_2)) * \ldots * (e_c, \pi_{e_c}(x_c, y_c)) \tag{9.21}$$

Définition 9.2.11. *Soient p combinaisons simultanées de couples $(csc_i)_{i=1}^{p}$, on appelle ccc la **combinaison concurrente de couples** des $(csc_i)_{i=1}^{p}$:*

$$ccc = \overset{p}{\underset{i=1}{*}} csc_i = csc_1 * csc_2 * \ldots * csc_p \tag{9.22}$$

Définition 9.2.12. *Soit un événement $e \in \breve{\mathcal{E}}$ et une combinaison concurrente de couples ccc, on définit $O_e(ccc)$ le nombre de fois que l'événement e apparaît dans les combinaisons simultanées de couples (csc) de ccc.*

Propriété 9.2.1. *Soit un événement $e \in \breve{\mathcal{E}}$ de degré d'idempotence η_e, une combinaison concurrente de couples $ccc = csc_1 * \ldots * csc_p$ avec p combinaisons simultanées de couples et un $O_e(ccc)$ donné, alors le produit de concurrence "$*$" a les propriétés suivantes :*

1. $$ccc = \underbrace{csc_1 * \ldots * csc_p}_{O_e(ccc)=\eta_e} = (e, \prod_{i\in[1..p]/e\in\breve{\mathcal{E}}_{csc_i}} \pi_e(x^{(i)}, y^{(i)})) \cdot (csc'_1 * \ldots * csc'_p)$$
$$si\ e \in \breve{\mathcal{E}}\ et\ csc'_i = \begin{cases} csc_i & si\ e \notin \breve{\mathcal{E}}_{csc_i} \\ csc_i^{-e} & si\ e \in \breve{\mathcal{E}}_{csc_i} \end{cases} \tag{9.23}$$

2. $$ccc = \underbrace{csc_1 * \ldots * csc_p}_{O_e(ccc)<\eta_e} = csc'_1 * \ldots * csc'_p$$
$$si\ e \in \bar{\mathcal{E}}\ et\ csc'_i = \begin{cases} csc_i & si\ e \notin \breve{\mathcal{E}}_{csc_i} \\ csc_i^{-e} & si\ e \in \breve{\mathcal{E}}_{csc_i} \end{cases} \tag{9.24}$$

3. $$ccc = \underbrace{csc_1 * \ldots * csc_p}_{O_e(ccc)<\eta_e} = O_{\breve{\mathcal{T}}} \qquad si\ e \in \mathcal{E} \tag{9.25}$$

Propriété 9.2.2. *Soit un événement $e \in \breve{\mathcal{E}}$, une combinaison concurrente de couples $ccc = csc_1 * \ldots * csc_N$ avec N combinaisons simultanées de couples, où une combinaison simultanée de couples csc_i représente l'occurrence simultanée de tuples d'événements dans l'automate complété $\breve{\mathcal{A}}^{(i)}$, et l'ensemble d'indices d'automates concernés $\mathcal{O}_e$ par cet événement e, alors le produit de concurrence "$*$" a les propriétés :*

- *Si pour tout csc_i, où $i \in \mathcal{O}_e$, $e \in \breve{\mathcal{E}}_{csc_i}$ alors*

$$ccc = csc_1 * \ldots * csc_N = (e, \prod_{i\in[1..N]/e\in\breve{\mathcal{E}}_{csc_i}} \pi_e(x^{(i)}, y^{(i)})) \cdot (csc'_1 * \ldots * csc'_N)$$
$$où\ csc'_i = \begin{cases} csc_i & si\ e \notin \breve{\mathcal{E}}_{csc_i} \\ csc_i^{-e} & si\ e \in \breve{\mathcal{E}}_{csc_i} \end{cases} \tag{9.26}$$

• *S'il existe* csc_i*, où* $i \in \mathcal{O}_e$*, tel que* $e \notin \breve{\mathcal{E}}_{csc_i}$ *alors*

si $e \in \bar{\mathcal{E}}$, $ccc = csc_1 * \ldots * csc_N = csc'_1 * \ldots * csc'_N$

$$\text{où } csc'_i = \begin{cases} csc_i & \text{si } e \notin \breve{\mathcal{E}}_{csc_i} \\ csc_i^{-e} & \text{si } e \in \breve{\mathcal{E}}_{csc_i} \end{cases} \tag{9.27}$$

si $e \in \mathcal{E}$, $ccc = csc_1 * \ldots * csc_N = O_{\breve{\mathcal{T}}}$ (9.28)

Corollaire 9.2.1. *Soit un événement* $e \in \mathcal{E}$ *et une combinaison concurrente de couples* $ccc = (ccc_A) * (ccc_B)$*, où* ccc_A *et* ccc_B *sont des combinaisons concurrentes de couples tel que* $ccc_A = csc_1 * \ldots * csc_p$ *et* $ccc_B = csc_{p+1} * \ldots * csc_N$*,*

s'il existe $i \in \mathcal{O}_e$*, où* $i \leq p$*, tel que* $e \notin \breve{\mathcal{E}}_{csc_i}$*, alors*

l'évaluation de $ccc_A = O_{\breve{\mathcal{T}}}$ *et par le produit de concurrence avec le tuple de transition nul (propriété 2 de la définition 9.2.9) ce qui donne* $ccc = O_{\breve{\mathcal{T}}}$*.*

Définition 9.2.13. *Soit un modèle SAN* $(\breve{\mathcal{E}}, \breve{\mathcal{A}}^{(i)}, \hat{\mathcal{S}}, \mathcal{F})$ *à temps discret (où* $i \in [1..N]$*) et une combinaison concurrente de couples* ccc *relative à une transition globale du modèle SAN telle que* $ccc = csc_1 * \ldots * csc_N$*, où* csc_i *représente l'occurrence simultanée de tuples d'événements dans l'automate complété* $\breve{\mathcal{A}}^{(i)}$*, on définit l'attribut de position "*a*" pour la combinaison simultanée de couples* csc_i *tel que :*

$$a(csc_i) = i$$

étant i *l'indice de l'automate complété* $\breve{\mathcal{A}}^{(i)}$ *du modèle SAN (où* $i \in [1..N]$*).*

9.3 Produit Tensoriel Complexe

Dans cette section, on va présenter les définitions et notations pour le *produit tensoriel complexe*. Le produit tensoriel complexe est un opérateur matriciel. Les éléments des matrices qui sont les facteurs d'un produit tensoriel complexe sont des tuples de transitions combinés sur un ensemble $\breve{\mathcal{T}}$.

Définition 9.3.1. *On appelle* **combinaison complexe** *sur* $\breve{\mathcal{T}}$ *une forme "multi-linéaire" du type :*

$$cx = \overset{n}{\underset{i=1}{+}} \overset{p_i}{\underset{j=1}{\cdot}} (e_{ij}, \pi_{e_{ij}}(x_{ij}, y_{ij}))$$

Par exemple, $(e_1, \pi_{e_1}(x_1, y_1)) \cdot (e_2, \pi_{e_2}(x_2, y_2)) + (e_3, \pi_{e_3}(x_3, y_3)) \cdot (e_4, \pi_{e_4}(x_4, y_4))$ est une combinaison complexe composée par les tuples de transition $(e_1, \pi_{e_1}(x_1, y_1))$, $(e_2, \pi_{e_2}(x_2, y_2))$, $(e_3, \pi_{e_3}(x_3, y_3))$ et $(e_4, \pi_{e_4}(x_4, y_4))$.

Définition 9.3.2. *On appelle matrice complexe sur* $\breve{\mathcal{T}}$ *une matrice dont les éléments sont des combinaisons complexes sur* $\breve{\mathcal{T}}$*.*

✍ **Soit**

A	une matrice complexe carrée[2] ;
n_A	la dimension de la matrice complexe A ;
a_{ij}	la combinaison complexe de la ligne i et la colonne j de la matrice complexe A ;
$a_{[ik],[jl]}$	la combinaison complexe de la ligne k du i-ème bloc horizontal et la colonne l du j-ème bloc vertical de la matrice complexe A.

Le produit de concurrence de deux matrices complexes carrés A et B de dimension n est une matrice carré de dimension n.

Définition 9.3.3. *On appelle produit de concurrence de deux matrices complexes A et B de dimension n la matrice complexe $C = A * B$ de dimension n dont les éléments sont définis par :*

$$\begin{array}{l} i \in [1..n], l \in [1..n], \\ \quad c_{il} = \underset{r=1}{\overset{n}{+}} (a_{ir} * b_{rl}) \end{array} \tag{9.29}$$

Par exemple, soient deux matrices complexes A et B :

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

Le *produit de concurrence* défini par $C = A * B$ est égal à :

$$C = \left(\begin{array}{c|c} \begin{array}{c} a_{11} * b_{11} \\ + \\ a_{12} * b_{21} \end{array} & \begin{array}{c} a_{11} * b_{12} \\ + \\ a_{12} * b_{22} \end{array} \\ \hline \begin{array}{c} a_{21} * b_{11} \\ + \\ a_{22} * b_{21} \end{array} & \begin{array}{c} a_{21} * b_{12} \\ + \\ a_{22} * b_{22} \end{array} \end{array}\right)$$

On pourrait aisément étendre cette définition à des matrices non-carrées avec les contraintes habituelles de cohérence des dimensions (ligne, colonne).

Ensuite, on va introduire l'opérateur matriciel nommé *produit tensoriel complexe*.

Le produit tensoriel complexe de deux matrices complexes A et B de dimensions n_A et n_B respectivement est une matrice de dimension $n_A n_B$. Cette matrice peut être vue comme une matrice avec n_A

[2] On se restreint ici pour nos besoin à des matrices carrées, mais l'opérateur du produit tensoriel complexe peut être utilisé par des matrices non-carrées.

blocs, chacun de dimension n_B. La définition de chacun des éléments de la matrice résultante est faite en précisant à quel bloc l'élément appartient et sa position interne dans le bloc.

Définition 9.3.4. *On appelle produit tensoriel complexe de deux matrices complexes A et B la matrice complexe $M = A \circledast B$ de dimension $n_M = n_A n_B$ dont les éléments sont définis par :*

$$m_{[ik],[jl]} = a_{ij} * b_{kl} \quad \textit{avec } i \in [1..n_A],\ j \in [1..n_A],\ k \in [1..n_B] \textit{ et } l \in [1..n_B] \tag{9.30}$$

Il faut garder à l'esprit que les éléments de la matrice complexe $M = A \circledast B$ sont des *produits de concurrence* (Définition 9.2.9, page 167) des combinaisons complexes des matrices A et B.

Par exemple, soient deux matrices complexes A et B :

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix}$$

Le *produit tensoriel complexe* défini par $C = A \circledast B$ est égal à :

$$C = \begin{pmatrix} a_{11} * B & a_{12} * B \\ a_{21} * B & a_{22} * B \end{pmatrix}$$

$$C = \left(\begin{array}{ccc|ccc} a_{11} * b_{11} & a_{11} * b_{12} & a_{11} * b_{13} & a_{12} * b_{11} & a_{12} * b_{12} & a_{12} * b_{13} \\ a_{11} * b_{21} & a_{11} * b_{22} & a_{11} * b_{23} & a_{12} * b_{21} & a_{12} * b_{22} & a_{12} * b_{23} \\ a_{11} * b_{31} & a_{11} * b_{32} & a_{11} * b_{33} & a_{12} * b_{31} & a_{12} * b_{32} & a_{12} * b_{33} \\ \hline a_{21} * b_{11} & a_{21} * b_{12} & a_{21} * b_{13} & a_{22} * b_{11} & a_{22} * b_{12} & a_{22} * b_{13} \\ a_{21} * b_{21} & a_{21} * b_{22} & a_{21} * b_{23} & a_{22} * b_{21} & a_{22} * b_{22} & a_{22} * b_{23} \\ a_{21} * b_{31} & a_{21} * b_{32} & a_{21} * b_{33} & a_{22} * b_{31} & a_{22} * b_{32} & a_{22} * b_{33} \end{array}\right)$$

Dans cet exemple, l'élément c_{53} (*i.e.*, l'élément $a_{21} * b_{23}$) est dans le bloc $(2, 1)$ et sa position interne dans ce bloc est $(2, 3)$.

Définition 9.3.5. *On appelle I_n la matrice identité complexe de dimension n telle que l'élément δ_{ij} (i.e., l'élément dans la ligne i et la colonne j de cette matrice) est égal à l'élément neutre $I_{\breve{\mathcal{I}}}$ (si $i = j$) ou égal à l'élément nul $O_{\breve{\mathcal{I}}}$ (si $i \neq j$), où $i \in [1..n]$ et $j \in [1..n]$.*

Un cas spécifique de produit tensoriel complexe est le produit tensoriel complexe d'une matrice complexe carrée par une matrice identité complexe. On appelle ce produit de *facteur normal*. Par exemple, soit la matrice complexe A et une matrice identité complexe I_3 :

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad I_3 = \begin{pmatrix} I_{\breve{\mathcal{I}}} & O_{\breve{\mathcal{I}}} & O_{\breve{\mathcal{I}}} \\ O_{\breve{\mathcal{I}}} & I_{\breve{\mathcal{I}}} & O_{\breve{\mathcal{I}}} \\ O_{\breve{\mathcal{I}}} & O_{\breve{\mathcal{I}}} & I_{\breve{\mathcal{I}}} \end{pmatrix}$$

Le facteur normal $A \circledast I_3$ est égal à :

$$\left(\begin{array}{ccc|ccc} a_{11}*I_{\check{\mathcal{I}}} & a_{11}*O_{\check{\mathcal{I}}} & a_{11}*O_{\check{\mathcal{I}}} & a_{12}*I_{\check{\mathcal{I}}} & a_{12}*O_{\check{\mathcal{I}}} & a_{12}*O_{\check{\mathcal{I}}} \\ a_{11}*O_{\check{\mathcal{I}}} & a_{11}*I_{\check{\mathcal{I}}} & a_{11}*O_{\check{\mathcal{I}}} & a_{12}*O_{\check{\mathcal{I}}} & a_{12}*I_{\check{\mathcal{I}}} & a_{12}*O_{\check{\mathcal{I}}} \\ a_{11}*O_{\check{\mathcal{I}}} & a_{11}*O_{\check{\mathcal{I}}} & a_{11}*I_{\check{\mathcal{I}}} & a_{12}*O_{\check{\mathcal{I}}} & a_{12}*O_{\check{\mathcal{I}}} & a_{12}*I_{\check{\mathcal{I}}} \\ \hline a_{21}*I_{\check{\mathcal{I}}} & a_{21}*O_{\check{\mathcal{I}}} & a_{21}*O_{\check{\mathcal{I}}} & a_{22}*I_{\check{\mathcal{I}}} & a_{22}*O_{\check{\mathcal{I}}} & a_{22}*O_{\check{\mathcal{I}}} \\ a_{21}*O_{\check{\mathcal{I}}} & a_{21}*I_{\check{\mathcal{I}}} & a_{21}*O_{\check{\mathcal{I}}} & a_{22}*O_{\check{\mathcal{I}}} & a_{22}*I_{\check{\mathcal{I}}} & a_{22}*O_{\check{\mathcal{I}}} \\ a_{21}*O_{\check{\mathcal{I}}} & a_{21}*O_{\check{\mathcal{I}}} & a_{21}*I_{\check{\mathcal{I}}} & a_{22}*O_{\check{\mathcal{I}}} & a_{22}*O_{\check{\mathcal{I}}} & a_{22}*I_{\check{\mathcal{I}}} \end{array}\right) = \left(\begin{array}{ccc|ccc} a_{11} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & a_{12} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} \\ O_{\check{\mathcal{I}}} & a_{11} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & a_{12} & O_{\check{\mathcal{I}}} \\ O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & a_{11} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & a_{12} \\ \hline a_{21} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & a_{22} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} \\ O_{\check{\mathcal{I}}} & a_{21} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & a_{22} & O_{\check{\mathcal{I}}} \\ O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & a_{21} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & a_{22} \end{array}\right)$$

Le facteur normal $I_3 \circledast A$ est égal à :

$$\left(\begin{array}{cc|cc|cc} I_{\check{\mathcal{I}}}*a_{11} & I_{\check{\mathcal{I}}}*a_{12} & O_{\check{\mathcal{I}}}*a_{11} & O_{\check{\mathcal{I}}}*a_{12} & O_{\check{\mathcal{I}}}*a_{11} & O_{\check{\mathcal{I}}}*a_{12} \\ I_{\check{\mathcal{I}}}*a_{21} & I_{\check{\mathcal{I}}}*a_{22} & O_{\check{\mathcal{I}}}*a_{21} & O_{\check{\mathcal{I}}}*a_{22} & O_{\check{\mathcal{I}}}*a_{21} & O_{\check{\mathcal{I}}}*a_{22} \\ \hline O_{\check{\mathcal{I}}}*a_{11} & O_{\check{\mathcal{I}}}*a_{12} & I_{\check{\mathcal{I}}}*a_{11} & I_{\check{\mathcal{I}}}*a_{12} & O_{\check{\mathcal{I}}}*a_{11} & O_{\check{\mathcal{I}}}*a_{12} \\ O_{\check{\mathcal{I}}}*a_{21} & O_{\check{\mathcal{I}}}*a_{22} & I_{\check{\mathcal{I}}}*a_{21} & I_{\check{\mathcal{I}}}*a_{22} & O_{\check{\mathcal{I}}}*a_{21} & O_{\check{\mathcal{I}}}*a_{22} \\ \hline O_{\check{\mathcal{I}}}*a_{11} & O_{\check{\mathcal{I}}}*a_{12} & O_{\check{\mathcal{I}}}*a_{11} & O_{\check{\mathcal{I}}}*a_{12} & I_{\check{\mathcal{I}}}*a_{11} & I_{\check{\mathcal{I}}}*a_{12} \\ O_{\check{\mathcal{I}}}*a_{21} & O_{\check{\mathcal{I}}}*a_{22} & O_{\check{\mathcal{I}}}*a_{21} & O_{\check{\mathcal{I}}}*a_{22} & I_{\check{\mathcal{I}}}*a_{21} & I_{\check{\mathcal{I}}}*a_{22} \end{array}\right) = \left(\begin{array}{cc|cc|cc} a_{11} & a_{12} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} \\ a_{21} & a_{22} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} \\ \hline O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & a_{11} & a_{12} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} \\ O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & a_{21} & a_{22} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} \\ \hline O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & a_{11} & a_{12} \\ O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & O_{\check{\mathcal{I}}} & a_{21} & a_{22} \end{array}\right)$$

✍ **Notation**

$a_{[i_1,\ldots,i_m,],[j_1,\ldots,j_m]}$ l'élément de la matrice complexe $A = \overset{N}{\underset{l=1}{\circledast}}$ de taille $\prod_{l=1}^{m} n_l$ localisé dans le bloc de taille $\prod_{l=2}^{m} n_l$ de coordonnées i_1, j_1, puis à l'intérieur de ce bloc, dans le bloc de taille $\prod_{l=3}^{m} n_l$ de coordonnées i_2, j_2 et ainsi de suite jusqu'à la position i_m, j_m du bloc (plus interne) de taille n_m.

Propriété 9.3.1. *Le produit tensoriel complexe est un opérateur **associatif**.*

Démonstration. Soient trois matrices complexes carrées A, B et C, on a :

$$(A \circledast B) \circledast C = A \circledast (B \circledast C)$$

Soit :
- D le produit tensoriel complexe $A \circledast B$
- E le produit tensoriel complexe $B \circledast C$
- H le produit tensoriel complexe $(A \circledast B) \circledast C$
- H' le produit tensoriel complexe $A \circledast (B \circledast C)$

Les éléments de la matrice complexe $D = A \circledast B$ ont la forme :
$i, j \in [1..n_A]$, $k, l \in [1..n_B]$,
$d_{[ik],[jl]} = a_{ij} * b_{kl}$

Les éléments de la matrice complexe $H = D \circledast C$ ont la forme :
$i, j \in [1..n_A]$, $k, l \in [1..n_B]$, $m, n \in [1..n_C]$,
$h_{[ikm],[jln]} = d_{[ik],[jl]} * c_{mn}$

En remplaçant les éléments de D par leur valeur dans l'égalité des éléments de H, puisque $*$ est un opérateur associatif :
$i, j \in [1..n_A]$, $k, l \in [1..n_B]$, $m, n \in [1..n_C]$,
$\boxed{h_{[ikm],[jln]} = (a_{ij} * b_{kl}) * c_{mn} = a_{ij} * b_{kl} * c_{mn}}$

Les éléments de la matrice complexe $E = B \circledast C$ ont la forme :
$i, j \in [1..n_B]$, $k, l \in [1..n_C]$,
$e_{[ik],[jl]} = b_{ij} * c_{kl}$

Les éléments de la matrice complexe $H' = A \circledast E$ ont la forme :
$i, j \in [1..n_A]$, $k, l \in [1..n_B]$, $m, n \in [1..n_C]$,
$h'_{[ikm],[jln]} = a_{ij} * e_{[km],[ln]}$

En remplaçant les éléments de E par leur valeur dans l'égalité des éléments de H', puisque $*$ est un opérateur associatif :
$i, j \in [1..n_A]$, $k, l \in [1..n_B]$, $m, n \in [1..n_C]$,
$\boxed{h'_{[ikm],[jln]} = a_{ij} * (b_{kl} * c_{mn}) = a_{ij} * b_{kl} * c_{mn}}$

Les éléments des matrices complexes H et H' sont égaux et par conséquence

$$(A \circledast B) \circledast C = A \circledast (B \circledast C)$$

❑

✍ **Notation**

σ	une permutation sur l'intervalle $[1..N]$ (une permutation σ servira à établir un nouvel ordre pour une suite de N matrices indexée de 1 à N) ;
$\sigma(i)$	le nouveau rang de la i-ème matrice dans l'ordre identifié par la permutation σ ;
σ_k	l'indice de la matrice placée au rang k de l'ordre identifié par la permutation σ (si $\sigma_k = i, \sigma(i) = k$) ;
$[i_1, \ldots, i_N]_\sigma$	une modification de l'ordre des coordonnés d'un vecteur selon une permutation σ ;
P_σ	la matrice de permutation de taille $\prod_{i=1}^{N} n_i$ définie par : $p_{[i_1,\ldots,i_N][j_1,\ldots,j_N]_\sigma} = \begin{cases} 1 & \text{si } j_m = i_{\sigma_m} \text{ avec } m \in [1..N] \\ 0 & \text{sinon} \end{cases}$
P_σ^T	la transposée et aussi l'inverse de la matrice P_σ.

Propriété 9.3.2. *Soient deux matrices complexes carrées A et B et la transposition σ sur $[1..2]$, où $\sigma(1) = 2$ et $\sigma(2) = 1$, alors avec la définition de P_σ ci-dessus :*

$$A \circledast B = P_\sigma \times (B \circledast A) \times P_\sigma^T$$

Démonstration. Nous allons prouver cette propriété pour le cas plus général qui s'applique à une suite de N matrices complexes dans les mêmes termes :

$$\overset{N}{\underset{k=1}{\circledast}} A^{(k)} = P_\sigma \times \overset{N}{\underset{k=1}{\circledast}} A^{(\sigma_k)} \times P_\sigma^T \tag{9.31}$$

Soit :

- H le produit tensoriel complexe, membre gauche de l'équation (9.31), $\overset{N}{\underset{k=1}{\circledast}} A^{(k)}$
- C le produit tensoriel complexe, $\overset{N}{\underset{k=1}{\circledast}} A^{(\sigma_k)}$
- D le membre droit de l'équation (9.31), $P_\sigma \times C \times P_\sigma^T = P_\sigma \times \overset{N}{\underset{k=1}{\circledast}} A^{(\sigma_k)} \times P_\sigma^T$

Les éléments de la matrice complexe H ont la forme :

$i_k, j_k \in [1..n_k]$,

$$\boxed{h_{[i_1,...,i_N][j_1,...,j_N]} = \overset{N}{\underset{k=1}{*}} a_{i_k j_k}^{(k)}}$$

Le produit $C \times P_\sigma^T$ permutant les colonnes de C et le produit $P_\sigma \times C$ permutant les lignes de C, on calcule les éléments de $P_\sigma \times C \times P_\sigma^T$ selon :

$i_{\sigma_k}, j_{\sigma_k} \in [1..n_{\sigma_k}]$,

$$d_{[i_1,...,i_N][j_1,...,j_N]} = c_{[i_{\sigma_1},...,i_{\sigma_N}]\sigma[j_{\sigma_1},...,j_{\sigma_N}]\sigma}$$

$$c_{[i_{\sigma_1},...,i_{\sigma_N}]\sigma[j_{\sigma_1},...,j_{\sigma_N}]\sigma} = \overset{N}{\underset{k=1}{*}} a_{i_{\sigma_k} j_{\sigma_k}}^{(\sigma_k)}$$

Étant donné que σ est une bijection sur $[1..N]$, il est possible de remplacer σ_k par l et k par $\sigma(l)$:

$i_l, j_l \in [1..n_l]$,

$$d_{[i_1,...,i_N][j_1,...,j_N]} = \overset{\sigma(N)}{\underset{l=\sigma(1)}{*}} a_{i_l j_l}^{(l)}$$

En sachant que l'opérateur "$*$" est commutatif :

$i_k, j_k \in [1..n_k]$,

$$\overset{N}{\underset{k=1}{*}} a_{i_k j_k}^{(k)} = \overset{\sigma(N)}{\underset{l=\sigma(1)}{*}} a_{i_l j_l}^{(l)}$$

Par la définition de C et H :

$$h_{[i_1,...,i_N][j_1,...,j_N]} = c_{[i_{\sigma_1},...,i_{\sigma_N}]\sigma[j_{\sigma_1},...,j_{\sigma_N}]\sigma}$$

Par la définition de D :

$$\boxed{h_{[i_1,...,i_N][j_1,...,j_N]} = d_{[i_1,...,i_N][j_1,...,j_N]}}$$

❑

Propriété 9.3.3. *Soient quatre matrices complexes carrées A, B, C et D (de dimensions n_A, n_B, n_C et n_D respectivement), on a la propriété de distributivité suivante, sous réserve que $n_A = n_B$ et $n_C = n_D$:*

$$(A \circledast C) * (B \circledast D) = (A * B) \circledast (C * D)$$

Démonstration. Soit :

- E le produit tensoriel complexe $A \circledast C$
- F le produit tensoriel complexe $B \circledast D$
- H le produit de concurrence $E * F$
- E' le produit de concurrence $A * B$
- F' le produit de concurrence $C * D$
- H' le produit tensoriel complexe $E' \circledast F'$

Les éléments de la matrice complexe E ont la forme :

$$\begin{array}{l} i, j \in [1..n_A], \quad k, l \in [1..n_C], \\ \quad e_{[ik],[jl]} = a_{ij} * c_{kl} \end{array}$$

Les éléments de la matrice complexe F ont la forme :

$$\begin{array}{l} x, y \in [1..n_B], \quad z, w \in [1..n_D], \\ \quad f_{[xz],[yw]} = b_{xy} * d_{zw} \end{array}$$

Les éléments de la matrice complexe H ont la forme :

$$\begin{array}{l} i \in [1..n_A], \quad y \in [1..n_B], \quad k \in [1..n_C], \quad w \in [1..n_D], \\ \quad h_{[ik],[yw]} = \underset{r=1}{\overset{n_A}{+}} \left(\underset{s=1}{\overset{n_C}{+}} (e_{[ik],[rs]} * f_{[rs],[yw]}) \right) \end{array}$$

En remplaçant les éléments de E et F par leurs valeurs dans les égalités des éléments de H :

$$\boxed{h_{[ik],[yw]} = \overset{n_A}{\underset{r=1}{+}} \left(\overset{n_C}{\underset{s=1}{+}} (a_{ir} * c_{ks} * b_{ry} * d_{sw}) \right)}$$

Les éléments de la matrice complexe E' ont la forme :

$$\begin{array}{l} i \in [1..n_A], \quad y \in [1..n_B], \\ \quad e'_{iy} = \overset{n_A}{\underset{r=1}{+}} (a_{ir} * b_{ry}) \end{array}$$

Les éléments de la matrice complexe F' ont la forme :

$$\begin{array}{l} k \in [1..n_C], \quad w \in [1..n_D], \\ \quad f'_{kw} = \overset{n_C}{\underset{s=1}{+}} (c_{ks} * d_{sw}) \end{array}$$

Les éléments de la matrice complexe H' ont la forme :

$$\begin{array}{l} i \in [1..n_A], \quad y \in [1..n_B], \quad k \in [1..n_C], \quad w \in [1..n_D], \\ \quad h'_{[ik],[yw]} = e'_{iy} * f'_{kw} \end{array}$$

En remplaçant les éléments de E' et F' par leurs valeurs dans les égalités des éléments de H' :

$$h'_{[ik],[yw]} = \left(\overset{n_A}{\underset{r=1}{+}} (a_{ir} * b_{ry}) \right) * \left(\overset{n_C}{\underset{s=1}{+}} (c_{ks} * d_{sw}) \right)$$

Par la propriété de la distributivité du produit de concurrence sur la somme de choix, les éléments de la matrice complexe H' ont la forme :

$$\boxed{h'_{[ik],[yw]} = \overset{n_A}{\underset{r=1}{+}} \left(\overset{n_C}{\underset{s=1}{+}} (a_{ir} * b_{ry} * c_{ks} * d_{sw}) \right)}$$

En sachant que le produit de concurrence "$*$" est un opérateur commutatif, les éléments des matrices complexes H et H' sont égaux et par conséquence :

$$(A \circledast C) * (B \circledast D) = (A * B) \circledast (C * D)$$

❑

Propriété 9.3.4. *Soient deux matrices complexes carrées A et B, le produit tensoriel complexe $A \circledast B$ peut être décomposé en facteurs normaux, i.e. :*

$$(A \circledast B) = (A \circledast I_{n_B}) * (I_{n_A} \circledast B) \tag{9.32}$$

Démonstration. La décomposition en facteurs normaux est facilement vérifiée comme un cas particulier de la propriété 9.3.3, en sachant que :

$$A * I_{n_A} = A \qquad \text{et} \qquad I_{n_A} * A = A$$

Il est possible de remplacer $A \circledast B$ par $(A * I_{n_A}) \circledast (I_{n_B} * B)$ dans l'équation (9.32), *i.e.* :

$$(A * I_{n_A}) \circledast (I_{n_B} * B) = (A \circledast I_{n_B}) * (I_{n_A} \circledast B) \tag{9.33}$$

L'équation (9.33) devient un corollaire de la propriété 9.3.3.

❑

✍ **Notation**

$nleft_i$	le produit des dimensions de toutes les matrices avant $A^{(i)}$, *i.e.*, $\prod_{k=1}^{i-1} n_k$ (cas particulier : $nleft_1 = 1$) ;
$nright_i$	le produit des dimensions de toutes les matrices après $A^{(i)}$, *i.e.*, $\prod_{k=i+1}^{N} n_k$ (cas particulier : $nright_N = 1$) ;

Propriété 9.3.5. *La propriété 9.3.4 peut être généralisée pour une suite de N matrices, i.e. :*

$$\overset{N}{\underset{i=1}{\circledast}} A^{(i)} = \overset{N}{\underset{i=1}{*}} \left(I_{nleft_i} \circledast A^{(i)} \circledast I_{nright_i} \right) \tag{9.34}$$

Démonstration. L'équation (9.34) peut être réécrite dans les termes suivants :

$$\begin{aligned} & A^{(1)} \circledast \cdots \circledast A^{(N-2)} \circledast A^{(N-1)} \circledast A^{(N)} = \\ & \quad A^{(1)} \circledast I_{nright_1} \\ & * I_{n_1} \circledast A^{(2)} \circledast I_{nright_2} \\ & * \cdots \\ & * I_{nleft_{N-1}} \circledast A^{(N-1)} \circledast I_{n_N} \\ & * I_{nleft_N} \circledast A^{(N)} \end{aligned} \tag{9.35}$$

On va démontrer cette propriété par récurrence.

Soit :
– $H^{(N)}$ la matrice complexe $A^{(N)}$;

– $H^{(N-1)}$ le produit tensoriel complexe $A^{(N-1)} \circledast H^{(N)}$ (une matrice complexe de dimension $n_{N-1}n_N = nright_{N-2}$) ;

– $H^{(k)}$ le produit tensoriel complexe $A^{(k)} \circledast H^{(k+1)}$ (une matrice complexe de dimension $n_k n_{k+1} \dots n_N = nright_{k-1}$) ;

En utilisant la propriété 9.3.4 sur $H^{(N-1)}$:

$$\begin{aligned} H^{(N-1)} =& A^{(N-1)} \circledast H^{(N)} \\ =& (A^{(N-1)} \circledast I_{n_N}) * (I_{n_{N-1}} \circledast H^{(N)}) \end{aligned}$$

En remplaçant $H^{(N)}$:

$$H^{(N-1)} = (A^{(N-1)} \circledast I_{n_N}) * (I_{n_{N-1}} \circledast A^{(N)})$$

En utilisant la propriété 9.3.4 sur $H^{(N-2)}$:

$$\begin{aligned} H^{(N-2)} =& A^{(N-2)} \circledast H^{(N-1)} \\ =& (A^{(N-2)} \circledast I_{nright_{N-2}}) * (I_{n_{N-2}} \circledast H^{(N-1)}) \end{aligned}$$

En remplaçant $H^{(N-1)}$:

$$\begin{aligned} H^{(N-2)} =& (A^{(N-2)} \circledast I_{nright_{N-2}}) \\ & * \Big(I_{n_{N-2}} \circledast [(A^{(N-1)} \circledast I_{n_N}) * (I_{n_{N-1}} \circledast A^{(N)})]\Big) \end{aligned}$$

En utilisant les propriétés 9.3.1 et 9.3.3 :

$$\begin{aligned} H^{(N-2)} =& (A^{(N-2)} \circledast I_{nright_{N-2}}) \\ & * (I_{n_{N-2}} \circledast A^{(N-1)} \circledast I_{n_N}) * (I_{n_{N-2}} \circledast I_{n_{N-1}} \circledast A^{(N)}) \end{aligned}$$

En sachant que $I_{n_{N-2}} \circledast I_{n_{N-1}} = I_{n_{N-2}n_{N-1}}$:

$$\begin{aligned} H^{(N-2)} =& (A^{(N-2)} \circledast I_{nright_{N-2}}) \\ & * (I_{n_{N-2}} \circledast A^{(N-1)} \circledast I_{n_N}) * (I_{n_{N-2}n_{N-1}} \circledast A^{(N)}) \end{aligned}$$

En appliquant ces mêmes pas jusqu'à $H^{(1)}$ on obtient :

$$\begin{aligned} H^{(1)} =& A^{(1)} \circledast I_{n_2 \dots n_N} \\ & * I_{n_1} \circledast A^{(2)} \circledast I_{n_3 \dots n_N} \\ & * \cdots \\ & * I_{n_1 \dots n_{N-2}} \circledast A^{(N-1)} \circledast I_{n_N} \\ & * I_{n_1 \dots n_{N-1}} \circledast A^{(N)} \end{aligned}$$

Par définition de $nleft$ et $nright$ on a :

$$\begin{aligned} H^{(1)} =& A^{(1)} \circledast I_{nright_1} \\ & * I_{n_1} \circledast A^{(2)} \circledast I_{nright_2} \\ & * \cdots \\ & * I_{nleft_{N-1}} \circledast A^{(N-1)} \circledast I_{n_N} \\ & * I_{nleft_N} \circledast A^{(N)} \end{aligned}$$

❑

9.4 Conclusion

Dans ce chapitre, on a présenté l'algèbre tensorielle complexe (ATX) et ses propriétés. Ces propriétés seront utilisées dans le chapitre 10 pour la définition d'une formule tensorielle pour le formalisme des Réseaux d'Automates Stochastiques (SAN) à temps discret.

Chapitre 10

Descripteur discret

La modélisation de systèmes complexes sur une échelle de temps discret est telle que plusieurs événements peuvent avoir lieu dans une même unité de temps. Une des grandes difficultés de la modélisation de ces systèmes n'est pas seulement de déterminer toutes les combinaisons possibles de tels événements (dans la même unité de temps), mais aussi de résoudre les cas de conflits lorsque le tirage d'un événement amène à un état où un autre n'est plus réalisable (dans la même unité de temps).

Dans ce chapitre, on va présenter[1] une façon compacte de représenter les transitions d'un modèle SAN à temps discret, utilisant une représentation tensorielle complexe dénommée *descripteur discret*. La représentation tensorielle complexe du descripteur discret est basée sur l'algèbre tensorielle complexe définie dans le chapitre 9.

Dans la section 10.1, on va reprendre la définition de l'automate global d'un modèle SAN à temps discret afin de réécrire sa matrice de transition globale $\mathcal{G}$ de façon à faciliter la présentation du descripteur discret du modèle. On présente dans la section 10.2 les matrices utilisées pour le descripteur discret. Ensuite, dans la section 10.3, on présente le descripteur discret d'un modèle SAN à temps discret et on donne un exemple d'un petit modèle SAN afin d'illustrer l'obtention du descripteur discret de ce modèle.

10.1 Automate global

Un modèle SAN est une façon modulaire de représenter l'automate global qui décrit le comportement de l'ensemble du réseau d'automates. On considère un modèle SAN tel que défini dans la définition 7.2.16 (page 130).

✍ **Rappelons**

$\mathcal{L}(\tilde{x}, \tilde{y})$: l'ensemble de chaînes de transitions globales $g \in \mathcal{L}_{pot(\tilde{x})}$ tel que $\tilde{x}_{1_g} = \tilde{x}$ et $\tilde{y}_{C_g} = \tilde{y}$, où C_g est le nombre de chaînons de transition globale de la chaîne de transitions globales g.

Comme on l'a présenté dans le chapitre 7 (Définition 7.3.4, page 135), soit un SAN complété $(\breve{\mathcal{E}}, \breve{\mathcal{A}}^{(i)}, \hat{\mathcal{S}}, \mathcal{F})$, où $i \in [1..N]$. Alors, l'automate global $\mathcal{A}$ est défini par :

[1] Les définitions et notations nécessaires à définir le descripteur discret sont communes dans cette thèse et dans [15].

1. l'ensemble d'états $\hat{\mathcal{S}} = \prod_{i=1}^{N} \mathcal{S}^{(i)}$;
2. l'ensemble d'événements $\breve{\mathcal{E}}$;
3. la fonction d'atteignabilité $\mathcal{F}$;
4. la matrice de transition globale $\mathcal{G} : \hat{\mathcal{S}} \times \hat{\mathcal{S}} \rightarrow \mathcal{L}$, telle que :

$$\forall \tilde{x} \in \hat{\mathcal{S}}, \forall \tilde{y} \in \hat{\mathcal{S}}$$
$$\mathcal{G}(\tilde{x}, \tilde{y}) = \bigcup_{\forall g \in \mathcal{L}(\tilde{x}, \tilde{y})} \{g\}$$

On rappelle que l'objet mathématique qui décrit l'automate global n'est pas le même que celui d'un automate local d'un modèle SAN. Dans un automate local, les éléments de transition de la matrice de transition contiennent un ensemble des *tuples de transitions locales* $(e, \pi(x, y))$, où l'occurrence de n'importe quel événement de l'élément de transition déclenche la transition. Dans l'automate global, les éléments de transition de la matrice de transition contiennent un ensemble de *chaînes de transitions globales* $\{(\tilde{x}_1, \tilde{y}_1, e_1, \Pi_{e_1}(\tilde{x}_1, \tilde{y}_1)), \ldots, (\tilde{x}_C, \tilde{y}_C, e_C, \Pi_{e_C}(\tilde{x}_C, \tilde{y}_C))\}$, où tous les événements associés aux chaînons de cette chaîne *doivent avoir lieu pour que la transition se réalise*.

✍ **Notation**

$(e, \Pi_e(\tilde{x}, \tilde{y}))$	le tuple de transition globale de l'état global $\tilde{x}$ vers l'état global $\tilde{y}$ correspondant au chaînon de transition globale $(\tilde{x}, \tilde{y}, e, \Pi_e(\tilde{x}, \tilde{y}))$.

Définition 10.1.1. *Soit une chaîne de transitions globales* $g = \{(\tilde{x}_1, \tilde{y}_1, e_1, \Pi_{e_1}(\tilde{x}_1, \tilde{y}_1)), \ldots, (\tilde{x}_{C_g}, \tilde{y}_{C_g}, e_{C_g}, \Pi_{e_{C_g}}(\tilde{x}_{C_g}, \tilde{y}_{C_g}))\}$ *composée de* C_g *chaînons de transitions globales, on peut représenter cette chaîne par un produit de simultanéité de tuples de transition globale, où un tuple de transition globale est typiquement un couple*[2] *composé par l'identificateur de l'événement du chaînon et de la probabilité de routage globale de ce chaînon. Par extension, on pourra écrire* g *de deux façon différentes, sous forme de chaîne de transitions globales ou sous forme de produit de simultanéité :*

$$g = \{(\tilde{x}_1, \tilde{y}_1, e_1, \Pi_{e_1}(\tilde{x}_1, \tilde{y}_1)), \ldots, (\tilde{x}_{C_g}, \tilde{y}_{C_g}, e_{C_g}, \Pi_{e_{C_g}}(\tilde{x}_{C_g}, \tilde{y}_{C_g}))\} \tag{10.1}$$

où bien

$$g = \overset{C_g}{\underset{i=1}{\cdot}} \left(e_i, \Pi_{e_i}(\tilde{x}_i, \tilde{y}_i)\right) \tag{10.2}$$

On va réécrire la matrice de transition $\mathcal{G}$ de l'automate global en utilisant l'algèbre tensorielle complexe comme le permet la définition précédente. Dans cette réécriture, l'*union* des chaînes de transitions globales sera une *somme de choix* de l'algèbre tensorielle complexe et une *chaîne de transitions globales* sera une *combinaison simultanée* de transitions globales, *i.e.* :

$$\forall \tilde{x} \in \hat{\mathcal{S}}, \forall \tilde{y} \in \hat{\mathcal{S}}$$
$$\mathcal{G}(\tilde{x}, \tilde{y}) = \underset{\forall g \in \mathcal{L}(\tilde{x}, \tilde{y})}{+} \overset{C_g}{\underset{i=1}{\cdot}} \Big(e_{i_g}, \Pi_{e_{i_g}}(\tilde{x}_{i_g}, \tilde{y}_{i_g})\Big) \tag{10.3}$$

Cette nouvelle vision en utilisant l'algèbre tensorielle complexe sur les éléments de la matrice de transition d'un modèle SAN prépare la présentation du descripteur discret de ce modèle.

[2] On remarque que la représentation d'un chaînon de transition globale par un tuple de transition globale maintient tout les informations nécessaires pour l'enchaînement puisque on conserve les connaissances des paramètres $\tilde{x}_i$ et $\tilde{y}_i$ de $\Pi_{e_i}(\tilde{x}_i, \tilde{y}_i)$.

Dans la section suivante, on va donner les définitions et notations des matrices d'événements utilisées pour le descripteur discret.

10.2 Matrices d'événements

Comme on l'a dit précédemment, plusieurs événements peuvent avoir lieu dans une même unité de temps dans un même automate. Les matrices d'événements représentent les possibilités d'occurrence de plusieurs événements à partir d'un état local du modèle. L'occurrence de plusieurs événements dans un même automate et dans une même unité de temps est représenté par une *chaîne de transitions locales*. On reprend maintenant quelques notations utilisées dans le chapitre 7 afin de présenter la définition des chaînes de transitions locales.

✍ **Notation**

$\mathcal{T}^{(i)}$	l'ensemble des tuples de transition de l'automate complété $\breve{\mathcal{A}}^{(i)}$, où $i \in [1..N]$ et $\mathcal{T}^{(i)} \subseteq \mathcal{T}$;
$\mathcal{T}_e^{(i)}$	l'ensemble des tuples de transition de l'automate complété $\breve{\mathcal{A}}^{(i)}$ déclenchés par l'événement e, où $i \in [1..N]$ et $\mathcal{T}_e^{(i)} \subseteq \mathcal{T}^{(i)}$;
$(e, \pi_e(x^{(i)}, y^{(i)}))$	le tuple de transition de $\mathcal{T}_e^{(i)}$ qui représente la transition de l'état local $x^{(i)} \in \mathcal{S}^{(i)}$ vers l'état local $y^{(i)} \in \mathcal{S}^{(i)}$ de l'automate complété $\breve{\mathcal{A}}^{(i)}$ déclenchée par l'événement e avec la probabilité de routage $\pi_e(x^{(i)}, y^{(i)})$;
ϵ	un sous-ensemble d'événements de $\breve{\mathcal{E}}$;
$\mathcal{T}_\epsilon^{(i)}$	l'union des $\mathcal{T}_e^{(i)}$, où $e \in \epsilon$, *i.e.*, $\mathcal{T}_\epsilon^{(i)} = \bigcup_{e \in \epsilon} \mathcal{T}_e^{(i)}$.

Définition 10.2.1. *Soit un ensemble d'évènements ϵ et un automate complété $\breve{\mathcal{A}}^{(i)}$, on appelle* **chaîne de transitions locales** *sur ϵ, une liste ordonnée $\{(e_1, \pi_{e_1}(x_1^{(i)}, y_1^{(i)})), \ldots, (e_C, \pi_{e_C}(x_C^{(i)}, y_C^{(i)}))\}$ composée de C tuples de transition locale de $\mathcal{T}_\epsilon^{(i)}$, qui respectent les règles suivantes :*

1. $\varrho_{e_1} < \varrho_{e_2} < \ldots < \varrho_{e_C}$
2. $\forall j \in [2..C]$
   $x_j^{(i)} = y_{j-1}^{(i)}$
3. $\nexists (e, \pi_e(x^{(i)}, y^{(i)})) \in \mathcal{T}_\epsilon^{(i)}$ *tel que*
   $\varrho_e < \varrho_{e_1}$ *et* $y^{(i)} = x_1^{(i)}$
4. $\nexists (e, \pi_e(x^{(i)}, y^{(i)})) \in \mathcal{T}_\epsilon^{(i)}$ *tel que*
   $\varrho_{e_C} < \varrho_e$ *et* $x^{(i)} = y_C^{(i)}$
5. $\forall j \in [1..C-1], \nexists (e, \pi_e(x^{(i)}, y^{(i)})) \in \mathcal{T}_\epsilon^{(i)}$ *tel que*
   $\varrho_{e_j} < \varrho_e < \varrho_{e_{j+1}}$ *et* $y_j^{(i)} = x^{(i)}$

Les chaînes de transitions locales d'intérêt sont les chaînes qui prennent leurs événements dans un ensemble $\epsilon = pot(x^{(i)})$ et qui réalisent un enchaînement maximal de tuples de transitions locales des événements potentiellement réalisables à partir d'un état $x^{(i)}$.

De façon similaire aux chaînes de transitions globales, on peut représenter une chaîne de transitions locales par un produit de simultanéité de ses tuples de transitions locales.

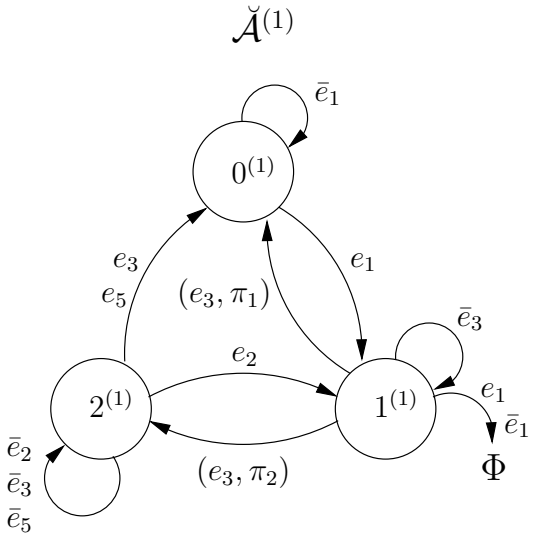
Définition 10.2.2. *Soit une chaîne de transitions locales $l = \{(e_1, \pi_{e_1}(x_1^{(i)}, y_1^{(i)})), \ldots, (e_{C_l}, \pi_{e_C}(x_{C_l}^{(i)}, y_{C_l}^{(i)}))\}$ composée de C_l tuples de transitions locales de l'automate complété $\breve{\mathcal{A}}^{(i)}$, on peut représenter cette chaîne par un produit de simultanéité de ses tuples de transition. Alors, la chaîne de transitions locales l peut être exprimée par une chaîne de transitions locales ou par un produit de simultanéité :*

$$l = \{(e_1, \pi_{e_1}(x_1^{(i)}, y_1^{(i)})), \ldots, (e_{C_l}, \pi_{e_C}(x_{C_l}^{(i)}, y_{C_l}^{(i)}))\} \tag{10.4}$$

ou bien,

$$l = \overset{C_l}{\underset{j=1}{\cdot}}(e_j, \pi_{e_j}(x_j^{(i)}, y_j^{(i)})) \tag{10.5}$$

Dans FIG. 10.1, on présente un exemple d'un automate complété $\breve{\mathcal{A}}^{(1)}$. Cet automate possède trois états, quatre événements (e_1, e_2, e_3, et e_5) et quatre événements complémentaires ($\bar{e}_1$, $\bar{e}_2$, $\bar{e}_3$, et $\bar{e}_5$).



Événement	Prob.	Priorité
e_1	ρ_1	3
e_2	ρ_2	1
e_3	ρ_3	2
e_5	ρ_5	5

Événement	Prob.	Priorité
$\bar{e}_1$	$1 - \rho_1$	3
$\bar{e}_2$	$1 - \rho_2$	1
$\bar{e}_3$	$1 - \rho_3$	2
$\bar{e}_5$	$1 - \rho_5$	5

FIG. 10.1 – Exemple d'un automate complété

Pour l'automate complété $\breve{\mathcal{A}}^{(1)}$ présenté dans FIG. 10.1, pour chaque état local $x^{(i)} \in \mathcal{S}^{(i)}$ de l'automate, on a l'ensemble d'événements $\epsilon = pot(x^{(i)})$ (voir Définition 7.2.8, page 121). À partir de l'ensemble d'événements ϵ relatif à chaque état de l'automate complété $\breve{\mathcal{A}}^{(1)}$, on peut obtenir pour cet automate les chaînes de transitions locales présentées dans TAB. 10.1.

État de départ $0^{(1)}$; $\epsilon = pot(0^{(1)}) = \{e_1, \bar{e}_1\}$	
État d'arrivée	Chaînes de transitions locales
$0^{(1)}$	$\{(\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)}))\}$
$1^{(1)}$	$\{(e_1, \pi_{e_1}(0^{(1)}, 1^{(1)}))\}$

État de départ $1^{(1)}$; $\epsilon = pot(1^{(1)}) = \{e_3, \bar{e}_3, e_1, \bar{e}_1\}$	
État d'arrivée	Chaînes de transitions locales
$0^{(1)}$	$\{(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})); (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)}))\}$
$1^{(1)}$	$\{(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})); (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)}))\}$
$1^{(1)}$	$\{(\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)}))\}$
$2^{(1)}$	$\{(e_3, \pi_{e_3}(1^{(1)}, 2^{(1)}))\}$

État de départ $2^{(1)}$; $\epsilon = pot(2^{(1)}) = \{e_2, \bar{e}_2, e_3, \bar{e}_3, e_5, \bar{e}_5\}$	
État d'arrivée	Chaînes de transitions locales
$0^{(1)}$	$\{(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})); (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)}))\}$
$0^{(1)}$	$\{(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})); (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})); (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)}))\}$
$0^{(1)}$	$\{(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})); (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)}))\}$
$0^{(1)}$	$\{(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})); (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})); (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)}))\}$
$1^{(1)}$	$\{(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})); (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)}))\}$
$2^{(1)}$	$\{(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})); (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})); (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)}))\}$
$2^{(1)}$	$\{(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})); (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})); (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)}))\}$

TAB. 10.1 – Les chaînes de transitions locales de l'automate complété $\breve{\mathcal{A}}^{(1)}$ de FIG. 10.1

Utilisant les chaînes de transitions locales, pour chaque automate complété $\breve{\mathcal{A}}^{(i)}$ d'un modèle SAN, on peut obtenir une *matrice d'événements* $P^{(i)}$. **Les éléments d'une matrice d'événements $P^{(i)}$ sont des produits de simultanéité des tuples de transition des chaînes de transitions locales**.

✍ **Notation**

$\mathcal{L}_{\epsilon}^{(i)}$ l'ensemble des chaînes de transitions locales de l'ensemble d'événements ϵ de l'automate complété $\breve{\mathcal{A}}^{(i)}$. On s'intéressera typiquement à un ϵ de type $pot(x^{(i)})$;

$\mathcal{L}^{(i)}(x^{(i)}, y^{(i)})$ l'ensemble de chaînes de transitions locales $l \in \mathcal{L}^{(i)}_{pot(x^{(i)})}$ tel que $x^{(i)}_{1_l} = x^{(i)}$ et $y^{(i)}_{C_l} = y^{(i)}$, où C_l est le nombre de tuples de transition locale de la chaîne de transitions locales l ;

e_{j_l} l'événement du j-ème tuple de transition de la chaîne de transitions locales l ;

$\pi_{e_{j_l}}(x^{(i)}_{j_l}, y^{(i)}_{j_l})$ la probabilité de routage pour la transition de l'état local $x^{(i)}_{j_l} \in \mathcal{S}^{(i)}$ vers l'état local $y^{(i)}_{j_l} \in \mathcal{S}^{(i)}$ de l'automate complété $\breve{\mathcal{A}}^{(i)}$ déclenchée par l'événement e_{j_l} du j-ème tuple de transition de la chaîne de transitions locales l.

Définition 10.2.3. *Les éléments de la matrice locale d'événements $P^{(i)}$ de l'automate complété $\breve{\mathcal{A}}^{(i)}$ sont définis par :*

$$\forall x^{(i)} \in \mathcal{S}^{(i)}, \forall y^{(i)} \in \mathcal{S}^{(i)},$$
$$P^{(i)}(x^{(i)}, y^{(i)}) = \sum_{\forall l \in \mathcal{L}^{(i)}(x^{(i)}, y^{(i)})} \prod_{j=1}^{C_l} \left(e_{j_l}, \pi_{e_{j_l}}(x_{j_l}^{(i)}, y_{j_l}^{(i)}) \right)$$

Cette définition exprime que la transition de $x^{(i)}$ vers $y^{(i)}$, peut être réalisée par l'une quelconque des chaînes de $\mathcal{L}^{(i)}_{pot(x^{(i)})}$ qui amènent à $y^{(i)}$, *i.e.*, de $\mathcal{L}^{(i)}(x^{(i)}, y^{(i)})$. Une chaîne de transitions locales est représentée par un *produit de simultanéité* ("$\cdot$") des C tuples de transition de cette chaîne. L'élément de la ligne $x^{(i)}$ et colonne $y^{(i)}$ de la matrice d'événements $P^{(i)}$ est une *somme de choix* ("$+$") de toutes les chaînes de transitions locales dont l'état de départ du premier tuple de transition locale ($x_1^{(i)}$) est égal à l'état $x^{(i)}$ et l'état d'arrivée du dernier tuple de transition locale ($y_C^{(i)}$) est égal à l'état $y^{(i)}$.

Pour l'automate complété $\breve{\mathcal{A}}^{(1)}$ présenté dans FIG. 10.1, à partir des chaînes de transitions locales de TAB. 10.1, on peut obtenir la matrice d'événements $P^{(1)}$:

$$P^{(1)} = \left(\begin{array}{c|c|c} (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) & (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) & O_{\bar{\mathcal{T}}} \\ \hline (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) & \begin{array}{c}(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) \\ + \\ (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)}))\end{array} & (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \\ \hline \begin{array}{c}(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \\ + \\ (e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) \\ + \\ (\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)})) \\ + \\ (\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)}))\end{array} & (e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) & \begin{array}{c}(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) \\ + \\ (e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)}))\end{array} \end{array} \right)$$

Dans la section suivante, on présente le *descripteur discret* d'un modèle SAN à temps discret utilisant les matrices d'événements définies précédemment.

10.3 Descripteur discret

Le *descripteur discret* est une façon compacte de représenter les transitions d'un modèle SAN à temps discret. Le descripteur discret d'un modèle SAN est représenté par une formule tensorielle complexe (*i.e.*, via l'algèbre tensorielle complexe défini dans le chapitre 9) utilisant les matrices d'événements de ce modèle.

En utilisant l'algèbre tensorielle complexe, les *événements* (Définition 7.2.3, page 118) d'un modèle SAN à temps discret sont des *éléments actifs* (Définition 9.1.1, page 159), et les *événements complémentaires* (Définition 7.2.9, page 122) du modèle sont des *événements complémentaires* (Définition 9.1.1, page 159) de l'algèbre tensorielle complexe. De plus, les *tuples de transition* (Définition 7.2.4, page 119) d'un modèle SAN sont de *couple d'éléments* (Définition 9.2.1, page 165) de l'algèbre tensorielle complexe.

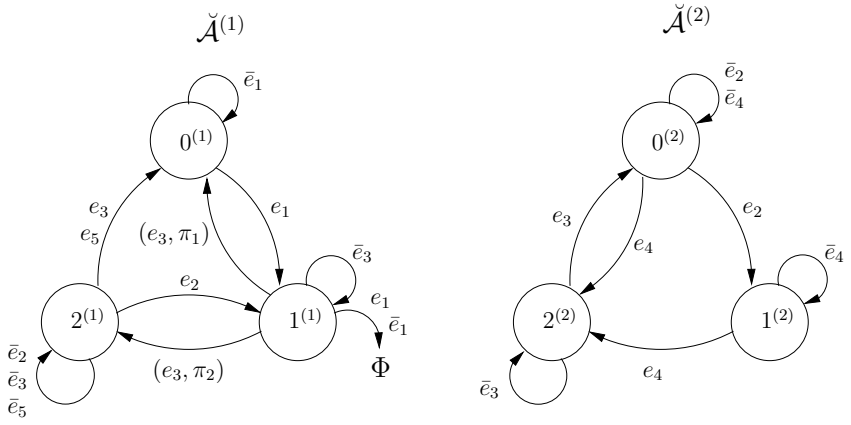
Définition 10.3.1. *Étant donné un SAN complété $(\breve{\mathcal{E}}, \breve{\mathcal{A}}^{(i)}, \hat{\mathcal{S}}, \mathcal{F})$, où $i \in [1..N]$, et pour chaque automate $\breve{\mathcal{A}}^{(i)}$ sa matrice locale d'événements $P^{(i)}$, alors le descripteur discret P est défini par la formule tensorielle complexe :*

$$P = \bigoplus_{i=1}^{N} P^{(i)} \tag{10.6}$$

qu'est égal à la matrice de transition $\mathcal{G}$ de l'automate global de ce modèle comme définie dans la définition 7.3.4 (page 135).

Le lecteur peut trouver dans [15] la démonstration de l'égalité entre le descripteur discret P d'un modèle SAN à temps discret et la matrice de transition $\mathcal{G}$ de l'automate global de ce modèle.

Ensuite, on va présenter, pas à pas, un exemple d'obtention du descripteur discret. Pour illustrer la procédure d'obtention du descripteur, on va reprendre l'exemple d'un réseau d'automate complété (FIG. 7.12, page 130) présenté au chapitre 7. On rappelle ce modèle dans FIG. 10.2.



Événement	Prob.	Priorité
e_1	ρ_1	3
e_2	ρ_2	1
e_3	ρ_3	2
e_4	ρ_4	4
e_5	f	5

Événement	Prob.	Priorité
$\bar{e}_1$	$1-\rho_1$	3
$\bar{e}_2$	$1-\rho_2$	1
$\bar{e}_3$	$1-\rho_3$	2
$\bar{e}_4$	$1-\rho_4$	4
$\bar{e}_5$	$1-f$	5

$f = (st\ \mathcal{A}^{(2)} == 2^{(2)}) \times \rho_5$

FIG. 10.2 – Exemple d'un réseau d'automates complétés

On remarque que l'automate $\breve{\mathcal{A}}^{(1)}$ est le même que celui de FIG. 10.1, donc on va présenter ici uniquement l'obtention de la matrice d'événements de l'automate $\breve{\mathcal{A}}^{(2)}$, car l'obtention de la matrice d'événements $P^{(1)}$ de l'automate $\breve{\mathcal{A}}^{(1)}$ a été déjà présentée dans la section 10.2 (page 186).

On va commencer par définir l'ensemble d'événements possibles ($pot(x^{(i)})$) pour chaque état de l'automate $\breve{\mathcal{A}}^{(2)}$. À partir du $pot(x^{(i)})$, on peut obtenir pour cet automate les chaînes de transitions locales de cet automate. Ces chaînes de transitions sont listées dans TAB. 10.2.

État de départ $0^{(2)}$; $pot(0^{(2)}) = \{e_2; \bar{e}_2; e_4; \bar{e}_4; \}$	
État d'arrivée	Chaînes de transitions locales
$0^{(2)}$	$\{(\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})); (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))\}$
$1^{(2)}$	$\{(e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})); (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))\}$
$2^{(2)}$	$\{(e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})); (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))\}$
$2^{(2)}$	$\{(\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})); (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))\}$

État de départ $1^{(2)}$; $pot(1^{(2)}) = \{e_4; \bar{e}_4\}$	
État d'arrivée	Chaînes de transitions locales
$1^{(2)}$	$\{(\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))\}$
$2^{(2)}$	$\{(e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))\}$

État de départ $2^{(2)}$; $pot(2^{(2)}) = \{e_3; \bar{e}_3\}$	
État d'arrivée	Chaînes de transitions locales
$0^{(2)}$	$\{(e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))\}$
$2^{(2)}$	$\{(\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))\}$

TAB. 10.2 – Les chaînes de transitions locales de l'automate complété $\breve{\mathcal{A}}^{(2)}$ de FIG. 10.2

On rappelle que les éléments d'une matrice d'événements sont des produits de simultanéité des tuples de transition des chaînes de transitions locales et sont définis selon la définition 10.2.3 (page 185).

Pour l'automate complété $\breve{\mathcal{A}}^{(2)}$ de FIG. 10.2, à partir des chaînes de transitions locales présentées dans TAB. 10.2, on obtient la matrice d'événements $P^{(2)}$ suivante :

$$P^{(2)} = \left(\begin{array}{c|c|c} (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)})) & (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)})) & \begin{array}{c}(e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)})) \\ + \\ (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))\end{array} \\ \hline O_{\breve{\mathcal{T}}} & (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)})) & (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)})) \\ \hline (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)})) & O_{\breve{\mathcal{T}}} & (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)})) \end{array} \right)$$

Le descripteur P est défini par le produit tensoriel complexe des matrices d'événements du modèle. Pour le réseau d'automates complété de FIG. 10.2, on a le descripteur suivant :

$$P = P^{(1)} \circledast P^{(2)}$$

Étant donné la taille de la matrice P du descripteur, on va présenter les pas intermédiaires de la construction de la matrice par bloc. En appliquant le produit tensoriel complexe sur les matrices d'événements $P^{(1)}$ et $P^{(2)}$, les blocs on la forme suivante :

$$P = \left(\begin{array}{c|c|c} p^{(1)}_{0^{(1)},0^{(1)}} * P^{(2)} & p^{(1)}_{0^{(1)},1^{(1)}} * P^{(2)} & p^{(1)}_{0^{(1)},2^{(1)}} * P^{(2)} \\ \hline p^{(1)}_{1^{(1)},0^{(1)}} * P^{(2)} & p^{(1)}_{1^{(1)},1^{(1)}} * P^{(2)} & p^{(1)}_{1^{(1)},2^{(1)}} * P^{(2)} \\ \hline p^{(1)}_{2^{(1)},0^{(1)}} * P^{(2)} & p^{(1)}_{2^{(1)},1^{(1)}} * P^{(2)} & p^{(1)}_{2^{(1)},2^{(1)}} * P^{(2)} \end{array}\right)$$

Le bloc $p^{(1)}_{0^{(1)},0^{(1)}} * P^{(2)} = (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * P^{(2)}$ dans la position $\left(\begin{smallmatrix}\bullet & \\ & \end{smallmatrix}\right)$ a les éléments suivants :

$(\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$	$(\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ $+$ $(\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))$
$(\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * O_{\breve{\mathcal{T}}}$	$(\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$
$(\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$	$(\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * O_{\breve{\mathcal{T}}}$	$(\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$

Par la propriété d'idempotence (Propriété 9.2.1, page 168), les éléments de ce bloc de la matrice sont réduits à :

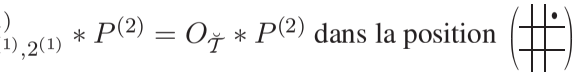
$(\bar{e}_1, \Pi_{\bar{e}_1}(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)})) \cdot (\bar{e}_4, \Pi_{\bar{e}_4}(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}))$	$O_{\breve{\mathcal{T}}}$	$(\bar{e}_1, \Pi_{\bar{e}_1}(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)})) \cdot (e_4, \Pi_{e_4}(0^{(1)}0^{(2)}, 0^{(1)}2^{(2)}))$
$O_{\breve{\mathcal{T}}}$	$(\bar{e}_1, \Pi_{\bar{e}_1}(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)})) \cdot (\bar{e}_4, \Pi_{\bar{e}_4}(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)}))$	$(\bar{e}_1, \Pi_{\bar{e}_1}(0^{(1)}1^{(2)}, 0^{(1)}1^{(2)})) \cdot (e_4, \Pi_{e_4}(0^{(1)}1^{(2)}, 0^{(1)}2^{(2)}))$
$O_{\breve{\mathcal{T}}}$	$O_{\breve{\mathcal{T}}}$	$(\bar{e}_1, \Pi_{\bar{e}_1}(0^{(1)}2^{(2)}, 0^{(1)}2^{(2)}))$

Le bloc $p^{(1)}_{0^{(1)},1^{(1)}} * P^{(2)} = (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * P^{(2)}$ dans la position $\left(\begin{smallmatrix} & \bullet & \\ & & \\ & & \end{smallmatrix}\right)$ a les éléments suivants :

$(e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$	$(e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ $+$ $(e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))$
$(e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * O_{\breve{\mathcal{I}}}$	$(e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$
$(e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$	$(e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * O_{\breve{\mathcal{I}}}$	$(e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$

Par la propriété d'idempotence (Propriété 9.2.1, page 168), les éléments de ce bloc de la matrice sont réduits à :

$(e_1, \Pi_{e_1}(0^{(1)}0^{(2)}, 1^{(1)}0^{(2)})) \cdot (\bar{e}_4, \Pi_{\bar{e}_4}(1^{(1)}0^{(2)}, 1^{(1)}0^{(2)}))$	$O_{\breve{\mathcal{I}}}$	$(e_1, \Pi_{e_1}(0^{(1)}0^{(2)}, 1^{(1)}0^{(2)})) \cdot (e_4, \Pi_{e_4}(1^{(1)}0^{(2)}, 1^{(1)}2^{(2)}))$
$O_{\breve{\mathcal{I}}}$	$(e_1, \Pi_{e_1}(0^{(1)}1^{(2)}, 1^{(1)}1^{(2)})) \cdot (\bar{e}_4, \Pi_{\bar{e}_4}(1^{(1)}1^{(2)}, 1^{(1)}1^{(2)}))$	$(e_1, \Pi_{e_1}(0^{(1)}1^{(2)}, 1^{(1)}1^{(2)})) \cdot (e_4, \Pi_{e_4}(1^{(1)}1^{(2)}, 1^{(1)}2^{(2)}))$
$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$	$(e_1, \Pi_{e_1}(0^{(1)}2^{(2)}, 1^{(1)}2^{(2)}))$

Le bloc $p^{(1)}_{0^{(1)},2^{(1)}} * P^{(2)} = O_{\breve{\mathcal{I}}} * P^{(2)}$ dans la position  a les éléments suivants :

$O_{\breve{\mathcal{I}}} * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$	$O_{\breve{\mathcal{I}}} * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$O_{\breve{\mathcal{I}}} * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ $+$ $O_{\breve{\mathcal{I}}} * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))$
$O_{\breve{\mathcal{I}}} * O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}} * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$O_{\breve{\mathcal{I}}} * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$
$O_{\breve{\mathcal{I}}} * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$	$O_{\breve{\mathcal{I}}} * O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}} * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$

Par la propriété (2) de la définition 9.2.9 (page 167), les éléments de ce bloc de la matrice sont réduits à :

$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$
$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$
$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$

Prenons maintenant les blocs de la deuxième ligne. Le bloc $p^{(1)}_{1^{(1)},0^{(1)}} * P^{(2)} = (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * P^{(2)}$ dans la position  a les éléments suivants :

$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$	$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 1^{(2)}))$	$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ $+$ $(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))$
$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * O_{\tilde{\mathcal{T}}}$	$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$
$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$	$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * O_{\tilde{\mathcal{T}}}$	$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (\bar{e}_1, \pi_{\bar{e}_1}(0^{(1)}, 0^{(1)})) * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$

Par la propriété d'idempotence (Propriété 9.2.1, page 168), les éléments de ce bloc de la matrice sont réduits à :

$O_{\check{\mathcal{T}}}$	$O_{\check{\mathcal{T}}}$	$O_{\check{\mathcal{T}}}$
$O_{\check{\mathcal{T}}}$	$O_{\check{\mathcal{T}}}$	$O_{\check{\mathcal{T}}}$
$(e_3, \Pi_{e_3}(1^{(1)}2^{(2)}, 0^{(1)}0^{(2)})) \cdot (\bar{e}_1, \Pi_{\bar{e}_1}(0^{(1)}0^{(2)}, 0^{(1)}0^{(2)}))$	$O_{\check{\mathcal{T}}}$	$O_{\check{\mathcal{T}}}$

Le bloc $p^{(1)}_{1^{(1)},1^{(1)}} * P^{(2)} = \big((e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) + (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)}))\big) * P^{(2)} = (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * P^{(2)} + (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * P^{(2)}$ dans la position $\left(\begin{smallmatrix} + & + & + \\ + & \bullet & + \\ + & + & + \end{smallmatrix}\right)$ a les éléments suivants :

$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$ $+$ $(\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$	$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$ $+$ $(\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ $+$ $(\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ $+$ $(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))$ $+$ $(\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))$
$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * O_{\check{\mathcal{T}}}$ $+$ $(\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * O_{\check{\mathcal{T}}}$	$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$ $+$ $(\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ $+$ $(\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$
$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$ $+$ $(\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$	$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * O_{\check{\mathcal{T}}}$ $+$ $(\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * O_{\check{\mathcal{T}}}$	$(e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) \cdot (e_1, \pi_{e_1}(0^{(1)}, 1^{(1)})) * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$ $+$ $(\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$

Par la propriété d'idempotence (Propriété 9.2.1, page 168), les éléments de ce bloc de la matrice sont réduits à :

$(\bar{e}_4, \Pi_{\bar{e}_4}(1^{(1)}0^{(2)}, 1^{(1)}0^{(2)}))$	$O_{\check{\mathcal{T}}}$	$(e_4, \Pi_{e_4}(1^{(1)}0^{(2)}, 1^{(1)}2^{(2)}))$
$O_{\check{\mathcal{T}}}$	$(\bar{e}_4, \Pi_{\bar{e}_4}(1^{(1)}1^{(2)}, 1^{(1)}1^{(2)})$	$(e_4, \Pi_{e_4}(1^{(1)}1^{(2)}, 1^{(1)}2^{(2)}))$
$(e_3, \Pi_{e_3}(1^{(1)}2^{(2)}, 0^{(1)}0^{(2)})) \cdot (e_1, \Pi_{e_1}(0^{(1)}0^{(2)}, 1^{(1)}0^{(2)}))$	$O_{\check{\mathcal{T}}}$	$(\bar{e}_3, \Pi_{\bar{e}_3}(1^{(1)}2^{(2)}, 1^{(1)}2^{(2)}))$

Le bloc $p^{(1)}_{1^{(1)},2^{(1)}} * P^{(2)} = (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) * P^{(2)}$ dans la position $\left(\begin{smallmatrix}+&+&+\\+&+&\bullet\\+&+&+\end{smallmatrix}\right)$ a les éléments suivants :

$(e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$	$(e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ $+$ $(e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))$
$(e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) * O_{\breve{\mathcal{I}}}$	$(e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$
$(e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$	$(e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) * O_{\breve{\mathcal{I}}}$	$(e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$

Par la propriété d'idempotence (Propriété 9.2.1, page 168), les éléments de ce bloc de la matrice sont réduits à :

$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$
$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$
$(e_3, \pi_{e_3}(1^{(1)}2^{(2)}, 2^{(1)}0^{(2)}))$	$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$

Prenons maintenant les blocs qui composent la troisième ligne. Le bloc $p^{(1)}_{2^{(1)},0^{(1)}} * P^{(2)} = \big((e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) + (e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) + (\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)})) + (\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)}))\big) * P^{(2)}$.

Par la propriété de distributivité de l'opérateur de concurrence sur l'opérateur de choix, on a $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) * P^{(2)} + (e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * P^{(2)} + (\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)})) * P^{(2)} + (\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * P^{(2)}$ dans la position $\left(\begin{smallmatrix} \cdot & | & | \\ | & | & | \end{smallmatrix}\right)$. Les les éléments de ce bloc sont les suivants :

$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$ + $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$ + $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ + $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ + $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ + $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))$
$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) * O_{\bar{T}}$ + $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * O_{\bar{T}}$ + $((\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)})) * O_{\bar{T}}$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * O_{\bar{T}}$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$ + $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)})) * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ + $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)})) * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$
$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$ + $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)})) * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) * O_{\bar{T}}$ + $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * O_{\bar{T}}$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)})) * O_{\bar{T}}$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * O_{\bar{T}}$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 0^{(1)})) * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$ + $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (e_3, \pi_{e_3}(2^{(1)}, 0^{(1)})) * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (e_5, \pi_{e_5}(2^{(1)}, 0^{(1)})) * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$

Par la propriété d'idempotence (Propriété 9.2.1, page 168), les éléments de ce bloc de la matrice sont réduits à :

$(\bar{e}_2, \Pi_{\bar{e}_2}(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)})) \cdot (\bar{e}_4, \Pi_{\bar{e}_4}(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)})) \cdot (e_5, \Pi_{e_5}(2^{(1)}0^{(2)}, 0^{(1)}0^{(2)}))$	$O_{\breve{\mathcal{I}}}$	$(\bar{e}_2, \Pi_{\bar{e}_2}(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)})) \cdot (e_4, \pi_{e_4}(2^{(1)}0^{(2)}, 2^{(1)}2^{(2)})) \cdot (e_5, \Pi_{e_5}(2^{(1)}2^{(2)}, 0^{(1)}2^{(2)}))$
$O_{\breve{\mathcal{I}}}$	$(\bar{e}_4, \Pi_{\bar{e}_4}(2^{(1)}1^{(2)}, 2^{(1)}1^{(2)})) \cdot (e_5, \Pi_{e_5}(2^{(1)}1^{(2)}, 0^{(1)}1^{(2)}))$	$(e_4, \Pi_{e_4}(2^{(1)}1^{(2)}, 2^{(1)}2^{(2)})) \cdot (e_5, \Pi_{e_5}(2^{(1)}2^{(2)}, 0^{(1)}2^{(2)}))$
$(e_3, \Pi_{e_3}(2^{(1)}2^{(2)}, 0^{(1)}0^{(2)}))$	$O_{\breve{\mathcal{I}}}$	$(\bar{e}_3, \Pi_{\bar{e}_3}(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)})) \cdot (e_5, \Pi_{e_5}(2^{(1)}2^{(2)}, 0^{(1)}2^{(2)}))$

Le bloc $p^{(1)}_{2^{(1)},1^{(1)}} * P^{(2)} = (e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * P^{(2)}$dans la position  a les éléments suivants :

$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ + $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))$
$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * O_{\breve{\mathcal{I}}}$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$
$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * O_{\breve{\mathcal{I}}}$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(1^{(1)}, 1^{(1)})) * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$

Par la propriété d'idempotence (Propriété 9.2.1, page 168), les éléments de ce bloc de la matrice sont réduits à :

$O_{\breve{\mathcal{I}}}$	$(e_2, \Pi_{e_2}(2^{(1)}0^{(2)}, 1^{(1)}1^{(2)})) \cdot (\bar{e}_4, \Pi_{\bar{e}_4}(1^{(1)}1^{(2)}, 1^{(1)}1^{(2)}))$	$(e_2, \Pi_{e_2}(2^{(1)}0^{(2)}, 1^{(1)}1^{(2)})) \cdot (e_4, \Pi_{e_4}(1^{(1)}1^{(2)}, 1^{(1)}2^{(2)}))$
$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$
$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$	$O_{\breve{\mathcal{I}}}$

Le bloc $p^{(1)}_{2^{(1)},2^{(1)}} * P^{(2)} =$
$\big((e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) + (e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)}))\big) * P^{(2)}$.

Par la propriété de distributivité de l'opérateur de concurrence sur l'opérateur de choix, on a
$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * P^{(2)} + (e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * P^{(2)}$ dans la position $\left(\begin{smallmatrix} + & + \\ + & \bullet \end{smallmatrix}\right)$. Les les éléments de ce bloc sont les suivants :

$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(0^{(2)}, 0^{(2)}))$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (e_2, \pi_{e_2}(0^{(2)}, 1^{(2)})) \cdot (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ + $(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (\bar{e}_2, \pi_{\bar{e}_2}(0^{(2)}, 0^{(2)})) \cdot (e_4, \pi_{e_4}(0^{(2)}, 2^{(2)}))$
$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * O_{\tilde{T}}$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * O_{\tilde{T}}$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (\bar{e}_4, \pi_{\bar{e}_4}(1^{(2)}, 1^{(2)}))$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (e_4, \pi_{e_4}(1^{(2)}, 2^{(2)}))$
$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (e_3, \pi_{e_3}(2^{(2)}, 0^{(2)}))$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * O_{\tilde{T}}$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * O_{\tilde{T}}$	$(e_2, \pi_{e_2}(2^{(1)}, 1^{(1)})) \cdot (e_3, \pi_{e_3}(1^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$ + $(\bar{e}_2, \pi_{\bar{e}_2}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_3, \pi_{\bar{e}_3}(2^{(1)}, 2^{(1)})) \cdot (\bar{e}_5, \pi_{\bar{e}_5}(2^{(1)}, 2^{(1)})) * (\bar{e}_3, \pi_{\bar{e}_3}(2^{(2)}, 2^{(2)}))$

Par la propriété d'idempotence (Propriété 9.2.1, page 168), les éléments de ce bloc de la matrice sont réduits à :

$(\bar{e}_2, \Pi_{\bar{e}_2}(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)})) \cdot (\bar{e}_4, \Pi_{\bar{e}_4}(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)})) \cdot (\bar{e}_5, \Pi_{\bar{e}_5}(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)}))$	$O_{\tilde{T}}$	$(\bar{e}_2, \Pi_{\bar{e}_2}(2^{(1)}0^{(2)}, 2^{(1)}0^{(2)})) \cdot (e_4, \Pi_{e_4}(2^{(1)}0^{(2)}, 2^{(1)}2^{(2)})) \cdot (\bar{e}_5, \Pi_{\bar{e}_5}(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}))$
$O_{\tilde{T}}$	$(\bar{e}_4, \Pi_{\bar{e}_4}(2^{(1)}1^{(2)}, 2^{(1)}1^{(2)})) \cdot (\bar{e}_5, \Pi_{\bar{e}_5}(2^{(1)}1^{(2)}, 2^{(1)}1^{(2)}))$	$(e_4, \Pi_{e_4}(2^{(1)}1^{(2)}, 2^{(1)}2^{(2)})) \cdot (\bar{e}_5, \Pi_{\bar{e}_5}(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}))$
$O_{\tilde{T}}$	$O_{\tilde{T}}$	$(\bar{e}_3, \Pi_{\bar{e}_3}(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)})) \cdot (\bar{e}_5, \Pi_{\bar{e}_5}(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}))$

À partir des chaînes de transitions globales du descripteur discret P (égal à la matrice de transition de l'automate global), on peut obtenir la matrice transition de la chaîne de Markov en évaluant les chaînes de transitions globales selon la définition 7.4.1 (page 136).

Évaluons par exemple la chaîne de transitions globales de la dernière ligne et dernière colonne (position $P_{[2^{(1)},2^{(2)}][2^{(1)},2^{(2)}]}$. Cette chaîne de transitions globales est égale à :

$$P_{[2^{(1)},2^{(2)}][2^{(1)},2^{(2)}]} = (\bar{e}_3, \Pi_{\bar{e}_3}(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)})) \cdot (\bar{e}_5, \Pi_{\bar{e}_5}(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}))$$

Évaluons chaque tuple de transition globale séparément. Le premier tuple de transition globale $(\bar{e}_3, \Pi_{\bar{e}_3}(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}))$ a la probabilité d'occurrence de l'événement $\bar{e}_3$ égale à $1 - \rho_3$. Cet événement a une probabilité d'occurrence constante $(1 - \rho_3)$, *i.e.*, sa valeur ne dépende pas de l'état global du modèle. La probabilité de routage globale (produit des probabilités de routage des tuples de transition locale) de ce tuple de transition globale est aussi constante et égale à 1.

Le deuxième tuple de transition globale $(\bar{e}_5, \Pi_{\bar{e}_5}(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)}))$ a la probabilité d'occurrence de l'événement $\bar{e}_5$ égale à $1 - f$, où f est la fonction :

$$f = (st\ \mathcal{A}^{(2)}\ == 2^{(2)}) \times \rho_5$$

En évaluant cette fonction pour l'état globale $2^{(1)}2^{(2)}$ (état de départ du tuple de transition globale), on obtient :

$$f = \rho_5$$

Alors, la probabilité d'occurrence de ce tuple de transition globale, lorsque sa fonction est évaluée pour l'état globale $2^{(1)}2^{(2)}$, est égale à $1 - \rho_5$. La probabilité de routage de ce tuple de transition globale est constante et égale à 1.

En évaluant la chaîne de transitions globales de la position $P_{[2^{(1)},2^{(2)}][2^{(1)},2^{(2)}]}$ (notée par g) selon la définition 7.4.1 (page 136), on a l'évaluation suivante :

$$\begin{aligned}\Pi_g &= \Big((1 - \rho_3(2^{(1)}2^{(2)})) \times \Pi_{\bar{e}_3}(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)})(2^{(1)}2^{(2)})\Big) \times \\ &\quad \Big((1 - f(2^{(1)}2^{(2)})) \times \Pi_{\bar{e}_5}(2^{(1)}2^{(2)}, 2^{(1)}2^{(2)})(2^{(1)}2^{(2)})\Big) \\ &= (1 - \rho_3)(1 - \rho_5)\end{aligned}$$

En évaluant toutes les chaînes de transitions globales du descripteur P du modèle de FIG. 10.2 (page 187), on obtient la matrice de transition suivante :

$$P = \left(\begin{array}{ccc|ccc|ccc}
(1-\rho_1)(1-\rho_4) & 0 & (1-\rho_1)\rho_4 & \rho_1(1-\rho_4) & 0 & \rho_1\rho_4 & 0 & 0 & 0 \\
0 & (1-\rho_1)(1-\rho_4) & (1-\rho_1)\rho_4 & 0 & \rho_1(1-\rho_4) & \rho_1\rho_4 & 0 & 0 & 0 \\
0 & 0 & (1-\rho_1) & 0 & 0 & \rho_1 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & (1-\rho_4) & 0 & \rho_4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & (1-\rho_4) & \rho_4 & 0 & 0 & 0 \\
\rho_3\pi_1(1-\rho_1) & 0 & 0 & \rho_3\pi_1\rho_1 & 0 & (1-\rho_3) & \rho_3\pi_2 & 0 & 0 \\
\hline
0 & 0 & (1-\rho_2)\rho_4\rho_5 & 0 & \rho_2(1-\rho_4) & \rho_2\rho_4 & (1-\rho_2)(1-\rho_4) & 0 & (1-\rho_2)\rho_4(1-\rho_5) \\
0 & 0 & \rho_4\rho_5 & 0 & 0 & 0 & 0 & (1-\rho_4) & \rho_4(1-\rho_5) \\
\rho_3 & 0 & (1-\rho_3)\rho_5 & 0 & 0 & 0 & 0 & 0 & (1-\rho_3)(1-\rho_5)
\end{array}\right)$$

10.4 Conclusion

Dans ce chapitre, on a présenté une représentation compacte pour la matrice de transition de l'automate global d'un modèle SAN à temps discret. Cette représentation compacte (appelé *descripteur discret*) est une représentation tensorielle qui utilise l'algèbre tensorielle complexe (présentée dans le chapitre 9).

L'association de l'algèbre tensorielle complexe et des règles de construction des chaînes de transitions locales (et par conséquence des matrices d'événements) cache à l'utilisateur toute la complexité d'un modèle à temps discret, ce qui fait de cette association (algèbre tensorielle complexe et algèbre des événements) un outil puissant pour la construction de modèles SAN à temps discret. Plus précisément, les règles de construction des chaînes de transitions locales masquent la complexité de l'occurrence simultanée de plusieurs événements dans un même automate pendant que le produit tensoriel complexe masque la modélisation de l'occurrence d'événements concurrentes (l'occurrence d'événements dans différents automates). Les cas de conflit sont gérés par la priorité et par la définition de l'ensemble d'événements possibles de chaque état.

Dans le chapitre suivant, on exploite la représentation compacte du descripteur discret (*i.e.*, la représentation tensorielle qui utilise l'algèbre tensorielle complexe) afin de proposer une méthode de résolution pour des modèles SAN à temps discret.

Chapitre 11

Multiplication vecteur-descripteur discret

Les systèmes ciblés par le formalisme des Réseaux d'Automates Stochastiques (SAN) sont les systèmes à grand espace d'états et pour ces systèmes les méthodes itératives sont les plus adéquates. La multiplication d'un vecteur de probabilité par le descripteur est l'opération fondamentale de toutes les méthodes itératives et pour le formalisme SAN à temps continu est été largement étudié [55, 9, 56]. Cependant, le formalisme SAN à temps discret a reçu beaucoup moins d'attention.

Dans ce chapitre, on présente une méthode de "multiplication" d'un vecteur de probabilité π par le descripteur discret P d'un modèle SAN à temps discret.

Notons d'emblée que le vecteur de probabilité π et le descripteur discret P sont des objets mathématiques différents. Le descripteur discret P d'un modèle SAN à temps discret, présenté dans le chapitre 10, est une formule tensorielle complexe qui permet de représenter d'une façon compacte les transitions du modèle. C'est pourquoi la terminologie "multiplication" est un abus de langage.

Dans la section suivante, on présente quelques notations et définitions des éléments de base utilisés dans la méthode de "multiplication" proposée dans ce chapitre. Dans la section 11.2, on présente la méthode de "multiplication" d'un vecteur de probabilité par le descripteur discret d'un modèle SAN à temps discret, sans jamais générer la matrice de transition de la chaîne de Markov de ce modèle. On présente, dans la section 11.3, quelques expériences numériques qui doivent être réalisées afin de tester la performance de la méthode. Et finalement on conclut ce chapitre par la section 11.4.

11.1 Notations et définitions

Le descripteur discret P d'un modèle SAN à temps discret est une formule tensorielle qui utilise l'algèbre tensorielle complexe (décrite dans le chapitre 9) pour représenter de façon compacte le modèle. Le descripteur discret permet aisément de générer la chaîne de Markov représentée par le modèle SAN à temps discret.

Dans tout le chapitre, on a un modèle SAN $(\breve{\mathcal{E}}, \breve{\mathcal{A}}^{(i)}, \hat{\mathcal{S}}, \mathcal{F})$ à temps discret, où $i \in [1..N]$, et le descripteur discret $P = \bigcircledast_{i=1}^{N} P^{(i)}$ de ce modèle, $\mathcal{M}(P)$ est la matrice de transition de la chaîne de Markov

obtenue à partir du descripteur discret P de ce modèle. La matrice $\mathcal{M}(P)$ est une matrice carré de dimension $\prod_{i=1}^{N} n_i$, où n_i est la dimension de la matrice d'événements $P^{(i)}$ du descripteur discret P.

Les valeurs des entrées de la matrice de transition $\mathcal{M}(P)$ sont des probabilités d'occurrence des chaînes de transitions globales (Définition 11.1.1). On rappelle qu'une chaîne de transitions globales est obtenue par le produit de concurrence des chaînes de transitions locales des matrices $P^{(i)}$ du descripteur discret P.

✍ **Notation**

$\rho_e(\tilde{x})$	la probabilité d'occurrence de l'événement e évaluée pour l'état global $\tilde{x}$;
$(e, \Pi_e(\tilde{x}, \tilde{y}))$	le tuple de transition globale de l'événement e avec la probabilité de routage globale $\Pi_e(\tilde{x}, \tilde{y})$ de l'état global $\tilde{x}$ vers l'état global $\tilde{y}$;
$\Pi_e(\tilde{x}, \tilde{y})(\tilde{x})$	la probabilité de routage globale $\Pi_e(\tilde{x}, \tilde{y})$ de l'état global $\tilde{x}$ vers l'état global $\tilde{y}$ de l'événement e évaluée pour l'état global $\tilde{x}$.

Définition 11.1.1. *Soit une chaîne de transitions globales* $g = \{(e_1, \Pi_{e_1}(\tilde{x}_1, \tilde{y}_1)), \ldots, (e_C, \Pi_{e_C}(\tilde{x}_C, \tilde{y}_C))\}$ *composée de* C *tuples de transition globale, la probabilité d'occurrence* Π_g *de cette chaîne est :*

$$\Pi_g = \Big(\rho_{e_1}(\tilde{x}_1) \times \Pi_{e_1}(\tilde{x}_1, \tilde{y}_1)(\tilde{x}_1)\Big) \times \cdots \times \Big(\rho_{e_C}(\tilde{x}_C) \times \Pi_{e_C}(\tilde{x}_C, \tilde{y}_C)(\tilde{x}_C)\Big) \tag{11.1}$$

Si on a obtenue la matrice de transition $\mathcal{M}(P)$ de la chaîne de Markov présentée précédemment, on peut calculer l'opération :

$$\pi \times \mathcal{M}(P) = \pi' \tag{11.2}$$

où π est le vecteur de probabilité et π' est le vecteur de probabilité résultant.

Toutefois, en réalisant l'opération de multiplication du vecteur π par la matrice de transition $\mathcal{M}(P)$, on ne profite pas de la représentation compacte du descripteur discret P de façon à calculer la multiplication de π par les matrices opérandes du descripteur. D'ailleurs, pour des modèles à très grand espace d'états, le coût de stockage de la matrice de transition $\mathcal{M}(P)$ peut être très élevé.

Alors, pour réaliser la "multiplication" du vecteur π par le descripteur discret P, sans jamais générer la matrice de transition $\mathcal{M}(P)$ de ce modèle, on a besoin que le vecteur de probabilité π et le descripteur P soient des objets mathématiques de même nature. Autrement dit, on a besoin que le vecteur de probabilité π soit un vecteur d'objet du même type que les éléments des matrices d'événements $P^{(i)}$ du descripteur P. Il faut garder à l'esprit que les éléments d'une matrice d'événements $P^{(i)}$ (Définition 10.2.3, page 185) sont en fait des chaînes de transitions locales (Définition 10.2.1, page 183) qui représentent l'occurrence simultanée des événements d'un état $x^{(i)}$ vers un état $y^{(i)}$ dans l'automate $\mathcal{A}^{(i)}$.

On va définir un élément, appelé *tuple de probabilité*, afin de permettre que les probabilités du vecteur π puissent réaliser les opérations *produit de simultanéité*, *somme de choix* et *produit de concurrence* de l'algèbre tensorielle complexe avec les éléments des matrices du descripteur.

Définition 11.1.2. *Soit une probabilité $\tau \in [0,1]$, on définit par le couple $(I_{\breve{\mathcal{E}}}, \tau)$ le tuple de probabilité de τ, où $I_{\breve{\mathcal{E}}}$ est l'événement neutre défini dans l'algèbre tensorielle complexe (Définition 9.1.3, page 160). Dans le contexte de l'algèbre tensorielle complexe, le tuple de probabilité a les propriétés suivantes*[1] *:*

1. $e \in \breve{\mathcal{E}}$,
$$(I_{\breve{\mathcal{E}}}, \tau) * (e, \pi_e(x,y)) = (e, \tau \times \pi_e(x,y))$$

2. $e_1 \in \breve{\mathcal{E}}, e_2 \in \breve{\mathcal{E}}$,
$$(I_{\breve{\mathcal{E}}}, \tau) * [(e_1, \pi_{e_1}(x_1,y_1)) \cdot (e_2, \pi_{e_2}(x_2,y_2))] = (e_1, \tau \times \pi_{e_1}(x_1,y_1)) \cdot (e_2, \pi_{e_2}(x_2,y_2))$$
$$= (e_1, \pi_{e_1}(x_1,y_1)) \cdot (e_2, \tau \times \pi_{e_2}(x_2,y_2))$$

3. $e_1 \in \breve{\mathcal{E}}, e_2 \in \breve{\mathcal{E}}$,
$$(I_{\breve{\mathcal{E}}}, \tau) * [(e_1, \pi_{e_1}(x_1,y_1)) + (e_2, \pi_{e_2}(x_2,y_2))] = (e_1, \tau \times \pi_{e_1}(x_1,y_1)) + (e_2, \tau \times \pi_{e_2}(x_2,y_2))$$

On remarque qu'un tuple de probabilité est de la même nature qu'un tuple de transition.

On va donc considérer dans la suite des vecteurs d'événements v qui peuvent avoir des éléments du type :
- *tuples de probabilité* et *tuples de transition* ;
- *expressions* de l'algèbre tensorielle complexe, telles que *produits de concurrence de chaînes de transitions locales* ;
- *chaînes de transitions globales*.

En utilisant le concept du tuple de probabilité, on peut obtenir un vecteur d'événements v (dont les éléments sont initialement de tuples de probabilités) à partir d'un vecteur de probabilité π.

✍ **Notation**

$v[i]$	l'élément dans la position i du vecteur d'événements v ;
$\pi[i]$	l'élément dans la position i du vecteur de probabilité π ;
$\hat{\mathcal{S}}$	l'espace d'états produit du modèle ;

Définition 11.1.3. *Soit un vecteur de probabilité π, on peut obtenir le vecteur d'événements v tel que :*
$$\forall i \in [1..|\hat{\mathcal{S}}|],$$
$$v[i] = (I_{\breve{\mathcal{E}}}, \pi[i])$$

On va étendre la définition de l'attribut de position (Définition 9.2.13, page 169) pour les tuples de probabilité du vecteur d'événements v de façon à ce que :
$$\forall i \in [1..|\hat{\mathcal{S}}|],$$
$$a\Big((I_{\breve{\mathcal{E}}}, \pi[i])\Big) = 0$$

[1] On rappelle que le *produit de simultanéité* est noté par "$\cdot$", la *somme de choix* est notée par "$+$" et le *produit de concurrence* est noté par "$*$". Le produit entre probabilités (valeurs réelles) est noté par "$\times$" et $(e, \pi_e(x,y))$ est le tuple de transition de l'événement e et $\pi_e(x,y) \in [0,1]$ est une probabilité, fonction de x et y, où x et y appartiennent à un espace $\mathcal{S}$.

Cet attribut de position d'un tuple de probabilité a un rôle important dans l'une des optimisations de la méthode présentée dans la section suivante (page 220).

À partir d'un vecteur d'événements v (dont les éléments sont des chaînes de transitions globales), on peut obtenir un vecteur de probabilité π.

✍ **Notation**

g	une chaîne de transitions globales ;
Π_g	la probabilité d'occurrence de la chaîne de transitions globales g (calculée comme décrite dans la définition 11.1.1) ;
Λ_i	l'ensemble de chaînes de transitions globales qui se trouvent dans $v[i]$.

Définition 11.1.4. *Soit un vecteur d'événements v, dont les éléments sont de chaînes de transitions globales, on peut obtenir le vecteur de probabilité π tel que :*

$$\forall i \in [1..|\hat{\mathcal{S}}|],$$
$$\pi[i] = \sum_{\forall g \in \Lambda_i} \Pi_g$$

La probabilité d'un élément du vecteur π est égal à somme des probabilités d'occurrence des chaînes de transitions globales de l'élément du vecteur v.

La procédure *ProbEv* implante la notion présentée dans la définition 11.1.3 et la procédure *EvProb* implante la notion présentée dans la définition 11.1.4. Ces deux procédures sont essentielles pour la méthode de "multiplication" d'un vecteur d'événements par le descripteur discret présentée dans la section 11.2. L'utilisation de ces procédures deviendra claire dans la section suivante.

Dans la section suivante, on présente la méthode de "multiplication" d'un vecteur de probabilité π par le descripteur discret P d'un modèle SAN à temps discret.

11.2 Multiplication des Facteurs normaux

Un des avantages de représenter un modèle SAN à temps discret par un descripteur est la façon compacte par laquelle on peut représenter les transitions du modèle : on remplace une description dans un espace produit par un unique produit tensoriel portant sur des facteurs qui décrivent ce qui se passe sur une seule dimension (une composante du modèle SAN). Afin de profiter de cette représentation, on présente dans cette section la méthode de la "multiplication" d'un vecteur d'événements v par le descripteur discret P du modèle, sans jamais générer la matrice qui représente ce descripteur, *i.e.*, sans jamais générer la matrice de transition $\mathcal{M}(P)$ de la chaîne de Markov du modèle. La méthode vise à profiter de la *propriété de la décomposition du produit tensoriel complexe en facteurs normaux* de façon à ce que la multiplication par un opérateur sur l'espace produit soit remplacée par une suite d'opérations qui manipulent des données de la taille d'une composante (et pour toutes les composantes). Nous verrons ici une démarche analogue aux résultats connus en temps continu.

On rappelle que le descripteur discret P est le produit tensoriel complexe composé d'une suite de N matrices d'événements notées $P^{(i)}$ avec $i \in [1..N]$, chacune associée à un automate complété $\breve{\mathcal{A}}^{(i)}$ d'un modèle SAN à temps discret, *i.e.* :

$$P = \bigcircledast_{i=1}^{N} P^{(i)} \tag{11.3}$$

Ensuite, on rappelle quelques notations pour des suites finies de matrices[2].

✍ **Rappelons**

n_i	la dimension de la i-ème matrice d'une suite ;
$nleft_i$	le produit des dimensions de toutes les matrices à gauche de la i-ème matrice d'une suite, *i.e.*, $\prod_{k=1}^{i-1} n_k$ (cas particulier : $nleft_1 = 1$) ;
$nright_i$	le produit des dimensions de toutes les matrices à droite de la i-ème matrice d'une suite, *i.e.*, $\prod_{k=i+1}^{N} n_k$ (cas particulier : $nright_N = 1$) ;
$\bar{n}_i$	le produit des dimensions de toutes les matrices sauf la i-ème matrice d'une suite, *i.e.*, $\prod_{k=1,k\neq i}^{N} n_k$ ($\bar{n}_i = nleft_i \times nright_i$) ;

Selon la propriété de décomposition des produits tensoriels complexes en facteurs normaux (Propriété 9.3.5, page 177), tout produit tensoriel complexe de N matrices d'événements est égal au produit de concurrence de N facteurs normaux. En utilisant cette propriété pour le terme $\bigcircledast_{i=1}^{N} P^{(i)}$ du descripteur discret, on a :

$$\begin{aligned} P^{(1)} \circledast P^{(2)} \circledast \cdots \circledast P^{(N-1)} \circledast P^{(N)} = & \left(P^{(1)} \circledast I_{nright_1}\right) \\ & * \left(I_{nleft_2} \circledast P^{(2)} \circledast I_{nright_2}\right) \\ & * \cdots \\ & * \left(I_{nleft_{N-1}} \circledast P^{(N-1)} \circledast I_{nright_{N-1}}\right) \\ & * \left(I_{nleft_N} \circledast P^{(N)}\right) \end{aligned} \tag{11.4}$$

Suivant quelques notations du vecteur d'événements v utilisé dans la méthode de "multiplication" du vecteur de probabilité π par les facteurs normaux du descripteur discret.

✍ **Notation**

$v^{(i)}$	le vecteur d'événements résultant du produit de concurrence de v par le facteur normal d'indice i ;
$v^{(0)}$	le vecteur initial d'événements obtenu à partir d'un vecteur de probabilité. Initialement, le vecteur $v^{(0)}$ possède seulement des tuples de probabilité.

La "multiplication" d'un vecteur de probabilité π par les facteurs normaux du descripteur discret P est alors réalisée par les étapes suivantes :

[2]Dans le reste de la thèse, les *suites finies de matrices* seront appelées seulement *suites de matrices*. Car, seules les suites finies seront abordées.

1. Obtention du vecteur d'événements v à partir du vecteur de probabilité π (*i.e.*, le vecteur initial d'événements $v^{(0)}$), utilisant la procédure *ProbEv* (Définition 11.1.3) : $v^{(0)} = \text{ProbEv}(\pi)$;
2. Sachant que les éléments du vecteur d'événements v sont du même type que les éléments des matrices d'événements des facteurs normaux, le vecteur v est multiplié (par l'opérateur de concurrence "$*$" - Définition 9.3.3, page 170) de façon itérative par chaque facteur normal du descripteur discret ;
3. Après la multiplication du vecteur v par le dernier facteur normal (*i.e.*, après l'obtention de $v^{(N)}$ qui possède des chaînes de transitions globales), on obtient le vecteur de probabilité résultant π' à partir du vecteur d'événement $v^{(N)}$, utilisant la procédure *EvProb* (Définition 11.1.4) : $\pi' = \text{EvProb}(v^{(N)})$.

Dans FIG. 11.1, on présente le schéma de la "multiplication" du vecteur de probabilité π par les facteurs normaux afin d'illustrer les étapes décrites précédemment.

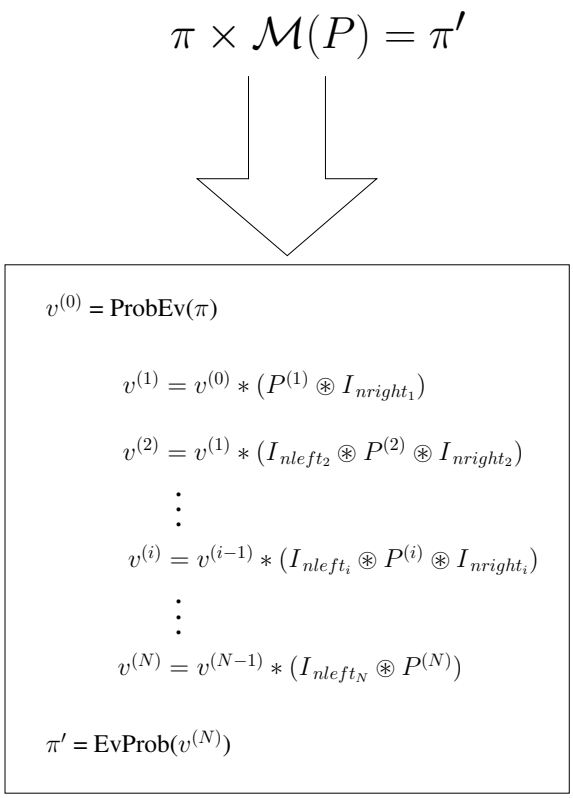


FIG. 11.1 – Schéma de la "multiplication" du vecteur de probabilité par les facteurs normaux

Les étapes 1 et 3 du schéma présenté précédemment sont facilement réalisables utilisant les procédures *ProbEv* et *EvProb* (Définitions 11.1.3 et 11.1.4). Pour réaliser la deuxième étape, il est nécessaire et suffisant de savoir multiplier un vecteur d'événements par les facteur normaux du descripteur discret. Le vecteur $v^{(0)}$ doit être multiplié par le premier facteur normal, le résultat (le vecteur $v^{(1)}$) est multiplié par le deuxième facteur normal et ainsi de suite jusqu'au dernier des facteurs normaux, ce qui donne le vecteur d'événements $v^{(N)}$. Cette multiplication est possible grâce à la propriété d'associativité du produit de concurrence de matrices. D'ailleurs, la propriété de commutativité entre facteurs normaux permet le produit de concurrence des facteurs normaux dans un ordre quelconque.

Cette méthode de "multiplication" d'un vecteur par les facteurs normaux présentée dans FIG. 11.1 a été inspirée de l'algorithme *Shuffle* utilisé pour les modèles SAN à temps continu [55]. L'idée de cette méthode est de réaliser, dès la multiplication du vecteur par le premier facteur normal, le *mélange parfait* (*shuffle*) des éléments du vecteur de façon à ce que le nombre de multiplications par les éléments des matrices du descripteur soit minimum.

Cette méthode est connue pour les modèles à temps continu, où les matrices du descripteur ne contiennent pas d'éléments fonctionnels. Cependant, dans la méthode présentée dans cette section, les éléments fonctionnels ne sont évalués qu'à la fin, *i.e.*, l'évaluation des éléments fonctionnels (probabilité d'occurrence et probabilité de routage fonctionnelles des événements) est réalisée après l'obtention des chaînes de transitions globales et leurs probabilités d'occurrence sont calculées comme présenté dans la définition 11.1.1. *Nous n'avons donc ici aucune restriction concernant la forme de ces relations fonctionnelles comme cela a pu être le cas dans le contexte des modèles à temps continu.*

Alors, même pour des modèles qui contiennent d'éléments fonctionnels, il est possible de réaliser la multiplication d'un vecteur par les facteurs normaux dans un ordre quelconque.

On remarque que $v^{(0)}$, $v^{(1)}$, ..., $v^{(N)}$ sont des notations pour les différentes valeurs du même vecteur v afin d'indiquer précisément les étapes de la multiplication du vecteur v par les facteurs normaux. Lorsqu'on multiplie $v^{(1)} = v^{(0)} * (P^{(1)} \circledast I_{nright_1})$, le vecteur v est multiplié par le premier facteur normal et le résultat de cette multiplication est stocké dans le *même* vecteur v.

Pour calculer la multiplication d'un vecteur v par le descripteur discret $P = \overset{N}{\underset{i=1}{\circledast}} P^{(i)}$ il est donc suffisant de savoir multiplier un vecteur par un facteur normal i, où $i \in [1..N]$:

$$v^{(i)} = v^{(i-1)} * \left(I_{nleft_i} \circledast P^{(i)} \circledast I_{nright_i} \right) \tag{11.5}$$

On présente d'abord la multiplication du vecteur v par le dernier facteur normal, *i.e.* :

$$v^{(N)} = v^{(N-1)} * \left(I_{nleft_N} \circledast P^{(N)} \right) \tag{11.6}$$

Pour réaliser la multiplication du vecteur v par le dernier facteur normal, il suffit de multiplier $nleft_N$ portions de taille n_N du vecteur par la matrice d'événements $P^{(N)}$, sachant que la matrice $I_{nleft_N} \circledast P^{(N)}$ est une matrice bloc diagonale (FIG. 11.2).

L'algorithme pour la multiplication du vecteur par le dernier facteur normal est présenté dans ALG. 11.1. Cet algorithme correspond à l'obtention des portions de taille n_N du vecteur $v^{(N-1)}$ (comme présenté dans FIG. 11.3) et leur multiplication par $P^{(N)}$.

Ensuite, on présente la multiplication du vecteur v par le premier facteur normal, *i.e.* :

$$v^{(1)} = v^{(0)} * \left(P^{(1)} \circledast I_{nright_1} \right) \tag{11.7}$$

Le premier facteur normal, grâce à la propriété de pseudo-commutativité (Propriété 9.3.2, page 174), est égal à :

$$P_\sigma \times (I_{nright_1} \circledast P^{(1)}) \times P_\sigma^T, \tag{11.8}$$

où σ est la permutation qui passe de $\{1, 2, ..., N\}$ à $\{2, ..., N, 1\}$.

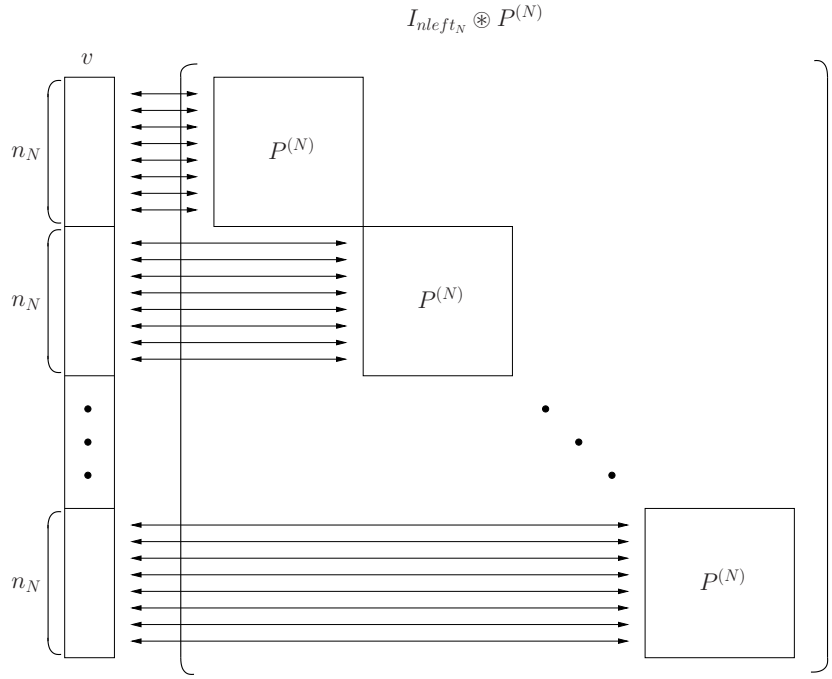


FIG. 11.2 – Multiplication du vecteur v par le dernier facteur normal

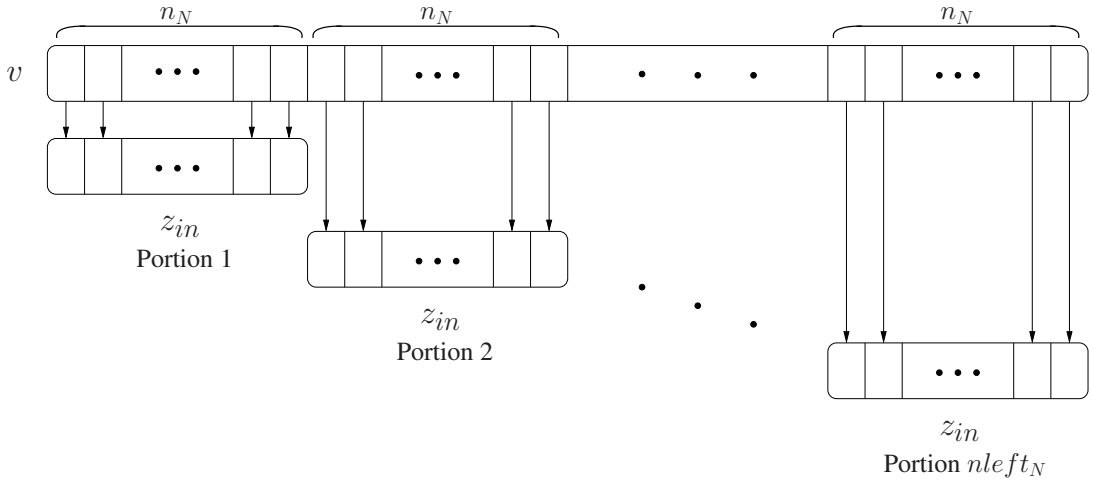


FIG. 11.3 – Obtention des portions du vecteur v pour la multiplication du dernier facteur normal

Alors, on peut calculer la multiplication du vecteur v par le premier facteur normal de façon analogue au cas précédent (*i.e.*, de la multiplication de v par le dernier facteur normal). Dans ce cas, la multiplication du vecteur v est réalisé en trois étapes :

1. la multiplication de v par la matrice de permutation P_σ ;
2. la multiplication du résultat de la première étape par le facteur normal commuté $I_{nright_1} \circledast P^{(1)}$;
3. la multiplication du résultat de la deuxième étape par la matrice de permutation ${P_\sigma}^T$.

ALG. 11.1 Multiplication du vecteur v par le dernier facteur normal

```
1:  base = 0 ;     /* base : indice du début de la portion dans le vecteur */
2:  /* Boucle pour les nleft_N portions du vecteur */
3:  for j = 1 to nleft_N do
4:     index = base + 1 ;     /* index : indice de l'élément courant du z_in dans le vecteur */
5:     /* Extraction du z_in */
6:     for k = 1 to n_N do
7:        z_in[k] = v[index] ;
8:        index = index + 1 ;
9:     end for
10:    multiply z_out = z_in * P^(N) ;     /* "*" est le produit de concurrence entre z_in et P^(N) */
11:    index = base + 1 ;     /* index : indice de l'élément courant du z_out dans le vecteur */
12:    /* Stockage du résultat z_out */
13:    for k = 1 to n_N do
14:       v[index] = z_out[k] ;
15:       index = index + 1 ;
16:    end for
17:    base = base + n_N ;     /* Saut dans le vecteur pour la portion suivante */
18: end for
```

La première étape correspond à une permutation du vecteur v. Cette permutation peut être exécutée lors de l'extraction de portions du vecteur v, *i.e.*, dans les remplissages des vecteurs z_{in} dans ALG. 11.1 (lignes 4-9). Pour le cas de la multiplication du dernier facteur normal (*i.e.*, le cas non-permuté), le vecteur z_{in} est rempli avec des portions successives de taille n_N.

En revanche, la permutation nécessaire pour le premier facteur normal équivaut à accéder au vecteur en prenant un élément à chaque intervalle de taille $nright_1$. On présente dans FIG. 11.4 la procédure de permutation, qu'il faut comparer à la procédure équivalente pour le cas sans permutation (décrit dans FIG. 11.3).

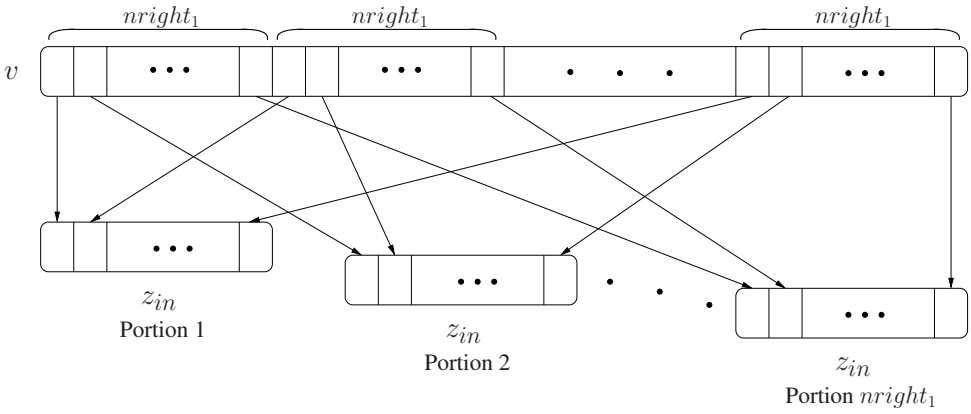


FIG. 11.4 – Permutations exécutés lors de la multiplication du premier facteur normal

La raison de cette permutation peut être compris en observant le format de la matrice $P^{(1)} \circledast I_{nright_1}$:

$$P^{(1)} \circledast I_{nright_1} = \begin{pmatrix} p^{(1)}_{1,1} * I_{nright_1} & p^{(1)}_{1,2} * I_{nright_1} & \cdots & p^{(1)}_{1,n_1} * I_{nright_1} \\ p^{(1)}_{2,1} * I_{nright_1} & p^{(1)}_{2,2} * I_{nright_1} & \cdots & p^{(1)}_{2,n_1} * I_{nright_1} \\ \vdots & \vdots & \ddots & \vdots \\ p^{(1)}_{n_1,1} * I_{nright_1} & p^{(1)}_{n_1,2} * I_{nright_1} & \cdots & p^{(1)}_{n_1,n_1} * I_{nright_1} \end{pmatrix}$$

La deuxième étape est identique au cas précédent, *i.e.*, la multiplication répétée des portions du vecteur v par la matrice $P^{(1)}$. La troisième étape est la permutation symétrique à la première et peut, donc, être exécutée lors du stockage des éléments du vecteur temporaire z_{out} (ALG. 11.1, lignes 11-16). L'algorithme pour cette multiplication (*i.e.*, pour la multiplication du vecteur par le premier facteur normal) est présenté dans ALG. 11.2.

ALG. 11.2 Multiplication du vecteur v par le premier facteur normal

1: $base = 0$; /* *base : indice du début de la portion dans le vecteur* */
2: /* *Boucle pour les* $nright_1$ *portions du vecteur* */
3: **for** $l = 1$ to $nright_1$ **do**
4: $index = base + l$; /* *index : indice de l'élément courant du* z_{in} *dans le vecteur* */
5: /* *Extraction du* z_{in} */
6: **for** $k = 1$ to n_1 **do**
7: $z_{in}[k] = v[index]$;
8: $index = index + nright_1$;
9: **end for**
10: **multiply** $z_{out} = z_{in} * P^{(1)}$; /* "$*$" *est le produit de concurrence entre* z_{in} *et* $P^{(1)}$ */
11: $index = base + l$; /* *index : indice de l'élément courant du* z_{out} *dans le vecteur* */
12: /* *Stockage du résultat* z_{out} */
13: **for** $k = 1$ to n_1 **do**
14: $v[index] = z_{out}[k]$;
15: $index = index + nright_1$;
16: **end for**
17: **end for**

Les autres facteurs normaux (*i.e.*, du deuxième facteur normal jusqu'à l'avant-dernier facteur normal) sont traités comme des combinaisons des deux cas précédents. La méthode de base consiste toujours à appliquer la propriété de pseudo-commutativité (Propriété 9.3.2, page 174) au facteur normal d'indice i, *i.e.* :

$$I_{nleft_i} \circledast P^{(i)} \circledast I_{nright_i} = P_\sigma \times \left(I_{nleft_i} \circledast I_{nright_i} \circledast P^{(i)} \right) {P_\sigma}^T,$$

où σ est la permutation qui passe de $\{1, \ldots, i-1, i, i+1, \ldots, N\}$ à $\{1, \ldots, i-1, i+1, \ldots, N, i\}$. Ceci amène à multiplier toujours des facteurs normaux de la forme :

$$I_{nleft_i} \circledast I_{nright_i} \circledast P^{(i)} = I_{\bar{n}_i} \circledast P^{(i)},$$

où $\bar{n}_i$ est le produit des dimensions de toutes les matrices sauf la i-ème matrice d'une suite, *i.e.*, $\bar{n}_i = nleft_i \times nright_i$.

De cette façon, les permutations sont faites selon la position de la matrice $P^{(i)}$. Dans ALG. 11.3, on présente l'algorithme pour réaliser la "multiplication" du vecteur de probabilité π par le descripteur discret $P = \overset{N}{\underset{i=1}{\circledast}} P^{(i)}$. Cet algorithme représente le schéma de la "multiplication" du vecteur par les facteurs normaux du descripteur présenté dans FIG. 11.1. On va appeler l'algorithme présenté dans ALG. 11.3 l'*algorithme de base*.

Dans ALG. 11.3, les facteurs normaux sont traités du premier ($i = 1$) jusqu'au dernier facteur normal ($i = N$) et π' est le vecteur de probabilité résultant de la "multiplication" du vecteur de probabilité π par les facteurs normaux du descripteur.

ALG. 11.3 *Algorithme de base* : "Multiplication" du vecteur de probabilité π par des facteurs normaux

```
1:  v = ProbEv(π) ;     /* Obtention du vecteur d'événements v à partir du vecteur de probabilité π */
2:  /* Boucle sur les facteurs normaux */
3:  for i = 1 to N do
4:     base = 0 ;     /* base : indice du début de la portion dans le vecteur */
5:     /* Boucle sur les portions */
6:     for j = 1 to nleft_i do
7:        /* Détail de la portion : boucle sur les z_in */
8:        for l = 1 to nright_i do
9:           index = base + l ;     /* index : indice de l'élément courant du z_in dans le vecteur */
10:          /* Extraction du z_in de la portion l (sauts de nright_i) */
11:          for k = 1 to n_i do
12:             z_in[k] = v[index] ;
13:             index = index + nright_i ;
14:          end for
15:          multiply z_out = z_in * P^(i) ;     /* "*" est le produit de concurrence entre z_in et P^(i) */
16:          index = base + l ;     /* index : indice de l'élément courant du z_out dans le vecteur */
17:          /* Stockage du résultat z_out (sauts de nright_i) */
18:          for k = 1 to n_i do
19:             v[index] = z_out[k] ;
20:             index = index + nright_i ;
21:          end for
22:       end for
23:       base = base + ( nright_i × n_i ) ;     /* Saut dans le vecteur pour la portion suivante */
24:    end for
25: end for
26: /* Stockage du résultat final dans π' */
27: π' = EvProb(v) ;     /* Obtention du vecteur de probabilité π' à partir du vecteur d'événements v */
```

Pour illustrer la "multiplication" d'un vecteur de probabilité par le descripteur discret d'un modèle SAN à temps discret, on prend par exemple le vecteur de probabilité π et le descripteur discret P suivants :

$$\pi = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \pi_1 & \pi_2 & \pi_3 & \pi_4 & \pi_5 & \pi_6 & \pi_7 & \pi_8 \\ \hline \end{array} \tag{11.9}$$

$$P = P^{(1)} \circledast P^{(2)} \circledast P^{(3)} = \begin{pmatrix} \bar{l}_1 & l_1 \\ s_1 & \bar{s}_1 \end{pmatrix} \circledast \begin{pmatrix} \bar{s}_1 & s_1 \\ l_2 & \bar{l}_2 \end{pmatrix} \circledast \begin{pmatrix} \bar{l}_3 & l_3 \\ s_1 & \bar{s}_1 \end{pmatrix} \tag{11.10}$$

où π_i est la probabilité dans la position i du vecteur de probabilité π, l_1, l_2 et l_3 sont événements locaux, s_1 est un événement synchronisant et l'ordre de priorité de ces événements est $\varrho_{s_1} < \varrho_{l_1} < \varrho_{l_2} < \varrho_{l_3}$.

A partir du vecteur de probabilité π de cet exemple - équation (11.9) - on peut obtenir le vecteur d'événements v utilisant la procédure *ProbEv* (Définition 11.1.3, page 205) afin de réaliser la multiplication de ce vecteur par les facteurs normaux du descripteur. Dans FIG. 11.5, on présente alors le vecteur d'événements v obtenu à partir du vecteur de probabilité π.

$$v = \text{ProbEv}(\pi)$$

	1	2	3	4	5	6	7	8
v =	$(I_{\breve{\mathcal{E}}}, \pi_1)$	$(I_{\breve{\mathcal{E}}}, \pi_2)$	$(I_{\breve{\mathcal{E}}}, \pi_3)$	$(I_{\breve{\mathcal{E}}}, \pi_4)$	$(I_{\breve{\mathcal{E}}}, \pi_5)$	$(I_{\breve{\mathcal{E}}}, \pi_6)$	$(I_{\breve{\mathcal{E}}}, \pi_7)$	$(I_{\breve{\mathcal{E}}}, \pi_8)$

FIG. 11.5 – Le vecteur d'événements v obtenu à partir du vecteur de probabilité π

Alors, la multiplication de $v * (P^{(1)} \circledast P^{(2)} \circledast P^{(3)})$ est égal à multiplication de v par les facteurs normaux $P^{(1)} \circledast I_{nright_1}$ (le premier facteur normal), $I_{nleft_2} \circledast P^{(2)} \circledast I_{nright_2}$ (le deuxième facteur normal) et $I_{nleft_3} \circledast P^{(3)}$ (le dernier facteur normal).

Pour éviter d'alourdir la représentation des éléments du vecteur v, on va représenter un tuple de transition locale, par exemple, $(l_1, \pi_{l_1}(x_1, y_1))$ par simplement (l_1, π_{l_1}). Dans cet exemple, toutes les probabilités de routage associées aux tuples de transition locale des événements sont égales à 1.

Sachant que les tuples de transition locale d'un événement complémentaire peuvent disparaître si cet événement n'atteint pas son degré d'idempotence (Équation (9.24), Propriété 9.2.1, page 168) lors de l'obtention d'un tuple de transition globale, on n'effectue pas les multiplications numériques de probabilité pour éviter de perdre l'information de la probabilité du tuple de probabilité. Toutefois, une fois qu'on a le tuple de transition globale d'un événement complémentaire, on applique la propriété 1 (Définition 11.1.2, page 204) sans risquer de perdre l'information de la probabilité du tuple de probabilité.

Pour le moment, on se restreint à appliquer la propriété 1 du tuple de probabilité *seulement* aux tuples de transition locale des *événements actifs*. Dans ces conditions, en multipliant le vecteur v présenté dans FIG. 11.5 par le premier facteur normal, on obtient le vecteur v résultant présenté dans FIG. 11.6.

Afin de faciliter la compréhension des figures suivantes dans cette section, on met en gras les valeurs des probabilités du vecteur π.

On peut observer dans FIG. 11.6 que la propriété 1 (Définition 11.1.2, page 204) des tuples de probabilité a été seulement appliquée aux tuples de transition locale des *événements actifs*. Par exemple, en observant la valeur de $v[1]$:

$$[(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_1}) * (\bar{l}_1, \pi_{\bar{l}_1})] + [(s_1, \boldsymbol{\pi_5} \times \pi_{s_1})], \tag{11.11}$$

la propriété 1 n'est pas appliquée aux tuples de probabilité $(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_1})$ et de transition locale $(\bar{l}_1, \pi_{\bar{l}_1})$, puisque $\bar{l}_1$ est un *événement complémentaire*. Par contre, sachant que s_1 est un *événement actif*, la propriété 1 est appliquée aux tuples de probabilité $(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_5})$ et de transition locale (s_1, π_{s_1}) ce qui donne $(s_1, \boldsymbol{\pi_5} \times \pi_{s_1})$.

En multipliant le vecteur v (de FIG. 11.6) par le deuxième facteur normal, on obtient le vecteur v présenté dans FIG. 11.7.

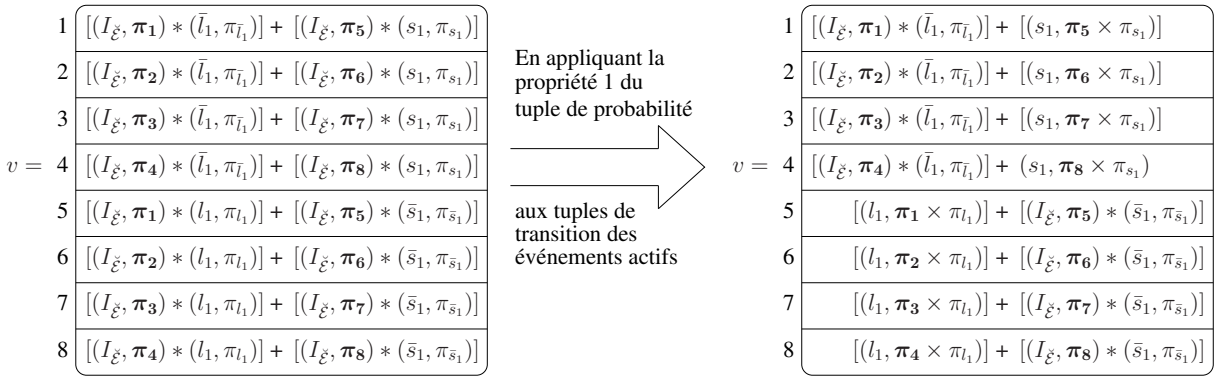


FIG. 11.6 – Le vecteur v après la multiplication par le premier facteur normal

$$
v = \begin{array}{|l|l|}
\hline
1 & [(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_1}) * (\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + [(s_1, \boldsymbol{\pi_5} \times \pi_{s_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + \\
 & [(\bar{l}_1, \pi_{\bar{l}_1}) * (l_2, \boldsymbol{\pi_3} \times \pi_{l_2})] + [(s_1, \boldsymbol{\pi_7} \times \pi_{s_1}) * (l_2, \pi_{l_2})] \\
\hline
2 & [(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_2}) * (\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + [(s_1, \boldsymbol{\pi_6} \times \pi_{s_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + \\
 & [(\bar{l}_1, \pi_{\bar{l}_1}) * (l_2, \boldsymbol{\pi_4} \times \pi_{l_2})] + [(s_1, \boldsymbol{\pi_8} \times \pi_{s_1}) * (l_2, \pi_{l_2})] \\
\hline
3 & [(\bar{l}_1, \pi_{\bar{l}_1}) * (s_1, \boldsymbol{\pi_1} \times \pi_{s_1})] + [(s_1, \boldsymbol{\pi_5} \times \pi_{s_1}) * (s_1, \pi_{s_1})] + \\
 & [(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_3}) * (\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{l}_2, \pi_{\bar{l}_2})] + [(s_1, \boldsymbol{\pi_7} \times \pi_{s_1}) * (\bar{l}_2, \pi_{\bar{l}_2})] \\
\hline
4 & [(\bar{l}_1, \pi_{\bar{l}_1}) * (s_1, \boldsymbol{\pi_2} \times \pi_{s_1})] + [(s_1, \boldsymbol{\pi_6} \times \pi_{s_1}) * (s_1, \pi_{s_1})] + \\
 & [(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_4}) * (\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{l}_2, \pi_{\bar{l}_2})] + [(s_1, \boldsymbol{\pi_8} \times \pi_{s_1}) * (\bar{l}_2, \pi_{\bar{l}_2})] \\
\hline
5 & [(l_1, \boldsymbol{\pi_1} \times \pi_{l_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + [(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_5}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + \\
 & [(l_1, \boldsymbol{\pi_3} \times \pi_{l_1}) * (l_2, \pi_{l_2})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (l_2, \boldsymbol{\pi_7} \times \pi_{l_2})] \\
\hline
6 & [(l_1, \boldsymbol{\pi_2} \times \pi_{l_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + [(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_6}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + \\
 & [(l_1, \boldsymbol{\pi_4} \times \pi_{l_1}) * (l_2, \pi_{l_2})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (l_2, \boldsymbol{\pi_8} \times \pi_{l_2})] \\
\hline
7 & [(l_1, \boldsymbol{\pi_1} \times \pi_{l_1}) * (s_1, \pi_{s_1})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (s_1, \boldsymbol{\pi_5} \times \pi_{s_1})] + \\
 & [(l_1, \boldsymbol{\pi_3} \times \pi_{l_1}) * (\bar{l}_2, \pi_{\bar{l}_2})] + [(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_7}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{l}_2, \pi_{\bar{l}_2})] \\
\hline
8 & [(l_1, \boldsymbol{\pi_2} \times \pi_{l_1}) * (s_1, \pi_{s_1})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (s_1, \boldsymbol{\pi_6} \times \pi_{s_1})] + \\
 & [(l_1, \boldsymbol{\pi_4} \times \pi_{l_1}) * (\bar{l}_2, \pi_{\bar{l}_2})] + [(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_8}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{l}_2, \pi_{\bar{l}_2})] \\
\hline
\end{array}
$$

FIG. 11.7 – Le vecteur v après la multiplication par le deuxième facteur normal

Et finalement, en multipliant le vecteur v (de FIG. 11.7) par le dernier facteur normal, on obtient le vecteur v présenté dans FIG. 11.8.

Il est intéressant de remarquer qu'après la multiplication du vecteur v par un facteur normal d'indice i, un élément de ce vecteur possède $nleft_i$ facteurs, car les éléments sont des éléments complexes (*i.e.*, des produits de concurrence des chaînes de transitions locales). Dans ce cas, le calcul de la probabilité d'occurrence de la chaîne de transitions globales d'un élément de v est seulement faite après la multiplication de v par le dernier facteur normal.

Alors, après la multiplication de v par le dernier facteur normal, on applique la propriété d'idempotence des événements (Propriété 9.2.1, page 168) pour les produits de concurrence des chaînes de transitions locales afin d'obtenir les chaînes de transitions globales et, par conséquence, on calcule la probabilité d'occurrence de ces chaînes de transitions globales. Par exemple, en prenant la valeur de $v[1]$ (de FIG. 11.8) qui possède la somme de 8 produits de concurrence des éléments des matrices du descripteur, on peut obtenir les chaînes de transitions globales suivantes :

$$
v = \begin{array}{c|l}
1 & \begin{array}{l}
[(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_1}) * (\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{l}_3, \pi_{\bar{l}_3})] + [(s_1, \boldsymbol{\pi_5} \times \pi_{s_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{l}_3, \pi_{\bar{l}_3})] + \\
[(\bar{l}_1, \pi_{\bar{l}_1}) * (l_2, \boldsymbol{\pi_3} \times \pi_{l_2}) * (\bar{l}_3, \pi_{\bar{l}_3})] + [(s_1, \boldsymbol{\pi_7} \times \pi_{s_1}) * (l_2, \pi_{l_2}) * (\bar{l}_3, \pi_{\bar{l}_3})] + \\
[(\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (s_1, \boldsymbol{\pi_2} \times \pi_{s_1})] + [(s_1, \boldsymbol{\pi_6} \times \pi_{s_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (s_1, \pi_{s_1})] + \\
[(\bar{l}_1, \pi_{\bar{l}_1}) * (l_2, \boldsymbol{\pi_4} \times \pi_{l_2}) * (s_1, \pi_{s_1})] + [(s_1, \boldsymbol{\pi_8} \times \pi_{s_1}) * (l_2, \pi_{l_2}) * (s_1, \pi_{s_1})]
\end{array} \\
\hline
2 & \begin{array}{l}
[(\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (l_3, \boldsymbol{\pi_1} \times \pi_{l_3})] + [(s_1, \boldsymbol{\pi_5} \times \pi_{s_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (l_3, \pi_{l_3})] + \\
[(\bar{l}_1, \pi_{\bar{l}_1}) * (l_2, \boldsymbol{\pi_3} \times \pi_{l_2}) * (l_3, \pi_{l_3})] + [(s_1, \boldsymbol{\pi_7} \times \pi_{s_1}) * (l_2, \pi_{l_2}) * (l_3, \pi_{l_3})] + \\
[(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_2}) * (\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + [(s_1, \boldsymbol{\pi_6} \times \pi_{s_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + \\
[(\bar{l}_1, \pi_{\bar{l}_1}) * (l_2, \boldsymbol{\pi_4} \times \pi_{l_2}) * (\bar{s}_1, \pi_{\bar{s}_1})] + [(s_1, \boldsymbol{\pi_8} \times \pi_{s_1}) * (l_2, \pi_{l_2}) * (\bar{s}_1, \pi_{\bar{s}_1})]
\end{array} \\
\hline
3 & \begin{array}{l}
[(\bar{l}_1, \pi_{\bar{l}_1}) * (s_1, \boldsymbol{\pi_1} \times \pi_{s_1}) * (\bar{l}_3, \pi_{\bar{l}_3})] + [(s_1, \boldsymbol{\pi_5} \times \pi_{s_1}) * (s_1, \pi_{s_1}) * (\bar{l}_3, \pi_{\bar{l}_3})] + \\
[(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_3}) * (\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (\bar{l}_3, \pi_{\bar{l}_3})] + [(s_1, \boldsymbol{\pi_7} \times \pi_{s_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (\bar{l}_3, \pi_{\bar{l}_3})] + \\
[(\bar{l}_1, \pi_{\bar{l}_1}) * (s_1, \boldsymbol{\pi_2} \times \pi_{s_1}) * (s_1, \pi_{s_1})] + [(s_1, \boldsymbol{\pi_6} \times \pi_{s_1}) * (s_1, \pi_{s_1}) * (s_1, \pi_{s_1})] + \\
[(\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (s_1, \boldsymbol{\pi_4} \times \pi_{s_1})] + [(s_1, \boldsymbol{\pi_8} \times \pi_{s_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (s_1, \pi_{s_1})]
\end{array} \\
\hline
4 & \begin{array}{l}
[(\bar{l}_1, \pi_{\bar{l}_1}) * (s_1, \boldsymbol{\pi_1} \times \pi_{s_1}) * (l_3, \pi_{l_3})] + [(s_1, \boldsymbol{\pi_5} \times \pi_{s_1}) * (s_1, \pi_{s_1}) * (l_3, \pi_{l_3})] + \\
[(\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (l_3, \boldsymbol{\pi_3} \times \pi_{l_3})] + [(s_1, \boldsymbol{\pi_7} \times \pi_{s_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (l_3, \pi_{l_3})] + \\
[(\bar{l}_1, \pi_{\bar{l}_1}) * (s_1, \boldsymbol{\pi_2} \times \pi_{s_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + [(s_1, \boldsymbol{\pi_6} \times \pi_{s_1}) * (s_1, \pi_{s_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + \\
[(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_4}) * (\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (\bar{s}_1, \pi_{\bar{s}_1})] + [(s_1, \boldsymbol{\pi_8} \times \pi_{s_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (\bar{s}_1, \pi_{\bar{s}_1})]
\end{array} \\
\hline
5 & \begin{array}{l}
[(l_1, \boldsymbol{\pi_1} \times \pi_{l_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{l}_3, \pi_{\bar{l}_3})] + [(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_5}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{l}_3, \pi_{\bar{l}_3})] + \\
[(l_1, \boldsymbol{\pi_3} \times \pi_{l_1}) * (l_2, \pi_{l_2}) * (\bar{l}_3, \pi_{\bar{l}_3})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (l_2, \boldsymbol{\pi_7} \times \pi_{l_2}) * (\bar{l}_3, \pi_{\bar{l}_3})] + \\
[(l_1, \boldsymbol{\pi_2} \times \pi_{l_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (s_1, \pi_{s_1})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (s_1, \boldsymbol{\pi_6} \times \pi_{s_1})] + \\
[(l_1, \boldsymbol{\pi_4} \times \pi_{l_1}) * (l_2, \pi_{l_2}) * (s_1, \pi_{s_1})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (l_2, \boldsymbol{\pi_8} \times \pi_{l_2}) * (s_1, \pi_{s_1})]
\end{array} \\
\hline
6 & \begin{array}{l}
[(l_1, \boldsymbol{\pi_1} \times \pi_{l_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (l_3, \pi_{l_3})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (l_3, \boldsymbol{\pi_5} \times \pi_{l_3})] + \\
[(l_1, \boldsymbol{\pi_3} \times \pi_{l_1}) * (l_2, \pi_{l_2}) * (l_3, \pi_{l_3})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (l_2, \boldsymbol{\pi_7} \times \pi_{l_2}) * (l_3, \pi_{l_3})] + \\
[(l_1, \boldsymbol{\pi_2} \times \pi_{l_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + [(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_6}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + \\
[(l_1, \boldsymbol{\pi_4} \times \pi_{l_1}) * (l_2, \pi_{l_2}) * (\bar{s}_1, \pi_{\bar{s}_1})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (l_2, \boldsymbol{\pi_8} \times \pi_{l_2}) * (\bar{s}_1, \pi_{\bar{s}_1})]
\end{array} \\
\hline
7 & \begin{array}{l}
[(l_1, \boldsymbol{\pi_1} \times \pi_{l_1}) * (s_1, \pi_{s_1}) * (\bar{l}_3, \pi_{\bar{l}_3})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (s_1, \boldsymbol{\pi_5} \times \pi_{s_1}) * (\bar{l}_3, \pi_{\bar{l}_3})] + \\
[(l_1, \boldsymbol{\pi_3} \times \pi_{l_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (\bar{l}_3, \pi_{\bar{l}_3})] + [(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_7}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (\bar{l}_3, \pi_{\bar{l}_3})] + \\
[(l_1, \boldsymbol{\pi_2} \times \pi_{l_1}) * (s_1, \pi_{s_1}) * (s_1, \pi_{s_1})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (s_1, \boldsymbol{\pi_6} \times \pi_{s_1}) * (s_1, \pi_{s_1})] + \\
[(l_1, \boldsymbol{\pi_4} \times \pi_{l_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (s_1, \pi_{s_1})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (s_1, \boldsymbol{\pi_8} \times \pi_{s_1})]
\end{array} \\
\hline
8 & \begin{array}{l}
[(l_1, \boldsymbol{\pi_1} \times \pi_{l_1}) * (s_1, \pi_{s_1}) * (l_3, \pi_{l_3})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (s_1, \boldsymbol{\pi_5} \times \pi_{s_1}) * (l_3, \pi_{l_3})] + \\
[(l_1, \boldsymbol{\pi_3} \times \pi_{l_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (l_3, \pi_{l_3})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (l_3, \boldsymbol{\pi_7} \times \pi_{l_3})] + \\
[(l_1, \boldsymbol{\pi_2} \times \pi_{l_1}) * (s_1, \pi_{s_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + [(\bar{s}_1, \pi_{\bar{s}_1}) * (s_1, \boldsymbol{\pi_6} \times \pi_{s_1}) * (\bar{s}_1, \pi_{\bar{s}_1})] + \\
[(l_1, \boldsymbol{\pi_4} \times \pi_{l_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (\bar{s}_1, \pi_{\bar{s}_1})] + [(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_8}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{l}_2, \pi_{\bar{l}_2}) * (\bar{s}_1, \pi_{\bar{s}_1})]
\end{array}
\end{array}
$$

FIG. 11.8 – Le vecteur v après la multiplication par le dernier facteur normal

1. $(I_{\breve{\mathcal{E}}}, \boldsymbol{\pi_1}) * (\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{l}_3, \pi_{\bar{l}_3}) \Rightarrow (\bar{l}_1, \boldsymbol{\pi_1} \times \Pi_{\bar{l}_1}) \cdot (\bar{l}_3, \Pi_{\bar{l}_3})$: L'événement complémentaire $\bar{s}_1$ ne respecte pas son degré d'idempotence dans le produit de concurrence et, par la propriété 9.2.1 (Équation (9.24), page 168), son tuple de transition sera égal à $I_{\breve{\mathcal{I}}}$ et par conséquence il va disparaître de la chaîne de transitions globales. On applique la propriété 1 (Définition 11.1.2, page 204) du tuple de probabilité à l'événement complémentaire $\bar{l}_1$, une fois qu'on a le tuple de transition globale de cet événement.

2. $(s_1, \boldsymbol{\pi_5} \times \pi_{s_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (\bar{l}_3, \pi_{\bar{l}_3}) \Rightarrow O_{\breve{\mathcal{I}}}$: L'événement s_1 ne respecte pas son degré d'idempotence dans le produit de concurrence et, par la propriété 9.2.1 (Équation (9.25), page 168), ce produit de concurrence des chaînes de transitions locales est évalué à $O_{\breve{\mathcal{I}}}$.

3. $(\bar{l}_1, \pi_{\bar{l}_1}) * (l_2, \boldsymbol{\pi_3} \times \pi_{l_2}) * (\bar{l}_3, \pi_{\bar{l}_3}) \Rightarrow (\bar{l}_1, \Pi_{\bar{l}_1}) \cdot (l_2, \boldsymbol{\pi_3} \times \Pi_{l_2}) \cdot (\bar{l}_3, \Pi_{\bar{l}_3})$: Tous les événements locaux

de ce produit de concurrence respectent leurs degré d'idempotence. On conservent les tuples de transition globale de ces événements qui forment la chaîne de transitions globales.

4. $(s_1, \boldsymbol{\pi_7} \times \pi_{s_1}) * (l_2, \pi_{l_2}) * (\bar{l}_3, \pi_{\bar{l}_3}) \Rightarrow O_{\check{\mathcal{I}}}$: Ce produit de concurrence, ainsi que les produits de concurrence des items 5, 6, 7 et 8 sont évalués à $O_{\check{\mathcal{I}}}$ pour la même raison que le produit de concurrence de l'item 2 est aussi évalué à $O_{\check{\mathcal{I}}}$ (*i.e.*, l'événement s_1 ne respecte pas son degré d'idempotence).
5. $(\bar{l}_1, \pi_{\bar{l}_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (s_1, \boldsymbol{\pi_2} \times \pi_{s_1}) \Rightarrow O_{\check{\mathcal{I}}}$
6. $(s_1, \boldsymbol{\pi_6} \times \pi_{s_1}) * (\bar{s}_1, \pi_{\bar{s}_1}) * (s_1, \pi_{s_1}) \Rightarrow O_{\check{\mathcal{I}}}$
7. $(\bar{l}_1, \pi_{\bar{l}_1}) * (l_2, \boldsymbol{\pi_4} \times \pi_{l_2}) * (s_1, \pi_{s_1}) \Rightarrow O_{\check{\mathcal{I}}}$
8. $(s_1, \boldsymbol{\pi_8} \times \pi_{s_1}) * (l_2, \pi_{l_2}) * (s_1, \pi_{s_1}) \Rightarrow O_{\check{\mathcal{I}}}$

En prenant la première chaîne de transitions globales $(\bar{l}_1, \boldsymbol{\pi_1} \times \Pi_{\bar{l}_1}) \cdot (\bar{l}_3, \Pi_{\bar{l}_3})$ de $v[1]$, on calcule la probabilité d'occurrence[3] (Définition 11.1.1) de cette chaîne de la façon suivante :

$$\boldsymbol{\pi_1} \times \rho_{\bar{l}_1} \times \rho_{\bar{l}_3}$$

Alors, en obtenant les chaînes de transitions globales[4] pour tous les éléments de v (de FIG. 11.8), on peut calculer le vecteur de probabilité π' résultant à partir de EvProb(v). Dans FIG. 11.9, on présente le vecteur de probabilité π' résultant de la multiplication du vecteur de probabilité π par le descripteur discret P (présentés en page 213).

$$\pi' = \text{EvProb}(v)$$

$\pi' =$

1	$(\boldsymbol{\pi_1} \times \rho_{\bar{l}_1} \times \rho_{\bar{l}_3}) + (\boldsymbol{\pi_3} \times \rho_{\bar{l}_1} \times \rho_{l_2} \times \rho_{\bar{l}_3})$
2	$(\boldsymbol{\pi_1} \times \rho_{\bar{l}_1} \times \rho_{l_3}) + (\boldsymbol{\pi_2} \times \rho_{\bar{l}_1}) + (\boldsymbol{\pi_3} \times \rho_{\bar{l}_1} \times \rho_{l_2} \times \rho_{l_3}) + (\boldsymbol{\pi_4} \times \rho_{\bar{l}_1} \times \rho_{l_2})$
3	$(\boldsymbol{\pi_3} \times \rho_{\bar{l}_1} \times \rho_{\bar{l}_2} \times \rho_{\bar{l}_3}) + (\boldsymbol{\pi_6} \times \rho_{s_1})$
4	$(\boldsymbol{\pi_3} \times \rho_{\bar{l}_1} \times \rho_{\bar{l}_2} \times \rho_{l_3}) + (\boldsymbol{\pi_4} \times \rho_{\bar{l}_1} \times \rho_{\bar{l}_2})$
5	$(\boldsymbol{\pi_1} \times \rho_{l_1} \times \rho_{\bar{l}_3}) + (\boldsymbol{\pi_3} \times \rho_{l_1} \times \rho_{l_2} \times \rho_{\bar{l}_3}) + (\boldsymbol{\pi_5} \times \rho_{\bar{l}_3}) + (\boldsymbol{\pi_7} \times \rho_{l_2} \times \rho_{\bar{l}_3})$
6	$(\boldsymbol{\pi_1} \times \rho_{l_1} \times \rho_{l_3}) + (\boldsymbol{\pi_2} \times \rho_{l_1}) + (\boldsymbol{\pi_3} \times \rho_{l_1} \times \rho_{l_2} \times \rho_{l_3}) + (\boldsymbol{\pi_4} \times \rho_{l_1} \times \rho_{l_2}) +$ $(\boldsymbol{\pi_5} \times \rho_{l_3}) + (\boldsymbol{\pi_6} \times \rho_{\bar{s}_1}) + (\boldsymbol{\pi_7} \times \rho_{l_2} \times \rho_{l_3}) + (\boldsymbol{\pi_8} \times \rho_{l_2})$
7	$(\boldsymbol{\pi_3} \times \rho_{l_1} \times \rho_{\bar{l}_2} \times \rho_{\bar{l}_3}) + (\boldsymbol{\pi_7} \times \rho_{\bar{l}_2} \times \rho_{\bar{l}_3})$
8	$(\boldsymbol{\pi_3} \times \rho_{l_1} \times \rho_{\bar{l}_2} \times \rho_{l_3}) + (\boldsymbol{\pi_4} \times \rho_{l_1} \times \rho_{\bar{l}_2}) + (\boldsymbol{\pi_7} \times \rho_{\bar{l}_2} \times \rho_{l_3}) + (\boldsymbol{\pi_8} \times \rho_{\bar{l}_2})$

FIG. 11.9 – Le vecteur de probabilité π' obtenu à partir du vecteur d'événements v

[3] On rappelle que par cet exemple toutes les probabilités de routage associées aux tuples de transition locale des événements sont égales à 1 ce qui donne une probabilité de routage globale égale à 1 associées aux tuples de transition globale.

[4] On s'intéresse *uniquement* aux chaînes de transitions globales qui sont différentes de $O_{\check{\mathcal{I}}}$, car une chaîne de transitions globales évaluée à $O_{\check{\mathcal{I}}}$ possède une probabilité d'occurrence égale à 0 (zéro).

Dans la section suivante, on présente la complexité de l'algorithme (présenté dans ALG. 11.3) de la "multiplication" du vecteur de probabilité π par les facteurs normaux du descripteur discret P d'un modèle SAN à temps discret.

11.2.1 Complexité

La complexité de la "multiplication" du vecteur par le descripteur discret $P = \circledast_{i=1}^{N} P^{(i)}$ est obtenue en comptant le nombre de produits de concurrence du produit vecteur-matrice (*i.e.*, le nombre de produits de concurrence du *vecteur* z_{in} par le *matrice* $P^{(i)}$ du facteur normal i) exécuté dans ALG. 11.3 (ligne 15).

En supposant que les matrices $P^{(i)}$ sont pleines, le nombre de produits de concurrence pour chaque produit vecteur-matrice est égal à $(n_i)^2$. On rappelle que les vecteurs v, z_{in} et z_{out} sont des vecteurs d'événements avec des expressions non réduites de l'algèbre de tensorielle complexe (*i.e.*, des produits de concurrence des chaînes de transitions locales). Dans ce cas, à chaque produit de concurrence de z_{in} par un facteur normal d'indice i (ligne 15, ALG. 11.3), chaque élément de z_{in} possède $nleft_i$ facteurs, d'où $nleft_i$ est le coût du produit d'un élément du vecteur par le facteur normal i. Encore, à chaque boucle en i de ALG. 11.3, $nleft_i \times nright_i$ (*i.e.*, $\bar{n}_i$) produits vecteur-matrice sont exécutés avec des matrices de taille n_i. La complexité de ALG. 11.3 est[5] :

$$\sum_{i=1}^{N} \left(\bar{n}_i \times (n_i)^2 \times nleft_i\right) = \prod_{i=1}^{N} n_i \times \sum_{i=1}^{N} \left(n_i \times nleft_i\right) \tag{11.12}$$

Si les matrices $P^{(i)}$ sont stockées dans un format creux, le nombre de produits de concurrence pour chaque produit vecteur-matrice est, en général, inférieur à $(n_i)^2$. Dans ce cas, si nz_i est le nombre d'éléments non-nuls de la matrice $P^{(i)}$, la complexité de ALG. 11.3 est :

$$\sum_{i=1}^{N} \left(\bar{n}_i \times nz_i \times nze_i\right) = \prod_{i=1}^{N} n_i \times \sum_{i=1}^{N} \left(\frac{nz_i}{n_i} \times nze_i\right), \tag{11.13}$$

où $nze_i = \prod_{k=1}^{i-1} \frac{nz_k}{n_k}$, et $nze_1 = 1$.

Dans la section suivante, on présente quelques optimisations afin de réduire le nombre de produits de concurrence (la complexité) de l'algorithme présenté dans ALG. 11.3.

11.2.2 Optimisations

Comme on l'a mentionné précédemment, l'idée de la méthode présentée dans ce chapitre est de réaliser un mélange parfait des éléments du vecteur, partant de la multiplication du vecteur par le premier facteur normal, de façon à ce que le nombre de multiplications entre ces éléments par les matrices du

[5] Rappelons que par la définition du formalisme des Réseaux d'Automates Stochastiques, ainsi que pour la définition des suites de matrices, $\bar{n}_i \times n_i = nleft_i \times nright_i \times n_i = \prod_{i=1}^{N} n_i$.

descripteur soit minimum. Les optimisations qu'on va présenter dans cette section visent à profiter de la sémantique du formalisme SAN à temps discret afin de réduire le nombre de produits de concurrence du produit vecteur-matrice exécuté dans ALG. 11.3 (ligne 15).

Optimisation Locale

La première optimisation est liée à l'utilisation du *degré d'idempotence* associé aux *événements locaux* du modèle.

Pour que le nombre de produits de concurrence du produit vecteur-matrice soit toujours minimum, il faut qu'on applique depuis les premières multiplications des éléments du vecteur par les éléments des matrices la propriété 1 (Définition 11.1.2, page 204). Dans un premier temps, on restreint l'application de cette propriété aux tuples de transition locale des *événements actifs*. Cette restriction évite la perte de la valeur de la probabilité du tuple de probabilité, sachant que le tuple de probabilité d'un *événement complémentaire* peut disparaître si cet événement n'atteint pas son degré d'idempotence (Équation (9.24), Propriété 9.2.1, page 168).

Toutefois, les *événements locaux*, par la définition du formalisme SAN (Définition 2.2.12, page 28), respectent toujours leurs degré d'idempotence, *i.e.*, le degré d'idempotence d'un événement local est égal à 1 et une fois qu'un événement local apparaît dans un produit de concurrence, il respecte déjà son degré d'idempotence. Alors, on peut appliquer la propriété 1 entre le tuple de probabilité et les tuples de transition locale des événements locaux (indépendamment du fait que l'événement soit du type *actif* ou *complémentaire*) et simplifier l'expression.

Si on reprend l'exemple de la multiplication du vecteur v (de FIG. 11.5) par le premier facteur normal du descripteur discret P (présenté en page 213), alors en appliquant l'*optimisation locale* on obtient le vecteur v présenté dans FIG. 11.10.

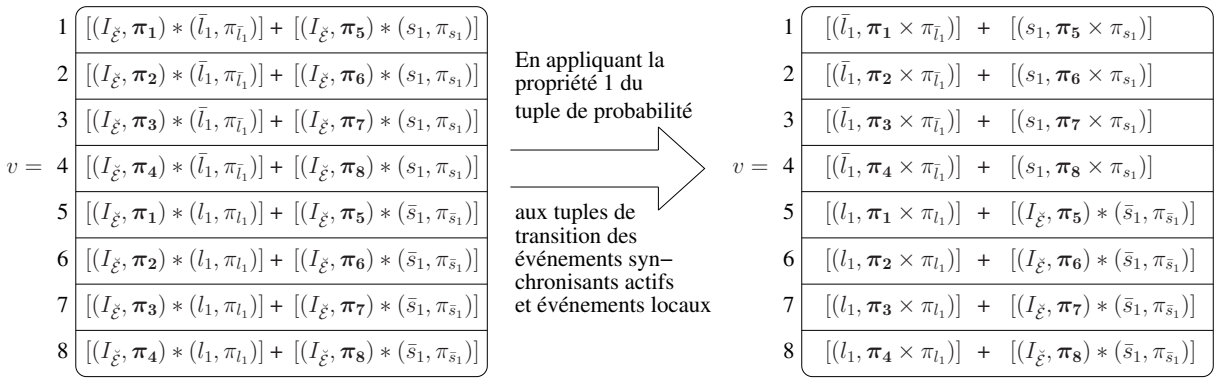


FIG. 11.10 – Le vecteur v optimisé après la multiplication par le premier facteur normal

Dans 11.10, on peut remarquer l'application de la propriété 1 (Définition 11.1.2, page 204) du tuple de probabilité aux tuples de transition locale $(\bar{l}_1, \pi_{\bar{l}_1})$ de l'événement complémentaire $\bar{l}_1$ dans $v[1]$, $v[2]$, $v[3]$ et $v[4]$. L'application de cette propriété, dès la multiplication du premier facteur normal, aux tuples de transition locale va se propager pour la multiplication du vecteur par les facteurs normaux suivants.

Optimisation Positionnelle

La deuxième optimisation repose sur l'*aspect positionnel* des *événements synchronisants* dans les automates du modèle.

Dans le contexte des SAN, un événement est réalisable uniquement s'il est réalisable au même instant dans *tous les automates concernés* par cet événement. On rappelle que l'ensemble d'indices d'automates concernés par un événement e est noté par $\mathcal{O}_e$ et $|\mathcal{O}_e| = \eta_e$ (le degré d'idempotence de l'événement e). En utilisant cet aspect positionnel des événements synchronisants dans un produit de concurrence de chaînes de transitions locales, on peut obtenir une optimisation appliquant le corollaire 9.2.1 (page 169). On rappelle ce corollaire :

Soit un événement $e \in \mathcal{E}$ et une combinaison concurrente de couples $ccc = (ccc_A) * (ccc_B)$, où ccc_A et ccc_B sont des combinaisons concurrentes de couples tel que $ccc_A = csc_1 * \ldots * csc_p$ et $ccc_B = csc_{p+1} * \ldots * csc_N$,

$$\textit{s'il existe } i \in \mathcal{O}_e, \textit{ où } i \leq p, \textit{ tel que } e \notin \breve{\mathcal{E}}_{csc_i}, \textit{ alors}$$

l'évaluation de $ccc_A = O_{\breve{\mathcal{T}}}$ et par le produit de concurrence avec le tuple de transition nul (propriété 2 de la définition 9.2.9, page 167) ce qui donne $ccc = O_{\breve{\mathcal{T}}}$.

Alors, dès la multiplication des éléments du vecteur par le deuxième facteur normal, on peut appliquer le corollaire 9.2.1 aux événements synchronisants actifs afin d'éviter le produit de concurrence (obtenu après la multiplication du deuxième facteur normal) par les facteurs normaux suivants, car le produit de concurrence obtenu après la multiplication du dernier facteur normal sera forcément évalué à $O_{\breve{\mathcal{T}}}$ lors de l'obtention de la chaîne de transitions globales.

Pour illustrer cet optimisation, on reprend l'exemple de la multiplication du vecteur v par le deuxième facteur normal du descripteur discret P (présenté en page 213). Par exemple, on prend le premier produit de concurrence de $v[3]$ (FIG. 11.7) après la multiplication par le deuxième facteur normal, *i.e.* :

$$[\underbrace{(\bar{l}_1, \pi_{\bar{l}_1})}_{csc_1} * \underbrace{(s_1, \boldsymbol{\pi_1} \times \pi_{s_1})}_{cscs_2}] \tag{11.14}$$

Dans ce modèle, l'ensemble d'indices d'automates concernés par l'événement s_1 est égal à $\mathcal{O}_{s_1} = \{1, 2, 3\}$. Dans ce cas, en observant le produit de concurrence présenté dans l'équation (11.14), on constate qu'il existe un $i \in \mathcal{O}_{s_1}$ (*i.e.*, $i = 1$) tel que $s_1 \notin \breve{\mathcal{E}}_{csc_i}$. Alors, même si l'on continue à multiplier ce produit de concurrence par les éléments des facteurs normaux, cette chaîne sera finalement évaluée à $O_{\breve{\mathcal{T}}}$. Dans ce cas, une fois que cette situation est constatée, le produit de concurrence $[(\bar{l}_1, \pi_{\bar{l}_1}) * (s_1, \boldsymbol{\pi_1} \times \pi_{s_1})]$ par les éléments du dernier facteur normal, *i.e.*, $(\bar{l}_3, \pi_{\bar{l}_3})$ et $(\bar{l}_3, \pi_{\bar{l}_3})$, n'est pas réalisée.

Cette optimisation peut, en fonction des caractéristiques du modèle, diminuer considérablement le nombre de produits de concurrence (complexité) de l'algorithme présenté dans ALG. 11.3.

Grâce à la *commutativité* du *produit de concurrence*, il est possible d'imaginer un *re-ordonnancement* des automates du modèle de façon à ce que les automates concernés par un événement synchronisant soient *proches* (*i.e.*, que les facteurs normaux relatifs à ces automates se trouvent dans un ordre tel que l'un soit après l'autre).

D'ailleurs, outre l'optimisation du nombre de produits de concurrence, cette optimisation réduit la quantité de mémoire nécessaire pour stocker les éléments (produits de concurrence des chaînes de transitions locales) du vecteur v. Il faut toujours garder à l'esprit que les éléments du vecteur v sont des expressions non réduites de l'algèbre tensorielle complexe (*i.e.*, des produits de concurrence de chaînes de transitions locales).

Prenons les valeurs $v[1]$ et $v[2]$ du vecteur v après la multiplication par le deuxième facteur normal de l'exemple présenté précédemment. Dans ce cas, on peut observer dans FIG. 11.11 qu'au lieu de 4 produits de concurrence, on effectue seulement 2 produits de concurrence avec cette optimisation.

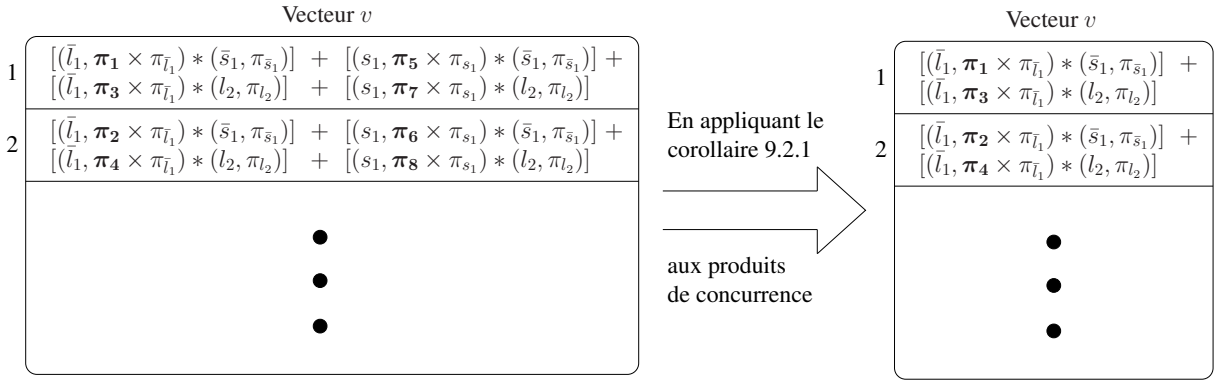


FIG. 11.11 – Le vecteur v optimisé après la multiplication par le deuxième facteur normal

De plus, cette optimisation se propage pour la multiplication du vecteur par les facteurs normaux suivants. Dans ce cas, après la multiplication du vecteur v par le dernier facteur normal, $v[1]$ et $v[2]$ auraient 4 produits de concurrence au lieu de 8 (FIG. 11.12).

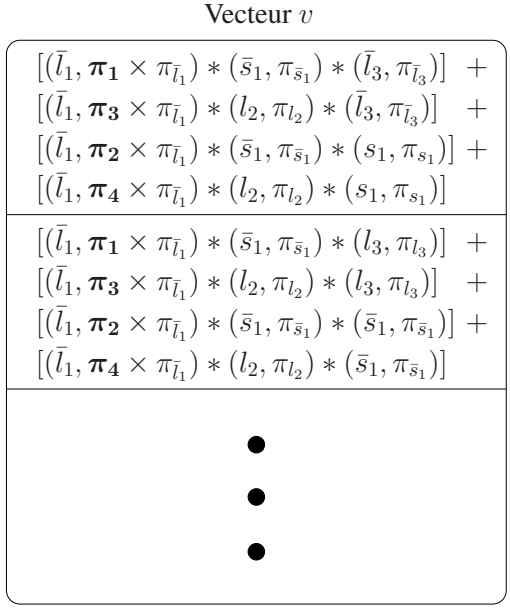


FIG. 11.12 – Le vecteur v optimisé après la multiplication par le dernier facteur normal

11.3 Expériences numériques

On va présenter dans cette section quelques expériences numériques qui doivent être faites afin de tester la performance de la méthode et des optimisations suggérées dans la section précédente. En raison de manque temps, la méthode et les optimisations présentées n'ont pas pu être développées dans un module du logiciel PEPS (*Performance Evaluation of Parallel Systems*) [17] au cours de cette thèse pour l'obtention des résultats numériques.

L'implantation de ce module sera faite comme un prolongement immédiat afin d'obtenir des mesures numériques pour la méthodes et les optimisations présentées dans ce chapitre. On envisage de réaliser des mesures numériques pour les expériences suivantes :

(1) l'*Algorithme de base* (ALG. 11.3) ;

(2) l'*Optimisation Locale* ;

(3) l'*Optimisation Positionnelle* ;

(4) l'*Optimisation Locale* ***et*** *Positionnelle*.

Les mesures numériques réalisées pour ces expériences doivent impérativement prendre en compte deux aspects :

– le **temps d'exécution** pour réaliser la *multiplication du vecteur de probabilité par le descripteur discret* (*i.e.*, le temps nécessaire pour réaliser les trois étapes de la multiplication décrites en page 208) ;

– la **quantité finale de mémoire** utilisée pour *stocker les éléments* du vecteur v (*i.e.*, la mémoire utilisée pour représenter les expressions non réduites de l'algèbre tensorielle complexe).

En réalisant l'expérience (1), *i.e.*, l'*algorithme de base*, on s'intéresse à la performance de la méthode de multiplication vecteur-descripteur proposée dans ce chapitre. L'expérience (2), l'*optimisation locale*, essaie de réduire le nombre de produits de concurrence du produit vecteur-matrice (*i.e.*, le nombre de produits de concurrence du *vecteur* z_{in} par le *matrice* $P^{(i)}$ du facteur normal i) exécutées dans ALG. 11.3 (ligne 15). L'expérience (3), l'*optimisation positionnelle*, tente de réduire le nombre d'éléments (produits de concurrence des chaînes de transitions locales) stockés dans le vecteur v et par conséquence de réduire aussi le nombre de produits de concurrence du produit vecteur-matrice. L'expérience (4), l'*optimisation locale* ***et*** *positionnelle*, envisage d'exploiter ensemble le gain de ces deux optimisations afin de réduire le maximum possible le nombre de produits de concurrence du produit vecteur-matrice, ainsi que la quantité de mémoire utilisée pour stocker les éléments du vecteur.

Alors que les optimisations proposées réduisent potentiellement le nombre de produits de concurrence du produit vecteur-matrice, ainsi que la quantité de mémoire nécessaire pour la représentation des éléments (produits de concurrence des chaînes de transitions locales) du vecteur, ces optimisations n'assurent pas un gain réel de temps d'exécution pour l'utilisation de la méthode, car le coût de l'implantation de ces optimisations peut être élevé et ceci mérite d'être évalué.

Afin d'analyser l'impact des optimisations suggérées pour la méthode, on envisage de tester les expériences pour des modèles avec des différentes caractéristiques telles que :

(a) *un modèle SAN avec des événements* ***locaux seulement*** *dans tous les automates* ;
Un modèle avec des événements locaux seulement dans tous les automates est un modèle intéressant pour tester l'*optimisation locale*. On envisage d'appliquer l'optimisation locale, dès la multiplication du premier facteur normal, aux tuples de transition locale afin de propager ce gain pour la multiplication des éléments du vecteur par les facteurs normaux suivants. Un modèle avec ces caractéristiques, par exemple, est le cas des modèles du *Dîner des Philosophes* (FIG. 8.1, page 151) et *Partage de Ressources* (FIG. 8.4, page 156).

(b) *un modèle SAN avec* ***un seul événement synchronisant****, où* ***tous les automates*** *sont concernés par cet événement* ;
Un modèle avec un seul événement synchronisant, où tous les automates sont concernés par cet événement, est spécialement intéressant pour analyser l'impact de l'*optimisation positionnelle*. Dans ce cas, une fois qu'un tuple de transition de cet événement apparaît dans un élément du vecteur (*i.e.*, dans un produit de concurrence des chaînes de transitions locales), on peut alors appliquer l'optimisation positionnelle à cet élément et pour toutes les multiplications suivantes de cet élément par les facteurs normaux, afin d'éviter des multiplications pour les cas où cet élément sera forcément évaluée à $O_{\check{T}}$ (l'élément nul) lors de l'obtention de la chaîne de transitions globales.

(c) *un modèle SAN avec un* ***automate*** *qui possède des* ***événements synchronisants seulement*** *et* ***tous les événements synchronisants*** ;
Un modèle avec un automate qui possède seulement des événements synchronisants et tous les événements synchronisants du modèle est aussi un modèle particulièrement intéressant pour tester l'*optimisation positionnelle* proposée précédemment. Par exemple, un modèle avec ces caractéristiques est le cas du modèle du *Patron de Service Alterné* (l'automate $\mathcal{A}^{(3)}$ - FIG. 8.2, page 153). Sachant que le produit de concurrence est un opérateur *commutatif*, on peut re-ordonner les automates de façon à ce que cet automate (qui possède seulement des événements synchronisants et tous les événement synchronisants du modèle) soit le premier automate et par conséquence le facteur normal relatif à cet automate soit le premier facteur normal. On peut aussi appliquer cette optimisation à presque tous les facteurs qui suivent. Dans ce cas, on peut envisager de re-ordonner les automates en fonction de nombre d'événements synchronisants de façon à ce que le *premier automate* soit l'automate qui possède *seulement des événements synchronisants* et *tous les événements synchronisants* du modèle, suivi de l'automate qui possède *quasiment tous les événements synchronisants* du modèle et ainsi de suite jusqu'à l'automate qui possède *un ou aucun événement synchronisant*. Pour des modèles avec cette caractéristique, en re-ordonnant les automates et appliquant l'optimisation positionnelle, on peut réduire le nombre de produits de concurrence du produit vecteur-matrice dès la multiplication des éléments du vecteur par le deuxième facteur normal. On peut encore envisager plusieurs heuristiques de re-ordonnancement des automates. Cependant, ce travail ne sera pas étudié dans cette thèse et sera l'objet d'un travail ultérieur.

(d) *des modèles SAN* ***sans caractéristiques particulières***.
Il est aussi intéressant de tester les expériences suggérées pour des modèles SAN sans caractéristiques particulières. Ceci est le cas, par exemple, du modèle de l'*Atelier avec kanbans* (FIG. 8.3, page 155). On envisage de voir l'impact des optimisations (*positionnelle* et *locale*) pour ce modèle.

Les expériences proposées dans cette section vont permettre de comparer le gain réel de l'implantation des optimisations suggérées pour la méthode de la multiplication du vecteur par le descripteur présentée dans ce chapitre.

Dans la littérature, la représentation de la chaîne de Markov sous-jacente d'un modèle structuré par un descripteur est largement appliquée à des formalismes de modélisation en temps continu [111, 48, 49, 83, 55, 129, 21, 22, 39]. La représentation du descripteur discret par un *unique* produit tensoriel portant sur des facteurs qui décrivent ce qui se passe sur une seule dimension (une composante du modèle SAN) est un résultat *nouveau* très favorable pour la proposition d'autres méthodes (que celle du *Shuffle*) de multiplication d'un vecteur de probabilité par le descripteur.

Par exemple, dans [22], la génération des éléments de la matrice de transition de la chaîne de Markov représentée par le descripteur est faite lorsqu'ils sont requis, *i.e.*, une génération des éléments qui a lieu *à la volée* [44]. Sachant que le descripteur de la méthode présentée dans ce chapitre est représenté par un *seul* produit tensoriel, on peut envisager une adaptation de cette méthode à la volée de façon à ce que les éléments du descripteur soient générés dynamiquement. Cette approche a comme effet de réduire la quantité de mémoire utilisée dans la multiplication vecteur-descripteur, car une fois qu'un élément du descripteur est requis pour multiplier un élément du vecteur, cet élément est généré (*i.e.*, le produit de concurrence des chaînes de transitions locales est réalisé et réduit à une chaîne de transitions globales), la probabilité de cet élément est calculée, puis l'élément est détruit.

D'ailleurs, les optimisations proposées précédemment (*i.e.*, l'optimisation locale **et/ou** positionnelle) peuvent aussi être employées dans la méthode qui génère les éléments du descripteur à la volée.

11.4 Conclusion

Dans ce chapitre, on a présenté une méthode de "multiplication" d'un vecteur de probabilité par un descripteur discret d'un modèle décrit par le formalisme SAN à temps discret. Le descripteur discret d'un modèle SAN est la façon compacte par laquelle on peut décrire les transitions du modèle. Un aspect important de cette méthode est qu'elle permet de "multiplier" un vecteur de probabilité par le descripteur discret du modèle, sans jamais générer la matrice de transition de la chaîne de Markov de ce modèle.

En profitant de la représentation tensorielle du descripteur, la méthode réalise une suite d'opérations qui manipulent des donnés de la taille de l'espace d'états d'une composante (*i.e.*, d'un automate) du modèle au lieu de réaliser des opérations sur l'espace d'états produit de ce modèle. Cette suite d'opérations est possible grâce à la propriété de décomposition des produits tensoriels complexes en facteurs normaux.

Cette méthode emploie une approche inspirée de l'algorithme *Shuffle*, utilisé pour les modèles SAN à temps continu [55], et elle est la première approche de la "multiplication" d'un vecteur par un descripteur (l'opération fondamentale des méthodes itératives pour la résolution d'un modèle) d'un modèle SAN à temps discret qui utilise la sémantique présentée dans le chapitre 7.

En se basant sur certaines propriétés des opérateurs utilisés pour la description du formalisme SAN à temps discret, telles que *le degré d'idempotence des événements locaux* et *l'aspect positionnel des événements synchronisants* dans un produit de concurrence, on a proposé des optimisations pour la méthode présentée dans ALG. 11.3 afin de réduire le nombre de produits de concurrence nécessaires pour réaliser

le produit vecteur-matrice. De plus, ces optimisations permettent une réduction dans la représentation (stockage) des éléments intermédiaires calculés (produits de concurrence des chaînes de transitions locales) du vecteur.

Chapitre 12

Conclusion et perspectives

L'apport scientifique de cette thèse est centré sur le formalisme des *Réseaux d'Automates Stochastiques* (SAN - *Stochastic Automata Networks*) et s'articule autour de deux axes de recherche : les *formalismes de modélisation* et les *méthodes de calcul*.

Les principales contributions de cette thèse sont : (1) la définition de *méthodes efficaces de génération de l'espace d'états atteignables* de modèles SAN à temps continu basées sur la formulation tensorielle ; (2) la définition d'une *méthode de multiplication d'un vecteur de probabilité par le descripteur* d'un modèle à temps discret exprimé sous forme d'un produit tensoriel complexe, sans jamais générer la matrice qui représente ce descripteur.

Dans cette conclusion, nous présentons un bilan de ces contributions en rappelant les principaux résultats obtenus et les mises en ouvre pour les obtenir. Et, dans la section 12.3, nous mentionnons des perspectives de travaux ouvertes grâce aux travaux développés dans cette thèse.

12.1 Méthodes de génération de l'espace d'états atteignables

Les travaux précédents de méthodes de génération de l'espace d'états atteignables (RSS - *Reachable State Space*) [94, 32, 30] considèrent l'utilisation de fonctions, où les arguments de ces fonctions sont limités à l'espace d'états local d'un sous-système (*i.e.*, fonctions qui peuvent être évaluées à priori).

Afin de permettre la génération de l'espace d'états atteignables de modèles qui utilisent des fonctions (*sans hypothèse concernant les arguments*), nous avons montré comment nous pouvons représenter par un *Diagramme de Décision Multi-valués* (MDD) l'*ensemble d'états où une fonction ne s'annule pas* et nous avons présenté une formulation tensorielle (dénommée *descripteur d'atteignabilité*) qui sépare les éléments des matrices suivant les sous-espaces où elles peuvent devenir identiquement nulles.

En se basant sur des méthodes précédentes de génération de l'espace d'états atteignables de modèles compositionnels, nous avons défini des *méthodes efficaces de génération de l'espace d'états atteignables de modèles SAN à temps continu basée sur la formulation tensorielle du descripteur d'atteignabilité*. Ces nouvelles méthodes de génération du RSS de modèles compositionnels *prennent en compte de l'utilisation de fonctions qui expriment des relations entre les composantes des modèles*.

La génération de l'espace d'états atteignables d'un modèle compositionnel est reconnue comme un problème difficile du fait de l'*explosion combinatoire de l'espace d'états* du modèle. Notamment, pour des modèles qui ont des transitions qui dépendent de l'état global, *i.e.*, qui expriment des relations entre les composantes des modèles. Des fonctions avec cette caractéristique ne peuvent pas être évaluées à priori, puisqu'elles doivent être évaluées pendant la génération de l'espace d'états atteignables du modèle, et rendent plus complexe le calcul de l'espace d'états atteignables.

L'utilisation des méthodes de génération présentées dans cette thèse n'est pas limitée au formalisme SAN, ces méthodes peuvent être employées pour d'autres formalismes structurés de haut-niveau qui possèdent une représentation généralisée de Kronecker.

Ces méthodes, outre qu'elles permettent l'utilisation de *fonctions sans hypothèse sur les arguments*, permettent aussi la représentation d'un très grand espace d'états atteignables d'un modèle SAN par un MDD. La représentation du RSS de modèles SAN par un MDD constitue une amélioration importante lorsqu'on compare avec la représentation vectorielle courante du RSS de modèles SAN, car nous pouvons *calculer le RSS de modèles plus grands d'une façon performante*, alors que *le pic de mémoire utilisée reste relativement bas*.

12.2 Méthode de multiplication vecteur-descripteur discret

Des méthodes numériques (notamment les méthodes itératives) sont particulièrement bien adaptées aux systèmes à très grand espace d'états. Une des grandes difficultés pour l'application de ces méthodes est la multiplication d'un vecteur de probabilité par le descripteur du modèle.

La représentation de la chaîne de Markov sous-jacente d'un modèle structuré par un descripteur est largement appliquée pour des formalismes de modélisation en temps continu [111, 48, 49, 83, 55, 129, 21, 22, 39].

En ce qui concerne les systèmes modélisés sur une *échelle de temps discrète*, à chaque intervalle de temps, *plusieurs* événements peuvent avoir lieu. La grande difficulté pour ces modèles est le calcul effectif de la formulation matricielle de la chaîne de Markov sous-jacente. Cette difficulté vient essentiellement de la combinatoire générée par la possibilité d'avoir des événements simultanés.

Dans la littérature, nous trouvons un certain nombre de propositions de formalismes pour modéliser des systèmes à temps discret : *Réseaux de Petri Stochastiques* [97, 29], *Réseaux de Files d'Attente* [68] et *Réseaux d'Automates Stochastiques* [112, 8].

Au niveau des Réseaux de Petri Stochastiques, Molloy dans [97] propose une sémantique pour traiter la concurrence de transitions (*i.e.*, deux transitions réalisables dans la même unité de temps), où on insère des probabilités de choix aux transitions. Ciardo dans [29] propose d'associer un poids à chaque transition et lorsque les deux transitions sont potentiellement réalisables en même temps, la probabilité de la transition effectivement réalisée est pondérée par la probabilité de chaque transition. Dans ces deux approches, la simultanéité est écartée alors qu'elle peut avoir un sens physique dans certains systèmes.

Au niveau des Réseaux d'Automates Stochastiques, Benoit dans [8] introduit la notion d'*ordre de priorité* entre les événements du modèle SAN à temps discret afin de préciser le comportement à suivre en cas de conflit, lorsque le tirage d'un événement amène à un état où un autre n'est plus réalisable. Ce travail a présenté un algorithme pour générer la chaîne de Markov à temps discret sous-jacente, mais il ne propose pas de représentation tensorielle du générateur de cette chaîne de Markov.

Dans le but de gérer cette combinatoire d'événements dans un modèle à temps discret, nous avons proposé une *algèbre tensorielle* (appelée *Algèbre Tensorielle compleXe - ATX*) adaptée à la composition parallèle des *Réseaux d'Automates Stochastiques à temps discret*. Grâce à cette algèbre, il est possible de représenter de façon compacte par une formulation tensorielle (appelée *descripteur discret*) les transitions de la chaîne de Markov sous-jacente d'un modèle SAN à temps discret.

La sémantique de l'algèbre tensorielle complexe *cache à l'utilisateur* du formalisme *toute la complexité d'un modèle à temps discret*, ce qui fait de cette sémantique un *outil puissant pour la construction de modèles SAN à temps discret*. Plus précisément, *les règles de l'algèbre appliquées sur les événements* masquent la complexité de la *combinatoire générée par la possibilité d'avoir des événements simultanés dans le même automate*, alors que *le produit tensoriel complexe*, outre qu'il masque l'*explosion combinatoire de l'espace d'états* du modèle, gère aussi *la modélisation de l'occurrence d'événements concurrents* (*i.e.*, l'occurrence d'événements dans différents automates dans la même unité de temps).

Nous avons démontré quelques propriétés de l'algèbre tensorielle complexe qui *permettent la décomposition du descripteur discret en facteurs normaux*. Cette décomposition sert de base aux méthodes itératives pour la résolution d'un modèle SAN à temps discret. Les *définitions de l'algèbre tensorielle complexe*, ainsi que *les propriétés de cette algèbre* présentées dans cette thèse sont des *travaux innovants* qui ont permis, outre *la représentation compacte* des modèles, la proposition de *méthodes de résolution de modèles pour une sémantique en temps discret*.

En se basant sur l'algorithme de *Shuffle* [55] largement utilisé pour la résolution de modèles SAN à temps continu, nous avons présenté une *méthode de multiplication d'un vecteur de probabilité par un descripteur discret d'un modèle SAN à temps discret, sans jamais générer la matrice de transition qui représente ce descripteur*.

L'idée de base de cette méthode est de profiter de la représentation tensorielle du descripteur de façon à ce que la multiplication par *un opérateur sur l'espace produit* soit remplacée par *une suite d'opérations qui manipulent des données de la taille d'une composante* (et pour toutes les composantes). De cette façon, la méthode réalise une suite d'opérations qui *manipulent des donnés de la taille de l'espace d'états d'une composante* (*i.e.*, d'un automate) du modèle *au lieu de réaliser des opérations sur l'espace d'états produit* de ce modèle. Cette suite d'opérations est possible grâce à la propriété de décomposition des produits tensoriels complexes en facteurs normaux de l'algèbre tensorielle complexe.

Nous avons aussi proposé des optimisations pour cette méthode en vue de réduire le nombre de produits de concurrence nécessaires pour réaliser la multiplication vecteur-descripteur, ainsi que la quantité de mémoire utilisée dans la représentation (stockage) des éléments intermédiaires calculés du vecteur.

12.3 Perspectives

Les perspectives de cette thèse sont multiples : les perspectives à *court*, *moyen* et *long* terme. Les perspectives à court terme concernent les travaux d'implantation de la méthode de multiplication en temps discret. Les perspectives à moyen terme concernent les travaux autour des *méthodes de génération* pour modèles à temps continu et de la *méthode de multiplication vecteur-descripteur* pour des modèles à temps discret. Les perspectives à long terme implique dans les travaux où des apports dans d'autres domaines sont requises.

12.3.1 Perspectives à court terme

Implantation de la méthode de multiplication vecteur-descripteur (SAN discret)

Nous envisageons dans un futur immédiat l'*implantation* de la méthode, ainsi que les optimisations suggérées, de multiplication vecteur-descripteur d'un modèle SAN à temps discret. Cette implantation sera développée dans un module du logiciel PEPS (*Performance Evaluation of Parallel Systems*) [17, 106] pour l'obtention des résultats numériques.

Les mesures numériques pour tester l'efficacité de la méthode proposée doivent être effectuées pour les expériences présentées dans le chapitre 11 (Section 11.3, page 222). Ces expériences vont permettre de comparer le gain réel (de *temps d'exécution* et *mémoire utilisée*) de l'implantation des optimisations suggérées pour la méthode de multiplication vecteur-descripteur présentée dans cette thèse.

12.3.2 Perspectives à moyen terme

Évaluation des fonctions pour la génération de l'espace d'états atteignables (SAN continu)

Dans le but de profiter de l'utilisation des taux et probabilités fonctionnels dans la description des modèles, nous avons montré comment les taux et probabilités fonctionnels ont été utilisées au sein des méthodes de génération présentées dans cette thèse.

Nous avons représenté par un MDD l'*ensemble d'états d'une fonction où la fonction ne s'annule pas*. La génération de ce MDD est une tâche relativement simple, sachant qu'il est obtenu par l'union de tous les états pour lesquels l'évaluation de la fonction est différente de zéro. Autrement dit, nous avons évalué une fonction (de façon *explicite*) sur tous les états de son domaine et donc nous avons produit un MDD en faisant l'union de tous les états pour lesquels l'évaluation de la fonction est différente de zéro. Cependant, comme nous l'avons montré dans les résultats présentés dans le chapitre 6, la génération de ce MDD d'une fonction sans hypothèse sur les arguments peut avoir un coût de calcul très élevé.

Il est possible d'envisager une évolution de façon à ce que *l'évaluation d'une fonction soit seulement faite lorsqu'elle est requise*. Cette démarche mérite une étude approfondie afin d'analyser l'impact de cette adaptation dans ce contexte.

Cette approche peut apporter des gains considérables en temps d'exécution de la génération de l'espace d'états de modèles qui utilisent des *fonctions à très grand espace d'états*.

Diagramme de Matrices (SAN continu)

Dans les méthodes de génération de l'espace d'états atteignables présentées dans cette thèse, nous avons représenté la fonction "*next-state*" par le descripteur d'atteignabilité (présenté dans le chapitre 4). On rappelle que le descripteur d'atteignabilité d'un modèle est un descripteur restreint aux relations d'atteignabilité de ce modèle, sans considérer les valeurs des taux.

La génération de l'espace d'états atteignables est une première étape pour l'évaluation d'un modèle. La représentation compacte de la matrice de transition pour la résolution du modèle est une autre étape à réaliser. Une démarche performante et largement utilisée pour la représentation de la matrice de transition de modèles décrits par le formalisme des Réseaux de Petri Stochastiques est l'utilisation des *diagrammes de matrices* (*MxD - Matrix Diagram*) [91, 92]. Un diagramme de matrice est une structure similaire à la structure d'un MDD, sauf que *les noeuds non-terminaux sont des matrices* dont les éléments sont des valeurs réelles et des pointeurs pour des autres noeuds.

Nous envisageons d'utiliser des *diagrammes de matrices* pour la représentation de la fonction "*next-state*" afin de profiter des méthodes de génération basées sur la saturation adaptées aux *modèles à temps continu qui utilisent des fonctions*.

Pré-évaluation des fonctions pour la méthode de multiplication (SAN discret)

L'utilisation des *éléments fonctionnels* (*i.e.*, la *probabilité d'occurrence fonctionnelle* d'un événement ou la *probabilité de routage fonctionnelle* d'une transition lors de l'occurrence d'un événement) est l'une des caractéristiques des modèles SAN.

Les éléments résultants de la multiplication vecteur-descripteur de la méthode présentée dans cette thèse sont des *chaînes de transitions globales qui peuvent disposer d'éléments fonctionnels*. Afin d'exploiter la puissance de l'algorithme *Shuffle* [54] (*i.e.*, de réaliser le produit de concurrence des éléments du vecteur par les matrices le plus tôt possible) dans les modèles à temps discret, nous envisageons de réaliser une *pré-évaluation des fonctions* des éléments fonctionnels.

La pré-évaluation d'une fonction peut être réalisée à partir du moment où les états des automates arguments de cette fonction sont connus. Dans ce cas, si le résultat de l'évaluation d'une fonction est *égal à zéro*, la chaîne de transitions globales qui contient cette fonction est *aussi évaluée à zéro*. Si le résultat est *différent de zéro*, ce résultat est multiplié par la probabilité du tuple de transition de l'événement qui contient cette fonction et alors cette fonction n'est plus évaluée. Toutefois, pour des modèles à temps discret, les règles qui permettent de savoir les états des automates arguments d'une fonction sont complexes à cause de la combinatoire d'événements possibles dans la même unité de temps. La pré-évaluation des fonctions est une optimisation qui mérite d'être étudiée attentivement.

En se basant sur l'idée employée dans l'algorithme de *Shuffle*, il est aussi possible d'imaginer un *re-ordonnancement* des automates du modèle afin de que les éléments fonctionnels soient évalués le plus tôt possible.

12.3.3 Perspectives à long terme

Génération de l'espace d'états atteignables de modèles à temps discret (du continu au discret)

Les méthodes proposées dans cette thèse pour la génération de l'espace d'états atteignables de modèles à temps continu tirent parti de la représentation et manipulation performantes d'espaces d'états par des MDD. Nous envisageons d'utiliser cette même démarche pour des modèles à temps discret. Cette démarche semble possible pour les *modèles SAN à temps discret*, cependant, une adaptation à la sémantique des événements (*i.e.*, la possibilité d'avoir des événements simultanés) doit être étudiée.

Comme nous l'avons déjà dit, contrairement aux modèles à temps continu, dans les systèmes modélisés sur une échelle de temps discrète, à chaque intervalle de temps, *plusieurs* événements peuvent avoir lieu dans le même instant. Dans ce cas, il est important de considérer la sémantique utilisée pour la simultanéité des événements afin d'obtenir le *descripteur d'atteignabilité d'un modèle SAN à temps discret*.

Les matrices du descripteur d'atteignabilité d'un *modèle SAN à temps continu* sont obtenues à partir des matrices du *descripteur Markovien* de ce modèle. Contrairement au descripteur Markovien des modèles SAN à temps continu, le *descripteur discret* est représenté par un unique produit tensoriel complexe portant sur des facteurs qui décrivent ce qui se passe sur un seul automate du modèle SAN à temps discret. Dans ce cas, l'obtention du descripteur d'atteignabilité d'un *modèle SAN à temps discret* n'est pas fait de la même façon. Une étude méticuleuse doit être réalisée afin d'analyser l'impact de la combinatoire d'événements possible pour des modèles à temps discret en vue d'obtenir le *descripteur d'atteignabilité d'un modèle SAN à temps discret*.

D'ailleurs, de la même façon que nous l'avions envisagé, d'utiliser des *diagrammes de matrices* pour la représentation de la fonction "*next-state*" pour des *modèles SAN à temps continu*, il est aussi possible d'envisager l'utilisation des *diagrammes de matrices pour des modèles SAN à temps discret*.

Proposition d'autres méthodes de multiplication (SAN discret)

La représentation tensorielle du descripteur discret par un unique produit tensoriel complexe portant sur des facteurs qui décrivent ce qui se passe sur une seule composante du modèle SAN est une représentation adéquate pour la *proposition d'autres méthodes de multiplication vecteur-descripteur*.

La méthode présentée dans cette thèse est inspirée de l'algorithme de *Shuffle* qui est amplement utilisée pour des modèles à temps continu. Il semble possible d'envisager la proposition d'autres méthodes de multiplication vecteur-descripteur pour une sémantique en temps discret qui tirent parti de caractéristiques des méthodes de multiplication vecteur-descripteur employées aux formalismes de modélisation en temps continu. Ceci est le cas, par exemple, du travail présenté dans [22] pour le temps continu.

Dans [22], la génération des éléments de la matrice de transition de la chaîne de Markov représentée par le descripteur est faite lorsqu'ils sont requis, *i.e.*, une génération des éléments qui a lieu *à la volée* [44]. Nous pouvons envisager une adaptation de cette méthode à la volée de façon à ce que les éléments du descripteur discret soient générés dynamiquement.

Utilisation de l'ATX dans d'autres formalismes de modélisation (SAN discret)

Comme nous l'avons mentionné précédemment, plusieurs formalismes de modélisation en temps continu [111, 48, 49, 83, 55, 129, 21, 22, 39] profitent de la représentation structurée du modèle pour représenter la chaîne de Markov sous-jacente par un descripteur.

En se basant sur l'ATX (*Algèbre Tensorielle compleXe*) présentée dans cette thèse et de *la représentation tensorielle particulière du descripteur discret* d'un modèle SAN, nous pouvons envisager d'utiliser la sémantique employée aux événements d'un modèle SAN à temps discret à d'autres formalismes de modélisation à temps continu dans le but de *proposer une formulation tensorielle pour ces formalismes*, tirant parti du pouvoir de description et de représentation du formalisme SAN à temps discret.

De plus, nous pouvons aussi envisager une comparaison du pouvoir de description du formalisme des *Réseaux d'Automates Stochastiques* à temps discret avec d'autres formalismes analogues (*e.g.*, *Réseaux de Petri Stochastiques* [97, 29] et *Réseaux de Files d'Attente* [68]) afin d'explorer et étudier l'ensemble des systèmes qui pourront être mieux décrits par ce formalisme.

Simulation (SAN discret)

En se basant sur le format particulier du descripteur discret (*i.e.*, représenté par un *unique* produit tensoriel complexe), il est possible d'envisager des méthodes de simulation qui profitent de la structure tensorielle du descripteur et tirent aussi parti de la puissance de description du formalisme des Réseaux d'Automates Stochastiques à temps discret.

Par exemple, des méthodes de simulation qui prennent en compte les priorités des événements, ou bien des méthodes qui réalisent la simulation directement sur *les chaînes de transitions globales* au lieu de l'occurrence de *simples événements*.

Des méthodes de simulation parfaite [131] ont déjà été proposée pour le formalisme SAN à temps continu [57]. Le même concept pourrait être exploité pour le formalisme SAN à temps discret.

Vérification de systèmes (SAN continu et discret)

Nous envisageons la possibilité de développer un module pour le logiciel PEPS que puisse réaliser de la vérification de propriétés. Le grand défi pour la plus grande partie des outils de vérification formelle (*e.g.*, le vérificateur de modèles [37]) est la génération de l'espace d'états atteignables du modèle.

Les méthodes de génération de l'espace d'états atteignables de modèles présentées dans cette thèse permettent la représentation performante de très grands espaces d'états par un MDD. Sachant que les MDD représentent et manipulent de façon efficace un très grand espace d'états, il semble possible d'envisager un vérificateur symbolique basé sur MDD (*CTL - Computation Tree Logic*) qui vérifie une *propriété* de modèles décrits par le formalisme des Réseaux d'Automates Stochastiques à temps continu et temps discret.

La proposition, ainsi que l'implantation d'un tel module pour le logiciel PEPS reste encore comme un défi à être remporté.

Bibliographie

[1] M. Ajmone-Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, 1995.

[2] M. Ajmone-Marsan, G. Conte, and G. Balbo. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 2(2) :93–122, 1984.

[3] S. Alouf, F. Huet, and P. Nain. Forwarders vs. centralized server : an evaluation of two approaches for locating mobile agents. *Performance Evaluation*, 49(1-4) :299–319, 2002.

[4] K. Atif. *Modélisation du parallélisme et de la synchronisation*. PhD thesis, Institut National Polytechnique de Grenoble, France, 1992.

[5] L. Baldo, L. G. Fernandes, P. Roisenberg, P. Velho, and T. Webber. Parallel PEPS Tool Performance Analysis using Stochastic Automata Networks. In M. Donelutto, D. Laforenza, and M. Vanneschi, editors, *Euro-Par 2004 International Conference on Parallel Processing*, volume 3149 of *Lecture Notes in Computer Science*, pages 214–219, Pisa, Italy, August/September 2004. Springer-Verlag Heidelberg.

[6] F. Baskett, K. Chandy, R. Muntz, and F. Palacios. Open, Closed, and Mixed Networks of Queues with Different Classes of Customers. *Journal of the ACM*, 22(2) :248–260, 1975.

[7] R. Bellman. *Introduction to Matrix Analysis*. McGraw-Hill, New York, 1960.

[8] A. Benoit. *Méthodes et algorithmes pour l'évaluation des performances des systèmes informatiques à grand espace d'états*. PhD thesis, Institut National Polytechnique de Grenoble, France, 2003.

[9] A. Benoit, P. Fernandes, B. Plateau, and W. J. Stewart. On the benefits of using functional transitions and Kronecker algebra. *Performance Evaluation*, 58(4) :367–390, 2004.

[10] A. Benoit, B. Plateau, and W. J. Stewart. Memory-efficient Kronecker algorithms with applications to the modelling of parallel systems. In *Proceedings of International Parallel and Distributed Processing Symposium*, pages 275–282, Nice, France, April 2003.

[11] M. Bernardo and R. Gorrieri. A tutorial on EMPA : A theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 2002(1-2) :1–54, 1998.

[12] C. Bertolini, L. Brenner, P. Fernandes, A. Sales, and A. F. Zorzo. Structured Stochastic Modeling of Fault-Tolerant Systems. In 12^{th} *IEEE/ACM Internacional Symposium on Modelling, Analysis and Simulation on Computer and Telecommunication Systems (MASCOTS'04)*, pages 139–146, Volendam, The Netherlands, October 2004. IEEE Computer Society.

[13] V. S. Borkar. Control of Markov chains with long-run average cost criterion : the dynamic programming equations. *SIAM Journal on Control and Optimization*, 27(3) :642–657, 1989.

[14] G. W. Brams. *Réseaux de Petri : théorie et pratique*. Masson, Paris, 1983.

[15] L. Brenner. *Réseaux d'Automates Stochastiques : Analyse transitoire en temps continu et Algèbre tensorielle pour une sémantique en temps discret*. PhD thesis, Institut Polytechnique de Grenoble, France, 2009.

[16] L. Brenner, P. Fernandes, J. M. Fourneau, and B. Plateau. Modelling Grid5000 point availability with SAN. *Electronic Notes in Theoretical Computer Science*, 232 :165–178, 2009.

[17] L. Brenner, P. Fernandes, B. Plateau, and I. Sbeity. PEPS2007 - Stochastic Automata Networks Software Tool. In *Fourth International Conference on the Quantitative Evaluation of Systems (QEST'07)*, pages 163–164, Edinburgh, UK, 2007. IEEE Computer Society Press.

[18] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8) :667–691, 1986.

[19] R. E. Bryant. Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3) :293–318, 1992.

[20] P. Buchholz. A class of hierarchical queueing networks and their analysis. *Queueing Systems*, 15(1-4) :59–80, 1994.

[21] P. Buchholz. Structured Analysis Approaches for Large Markov Chains. *Applied Numerical Mathematics*, 31(4) :375–404, 1999.

[22] P. Buchholz, G. Ciardo, S. Donatelli, and P. Kemper. Complexity of memory-efficient Kronecker operations with applications to the solution of Markov models. *INFORMS Journal on Computing*, 13(3) :203–222, 2000.

[23] P. Buchholz and P. Kemper. Hierarchical reachability graph generation for Petri nets. *Formal Methods in Systems Design*, 21(3) :281–315, 2002.

[24] J. R. Burch, E. M. Clarke, and D. E. Long. Symbolic Model Checking with Partitioned Transition Relations. In *International Conference on Very Large Scale Integration*, pages 49–58, Edinburgh, Scotland, 1991. North-Holland.

[25] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic Model Checking : 10^{20} states and beyond. *Information and Computation*, 98(2) :142–170, June 1992.

[26] K. M. Chandy and C. H. Sauer. Approximate Methods for Analyzing Queueing Network Models of Computing Systems. *ACM Computing Surveys*, 10(3) :281–317, 1978.

[27] R. Chanin, M. Corrêa, P. Fernandes, A. Sales, R. Scheer, and A. F. Zorzo. Analytical Modeling for Operating System Schedulers on NUMA Systems. *Electronic Notes in Theoretical Computer Science*, 151(3) :131–149, 2006.

[28] G. Chiola. Compiling Techniques for the Analysis of Stochastic Petri Nets. In *Proceedings of the 4^{th} International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, pages 11–24, Palma de Mallorca, Spain, Septembre 1988.

[29] G. Ciardo. Discrete-time Markovian stochastic Petri nets. In *Proceedings of the 2^{th} International Conference on the Numerical Solution of Markov Chains*, pages 339–358, Raleigh, NC, USA, 1995.

[30] G. Ciardo, G. Lüttgen, and A. S. Miner. Exploiting interleaving semantics in symbolic state-space generation. *Formal Methods in System Design*, 31(1) :63–100, 2007.

[31] G. Ciardo, G. Lüttgen, and R. Siminiceanu. Efficient Symbolic State-Space Construction for Asynchronous Systems. In *International Conference on Applications and Theory of Petri Nets*, volume 1825 of *Lecture Notes in Computer Science*, pages 103–122. Springer-Verlag, 2000.

[32] G. Ciardo, G. Lüttgen, and R. Siminiceanu. Saturation : An Efficient Iteration Strategy for Symbolic State-Space Generation. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 2031 of *Lecture Notes in Computer Science*, pages 328–342. Springer-Verlag, 2001.

[33] G. Ciardo and A. S. Miner. Storage Alternatives for Large Structured State Spaces. In *9th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, volume 1245 of *Lecture Notes in Computer Science*, pages 44–57, St. Malo, France, 1997. Springer-Verlag Heidelberg.

[34] G. Ciardo and A. J. Yu. Saturation-based symbolic reachability analysis using conjunctive and disjunctive partitioning. In *Proceedings of the Correct Hardware Design and Verification Methods (CHARME'05)*, volume 3725 of *Lecture Notes in Computer Science*, pages 146–161, Saarbrücken, Germany, 2005. Springer.

[35] A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri. NuSMV : A New Symbolic Model Verifier. In *Computer-Aided Verification (CAV'99)*, volume 1633 of *Lecture Notes in Computer Science*, pages 495–499, Trento, Italy, 1999. Springer-Verlag Heidelberg.

[36] 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, 1986.

[37] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

[38] P. J. Courtois. *Decomposability : queueing and computer systems applications*. London : Academic Press, 1977.

[39] R. M. Czekster, P. Fernandes, J. M. Vincent, and T. Webber. Split : a flexible and efficient algorithm to vector-descriptor product. In *Proceedings of the* 2^{nd} *International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'07)*, page 83, Nantes, France, 2007. ACM.

[40] Y. Dallery. Approximate Analysis of General Open Queuing Networks with Restricted Capacity. *Performance Evaluation*, 11(3) :209–222, 1990.

[41] Y. Dallery, Z. Liu, and D. Towsley. Equivalence, reversibility, symmetry and concavity properties in fork-join queuing networks with blocking. *Journal of the ACM*, 41(5) :903–942, 1994.

[42] M. Davio. Kronecker Products and Shuffle Algebra. *IEEE Transactions on Computers*, C-30(2) :116–125, 1981.

[43] M. Davio, J. P. Deschamps, and A. Thayse. *Discrete and switching functions*. New York : McGraw-Hill, 1978.

[44] D. D. Deavours and W. H. Sanders. "On-the-Fly" Solution Techniques for Stochastic Petri Nets and Extensions. *IEEE Transactions on Software Engineering*, 24(10) :889–902, 1998.

[45] F. Delamare, F. L. Dotti, P. Fernandes, C. M. Nunes, and L. C. Ost. Analytical modeling of random waypoint mobility patterns. In *Proceedings of the* 3^{rd} *ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN'06)*, pages 106–113, Terromolinos, Spain, October 2006. ACM Press.

[46] P. J. Denning and J. P. Buzen. The Operational Analysis of Queueing Network Models. *ACM Computing Surveys*, 10(3) :225–261, 1978.

[47] E. Dijkstra. Hierarchical Ordering of Sequential Processes. *Acta Informatica*, 1 :115–138, 1971.

[48] S. Donatelli. Superposed stochastic automata : a class of stochastic Petri nets with parallel solution and distributed state space. *Performance Evaluation*, 18 :21–36, 1993.

[49] S. Donatelli. Superposed generalized stochastic Petri nets : definition and efficient solution. In R. Valette, editor, *Proceedings of the* 15^{th} *International Conference on Applications and Theory of Petri Nets*, pages 258–277. Springer-Verlag Heidelberg, 1994.

[50] F. L. Dotti, P. Fernandes, A. Sales, and O. M. Santos. Modular Analytical Performance Models for Ad Hoc Wireless Networks. In 3^{rd} *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 164–173, Trentino, Italy, April 2005. IEEE Computer Society.

[51] C. Durbach, F. Quessette, and A. Troubnikoff. Solving Large Markov Model based on Stochastic Automata Network. In *Advances in Computer and Informmation Sciences 1998, IOS press*, Belek-Antalya, Turkey, 1998.

[52] E. A. Souza e Silva and R. R. Muntz. Métodos Computacionais de solução de Cadeias de Markov : aplicações a sistemas de computação e comunicação. In *VIII Escola de Computação*, Instituto de Informática, UFRGS, Porto Alegre, 1992.

[53] A. G. Farina, P. Fernandes, and F. M. Oliveira. Representing software usage models with Stochastic Automata Networks. In *Proceedings of the* 14^{th} *International Conference on Software Engineering and Knowledge Engineering*, pages 401–407. ACM Press, 2002.

[54] P. Fernandes. *Méthodes numériques pour la solution de systèmes Markoviens à grand espace d'états*. PhD thesis, Institut National Polytechnique de Grenoble, France, 1998.

[55] P. Fernandes, B. Plateau, and W. J. Stewart. Efficient descriptor-vector multiplication in Stochastic Automata Networks. *Journal of the ACM*, 45(3) :381–414, 1998.

[56] P. Fernandes, R. Presotto, A. Sales, and T. Webber. An Alternative Algorithm to Multiply a Vector by a Kronecker Represented Descriptor. In 21^{st} *UK Performance Engineering Workshop*, pages 57–67, Newcastle, UK, June 2005.

[57] P. Fernandes, J-M. Vincent, and T. Webber. Perfect Simulation of Stochastic Automata Networks. In 15^{th} *International Conference Analytical and Stochastic Modeling Techniques and Applications (ASMTA 2008)*, volume 5055 of *Lecture Notes in Computer Science*, pages 249–263, Nicosia, Cyprus, June 2008. Springer-Verlag Heidelberg.

[58] G. Florin and S. Natkin. Les réseaux de Petri stochastiques. *Tecniques et Sciences Informatiques*, 4(1) :143–160, 1985.

[59] J. M. Fourneau. Discrete time stochastic automata networks : using structural properties and stochastic bounds to simplify the SAN. In *Proceedings of the* 2^{nd} *International Conference on Performance Evaluation Methodolgies and Tools*, ACM International Conference Proceeding Series, pages 1–10, Nantes, France, October 2007. ACM.

[60] J. M. Fourneau. Product Form Steady-State Distribution for Stochastic Automata Networks with Domino Synchronizations. In *Proceedings of the* 5^{th} *European Performance Engineering Workshop*, volume 5261 of *Lecture Notes in Computer Science*, pages 110–124, Palma de Mallorca, Spain, September 2008. Springer.

[61] J. M. Fourneau, M. Lecoz, and F. Quessette. Algorithms for an irreducible and lumpable strong stochastic bound. *Linear Algebra and Applications*, 386 :167–186, 2003.

[62] J. M. Fourneau, B. Plateau, and W. J. Stewart. Product form for stochastic automata networks. In *Proceedings of the* 2^{nd} *International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'07)*, page 32, Nantes, France, 2007. ACM.

[63] J. M. Fourneau, B. Plateau, and W. J. Stewart. An algebraic condition for product form in stochastic automata networks without synchronizations. *Performance Evaluation*, 65(11-12) :854–868, 2008.

[64] J. M. Fourneau and F. Quessette. Graphs and Stochastic Automata Networks. In *International Workshop on the Numerical Solution of Markov Chains*, Raleigh, USA, Janvier 1995.

[65] H. Garavel and H. Hermanns. On Combining Functional Verification and Performance Evaluation Using CADP. In *Proceedings of the International Symposium of Formal Methods Europe*, volume 2391 of *Lecture Notes in Computer Science*, pages 410–429, Copenhagen, Denmark, July 2002. Springer-Verlag.

[66] E. Gelenbe. Product form queueing networks with negative and positive customers. *Journal of Applied Probability*, 28 :656–663, 1991.

[67] E. Gelenbe and I. Mitrani. *Analysis and synthesis of computer systems*. Academic Press, London and New York, 1980.

[68] E. Gelenbe and G. Pujolle. *Introduction to queueing networks*. John Wiley & Sons, New York, USA, 1987.

[69] N. Gootz, U. Herzog, and M. Rettelbach. Tipp - a stochastic process algebra. In *Proceedings of Workshop on Process Algebra and Performance Modelling (PAPM'93)*, pages 427–430, Edinburgh, UK, 1993.

[70] O. Gusak, T. Dayar, and J. M. Fourneau. Iterative disaggregation for a class of lumpable discrete-time stochastic automata networks. *Performance Evaluation*, 53(1) :43–69, 2003.

[71] S. Haddad. *Une catégorie régulière de réseaux de Petri de haut niveau : définition, propriétés et réduction*. PhD thesis, Université Paris VI, Paris, 1987.

[72] S. Haddad. A reduction theory for colored nets. In *High-Level Petri Nets : Theory and Application*, Springer-Verlag Heidelberg, 1991.

[73] B. R. Haverkort, A. Bell, and H. Bohnenkamp. On the Efficient Sequential and Distributed Generation of Very Large Markov Chains from Stochastic Petri Nets. In *Proceedings of the The* 8^{th} *International Workshop on Petri Nets and Performance Models*, pages 12–21, Zaragoza, Spain, 1999. IEEE Computer Society.

[74] J. Hillston. Compositional Markovian Modelling Using a Process Algebra. In W. J. Stewart, editor, *Proceedings of the* 2^{nd} *International Workshop on the Numerical Solution of Markov Chains*. Kluwer, 1995.

[75] J. Hillston and L. Kloul. An Efficient Kronecker Representation for PEPA models. In L. de Alfaro and S. Gilmore, editors, *Proceedings of the first joint PAPM-PROBMIV Workshop)*, pages 120–135, Aachen, Germany, September 2001. Springer-Verlag Heidelberg.

[76] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages and computation*. Addison-Welsey, 1979.

[77] J. R. Jackson. Networks of waiting lines. *Operations Research*, 5 :518–521, 1957.

[78] J. R. Jackson. Jobshop-like queueing systems. *Management Science*, 10 :131–142, 1963.

[79] K. Jensen. Colored Petri nets. In *High-Level Petri Nets : Theory and Application*, Springer-Verlag Heidelberg, 1991.

[80] S. Stiham Jr. and R. Weber. A survey of Markov decision models for control of networks of queues. *Queueing Systems*, 13(1-3) :291–314, 1993.

[81] R. Jungblut-Hessel, B. Plateau, W. J. Stewart, and B. Ycart. Fast simulation for road traffic network. *RAIRO - Operations Research - Recherche Opérationnelle*, 35(2) :229–250, 2001.

[82] T. Kam, T. Villa, R. K. Bryaton, and A. Sangiovanni-Vincentelli. Multi-valued decision diagrams : theory and applications. *Multiplie-Valued Logic*, 4(1-2) :9–62, 1998.

[83] P. Kemper. Numerical Analysis of Superposed GSPNs. *IEEE Transactions on Software Engineering*, 22(9) :615–628, 1996.

[84] P. Kemper. Reachability Analysis Based on Structured Representations. In *Proceedings of the* 17^{th} *International Conference on Application and Theory of Petri Nets*, volume 1091 of *Lecture Notes in Computer Science*, pages 269–288, Osaka, Japan, June 1996. Springer-Verlag.

[85] L. Kleinrock. *Queueing Systems*. John Wiley & Sons, New York, 1975.

[86] P. Lascaux and R. Théodor. *Analyse numérique matricielle appliquée à l'art de l'ingénieur (vol. 1 & 2)*. Masson, Paris, 1994.

[87] R. J. Lechner. Transformations among switching function canonical forms. *IEEE Transactions on Eletronic Computers*, EC-12(2) :129–130, 1963.

[88] J. D. C. Little. A proof of the queueing formula $L = \lambda W$. *Operating Research*, 9 :383–387, 1961.

[89] W. A. Massey. Open Networks of Queues : Their Algebraic Structure and Estimating Their Transient Behavior. *Advances in Applied Probability*, 16(1) :176–201, 1984.

[90] K. L. McMillan. *Symbolic model checking : An approach to the state explosion problem*. PhD thesis, Carnegie Mellon University, Pittsburgh, USA, 1992.

[91] A. S. Miner. Implicit GSPN reachability set generation using decision diagrams. *Performance Evaluation*, 56(1-4) :145–165, 2004.

[92] A. S. Miner. Saturation for a General Class of Models. *IEEE Transactions on Software Engineering*, 32(8) :559–570, 2006.

[93] A. S. Miner and S. Cheng. Improving efficiency of implicit Markov chain state classification. In *Proceedings of the* 1^{st} *International Conference on the Quantitative Evaluation of Systems (QEST'04)*, pages 262–271, Washington, DC, USA, 2004. IEEE Computer Society.

[94] A. S. Miner and G. Ciardo. Efficient Reachability Set Generation and Storage Using Decision Diagrams. In *Proceedings of the* 20^{th} *International Conference on Applications and Theory of Petri Nets*, volume 1639 of *Lecture Notes in Computer Science*, pages 6–25, Williamsburg, VA, USA, June 1999. Springer-Verlag Heidelberg.

[95] A. S. Miner, G. Ciardo, and S. Donatelli. Using the exact state space of a Markov model to compute approximate stationary measures. In *Proceedings of the 2000 ACM SIGMETRICS Conference on Measurements and Modeling of Computer Systems*, pages 207–216, Santa Clara, California, USA, June 2000. ACM Press.

[96] L. Mokdad, J. Ben-Othman, and A. Gueroui. Quality of Service of a Rerouting Algorithm Using Stochastic Automata Networks. In *Proceedings of the* 6^{th} *IEEE Symposium on Computers and Communications*, pages 338–343, Hammamet, Tunisia, July 2001. IEEE Computer Society.

[97] M. K. Molloy. Discrete Time Stochastic Petri Nets. *IEEE Transactions on Software Engineering*, 11(4) :417–423, 1985.

[98] P. Moreaux. *Struturation des chaînes de Markov des réseaux de Petri stochastiques - décomposition tensorielle et agrégation*. PhD thesis, Université Paris Dauphine, Paris, France, 1996.

[99] M. F. Neuts. *Matrix geometric solutions in stochastic models, an algorithmic approach*. John Hopkins University Press, Baltimore, 1981.

[100] M. F. Neuts. *Structured stochastic matrices of M/G/1 type and their applications*. Marcel Dekker, New York, NY, 1989.

[101] V. Nguyen, R. Mathias, and G. D. Smith. A stochastic automata network descriptor for Markov chain models of instantaneously coupled intracellular Ca^{2+} channels. *Bulletin of Mathematical Biology*, 67(3) :393–432, 2005.

[102] E. Pastor and J. Cortadella. Efficient Encoding Schemes for Symbolic Analysis of Petri Nets. In *Proceedings of the Design, Automation and Test in Europe (DATE)*, pages 790–795, Paris, France, February 1998. IEEE Computer Society.

[103] E. Pastor and J. Cortadella. Structural Methods Applied to the Symbolic Analysis of Petri Nets. In *Proceedings of IEEE/ACM International Workshop on Logic Synthesis*, June 1998.

[104] E. Pastor, J. Cortadella, and M. A. Pena. Structural Methods to Improve the Symbolic Analysis of Petri Nets. In *Proceedings of the* 20^{th} *International Conference on Applications and Theory of Petri Nets*, volume 1639 of *Lecture Notes in Computer Science*, pages 26–45, Williamsburg, VA, USA, June 1999. Springer-Verlag Heidelberg.

[105] E. Pastor, O. Roig, J. Cortadella, and R. M. Badia. Petri net Analysis Using Boolean Manipulation. In *15th International Conference on Application and Theory of Petri Nets*, volume 815 of *Lecture Notes in Computer Science*, pages 416–435, Zaragosa, Spain, 1994. Springer-Verlag Heidelberg.

[106] PEPS. Performance Evaluation of Parallel Systems. http ://www-id.imag.fr/Logiciels/peps/.

[107] J. L. Peterson. Petri Nets. *ACM Computing Surveys*, 9(3) :223–252, 1977.

[108] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn, 1962.

[109] C. A. Petri. General net theory. In *Computing System Design : Proc. of the Joint IBM University of Newcastle upon Tyne Seminar*, pages 131–169. University of Newcastle upon Tyne, 1977.

[110] B. Plateau. *De l'Evaluation du Parallélisme et de la Synchronisation*. PhD thesis, Paris-Sud, Orsay, 1984.

[111] B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proceedings of the 1985 ACM SIGMETRICS conference on Measurements and Modeling of Computer Systems*, pages 147–154, Austin, Texas, USA, 1985. ACM Press.

[112] B. Plateau and K. Atif. Stochastic Automata Networks for modelling parallel systems. *IEEE Transactions on Software Engineering*, 17(10) :1093–1108, 1991.

[113] B. Plateau and J. M. Fourneau. A methodology for solving Markov models of parallel systems. *Journal of Parallel and Distributed Computing*, 12 :370–387, 1991.

[114] B. Plateau and W. J. Stewart. Stochastic Automata Networks : product forms and iterative solutions. Technical Report 2939, INRIA, Grenoble, 1996. ftp ://ftp.inria.fr/INRIA/publication/RR/RR-2939.ps.gz.

[115] F. Quessette. *De nouvelles méthodes de résolution pour l'analyse quantitative des systèmes parallèles et des protocoles*. PhD thesis, Paris-Sud, Orsay, 1994.

[116] M. Reiser and S. S. Lavenberg. Mean-value analysis of closed multichain queueing networks. *Journal of the ACM*, 27(2) :313–322, 1980.

[117] W. Reisig. *Petri nets : an introduction*. Springer-Verlag Heidelberg, 1985.

[118] G. Richter. Clocks and their use for time modelling. In *Information Systems : Theoretical and Formal Aspects*, pages 49–66, North Holland, Amsterdam, 1985.

[119] T. G. Robertazzi. *Computer networks and systems : queueing theory and performance evaluation*. New York : Springer-Verlag, 1990.

[120] S. M. Ross. *Simulation*. 2nd Edition, Academic Press, 1997.

[121] O. H. Roux, D. Delfieu, and P. Molinaro. Discrete time approach of time Petri nets for real-time systems analysis. In *Proceedings of the* 8^{th} *IEEE International Conference on Emerging Technologies and Factory Automation*, volume 2, pages 197–204, Nice, France, 2001. IEEE Computer Society Press.

[122] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Boston : PWS Publishing Company, 1996.

[123] A. Sales and B. Plateau. Reachable state space generation for structured models which use functional transitions. In *Proceedings of the* 6^{th} *International Conference on the Quantitative Evaluation of Systems (QEST'09)*, pages 269–278, Budapest, Hungary, September 2009. IEEE Computer Society.

[124] I. Sbeity. *Évaluation de Performance et Conception de Logiciel*. PhD thesis, Institut National Polytechnique de Grenoble, France, 2006.

[125] R. Stansifer and D. Marinescu. Petri net models of concurrent Ada programs. *Microelectronics and reliability*, 31(4) :577–594, 1991.

[126] W. J. Stewart. *Introduction to the numerical solution of Markov chains*. Princeton University Press, 1994.

[127] M. Tazza. *Análise Quantitativa de Sistemas*. Escola Brasileira-Argentina de Informática, 1988.

[128] K. S. Trivedi. *Probability & statistics with reliability, queuing, and computer science applications*. Englewood Cliffs : Prentice-Hall, 1982.

[129] E. Uysal and T. Dayar. Iterative methods based on splitting for stochastic automata networks. *European Journal of Operational Research*, 110(1) :166–186, 1998.

[130] V. Vèque and J. Ben-Othman. MRAP : a multiservices resource allocation policy for wireless ATM network. *Computer Networks and ISDN Systems*, 29(17-18) :2187–2200, 1998.

[131] J. M. Vincent. Perfect Simulation of Queueing Networks with Blocking and Rejection. In *Proceedings of the IEEE/IPSJ International Symposium on Applications and the Internet Workshops (SAINT 2005 Workshops)*, pages 268–271, Trento, Italy, February 2005. IEEE Computer Society.

[132] J. B. Waldner. *Cim : Principles of Computer-Integrated Manufacturing*. John Wiley & Sons, 1992.

[133] M. Wan and G. Ciardo. Symbolic State-Space Generation of Asynchronous Systems Using Extensible Decision Diagrams. In *Proceedings of the* 35^{th} *Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'05)*, volume 5404 of *Lecture Notes in Computer Science*, pages 582–594, Spindleruv Mlýn, Czech Republic, 2009. Springer.

[134] B. Ycart. *Modèles et Algorithmes Markoviens*. Springer-Verlag, 2002.

[135] T. Yoneda, H. Hatori, A. Takahara, and S.I. Minato. BDDs vs. Zero-Suppressed BDDs : for CTL Symbolic Model Checking of Petri Nets. In *Formal Methods in Computer-Aided Design*, volume 1166 of *Lecture Notes in Computer Science*, pages 435–449. Springer-Verlag, 1996.

[136] A. Zimmermann, J. Freiheit, and G. Hommel. Discrete Time Stochastic Petri Nets for the Modeling and Evaluation of Real-Time Systems. In *Proceedings of the* 15^{th} *International Parallel and Distributed Processing Symposium (IPDPS'01)*, pages 1069–1074, Washington, DC, USA, 2001. IEEE Computer Society Press.

Annexe A

Algèbre Tensorielle

Dans cet annexe, on présente l'*algèbre tensorielle classique* [42, 43, 7] (CTA - *Classical Tensor Algebra*) ainsi que l'*algèbre tensorielle généralisée* [54, 4, 110] (GTA - *Generalized Tensor Algebra*). Dans la section A.1, on présente l'algèbre tensorielle classique et ses propriétés et l'algèbre tensorielle généralisée et ses propriétés sont présentées dans la section A.2.

A.1 Algèbre Tensorielle Classique (CTA)

L'algèbre tensorielle classique est définie par deux opérateurs matriciels :

- les produits tensoriels (aussi appelés produits de Kronecker) ; et
- les sommes tensorielles.

Les définitions traditionnelles des ensembles de nombres naturels et réels, ainsi que des intervalles contenus dans ces ensembles sont adoptées. Cependant, il est utile de rappeler la définition pour trois cas précis :

✍ **Notation**

$[a..b]$	le sous-ensemble de $\mathbb{N}$ contenant tout les valeurs de a jusqu'à b (ces valeurs incluses) ;
$[a, b]$	le sous-ensemble de $\mathbb{R}$ contenant toutes les valeurs de a jusqu'à b (ces valeurs incluses) ;
$]a, b]$	le sous-ensemble de $\mathbb{R}$ contenant toutes les valeurs de a jusqu'à b (a exclue, b incluse).

A.1.1 Produit Tensoriel

Le produit tensoriel de deux matrices A et B, de dimensions $(\alpha_1 \times \alpha_2)$ et $(\beta_1 \times \beta_2)$ respectivement, est une matrice de dimensions $(\alpha_1\beta_1 \times \alpha_2\beta_2)$. Cette matrice peut être vue comme une matrice avec

$\alpha_1 \times \alpha_2$ blocs, chacun avec dimension $\beta_1 \times \beta_2$. La définition de chacun des éléments de la matrice résultante est faite en déterminant à quel bloc l'élément appartient et sa position interne dans le bloc.

✍ **Notation**

A	la matrice nommée A ;
$A \otimes B$	le produit tensoriel des matrices A et B ;
$A \times B$	le produit (traditionnel) des matrices A et B ;
a_{ij}	l'élément dans la ligne i et la colonne j de la matrice A ;
$a_{[ik],[jl]}$	l'élément dans la ligne k du i-ème bloc horizontal et la colonne l du j-ème bloc vertical de la matrice A.

Par exemple, prenons deux matrices A et B :

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \end{pmatrix}$$

Le *produit tensoriel* défini par $C = A \otimes B$ est égal à :

$$C = \begin{pmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{pmatrix}$$

$$C = \left(\begin{array}{cccc|cccc} a_{11}b_{11} & a_{11}b_{12} & a_{11}b_{13} & a_{11}b_{14} & a_{12}b_{11} & a_{12}b_{12} & a_{12}b_{13} & a_{12}b_{14} \\ a_{11}b_{21} & a_{11}b_{22} & a_{11}b_{23} & a_{11}b_{24} & a_{12}b_{21} & a_{12}b_{22} & a_{12}b_{23} & a_{12}b_{24} \\ a_{11}b_{31} & a_{11}b_{32} & a_{11}b_{33} & a_{11}b_{34} & a_{12}b_{31} & a_{12}b_{32} & a_{12}b_{33} & a_{12}b_{34} \\ \hline a_{21}b_{11} & a_{21}b_{12} & a_{21}b_{13} & a_{21}b_{14} & a_{22}b_{11} & a_{22}b_{12} & a_{22}b_{13} & a_{22}b_{14} \\ a_{21}b_{21} & a_{21}b_{22} & a_{21}b_{23} & a_{21}b_{24} & a_{22}b_{21} & a_{22}b_{22} & a_{22}b_{23} & a_{22}b_{24} \\ a_{21}b_{31} & a_{21}b_{32} & a_{21}b_{33} & a_{21}b_{34} & a_{22}b_{31} & a_{22}b_{32} & a_{22}b_{33} & a_{22}b_{34} \end{array}\right)$$

Dans cet exemple, l'élément $c_{47}(= a_{22}b_{13})$ est dans le bloc $(2, 2)$ et sa position interne dans ce bloc est $(1, 3)$. Le produit tensoriel $C = A \otimes B$ est défini algébriquement par l'affectation de la valeur $a_{ij}b_{kl}$ à l'élément dans la position (k, l) du bloc (i, j), *i.e.* :

$$c_{[ik],[jl]} = a_{ij}b_{kl} \quad \text{avec } i \in [1..\alpha_1],\ j \in [1..\alpha_2],\ k \in [1..\beta_1] \text{ et } l \in [1..\beta_2] \tag{A.1}$$

Cette représentation des éléments d'une matrice correspondant à un produit tensoriel induit un ordre sur les éléments $c_{[ik],[jl]}$ qui est l'ordre lexicographique sur les doublets d'indice.

A.1.1.1 Facteurs Normaux

Un cas spécifique de produit tensoriel est le produit tensoriel d'une matrice carrée par une matrice identité. Appelons ces produits des *facteurs normaux*.

✍ **Notation**

I_n la matrice identité de dimension n.

Avec une matrice carrée A et une matrice identité I_n deux facteurs normaux sont possibles : $(A \otimes I_n)$ et $(I_n \otimes A)$.

Reprenons la matrice A de l'exemple précédent et une matrice identité de taille 3 :

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Le facteur normal $A \otimes I_3$ est égal à

$$\left(\begin{array}{ccc|ccc} a_{11} & 0 & 0 & a_{12} & 0 & 0 \\ 0 & a_{11} & 0 & 0 & a_{12} & 0 \\ 0 & 0 & a_{11} & 0 & 0 & a_{12} \\ \hline a_{21} & 0 & 0 & a_{22} & 0 & 0 \\ 0 & a_{21} & 0 & 0 & a_{22} & 0 \\ 0 & 0 & a_{21} & 0 & 0 & a_{22} \end{array}\right)$$

Le facteur normal $I_3 \otimes A$ est égal à

$$\left(\begin{array}{cc|cc|cc} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & a_{11} & a_{12} & 0 & 0 \\ 0 & 0 & a_{21} & a_{22} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & a_{11} & a_{12} \\ 0 & 0 & 0 & 0 & a_{21} & a_{22} \end{array}\right)$$

Un cas encore plus particulier de produit tensoriel est le produit de deux matrices identités. Ce produit est une matrice identité dont la dimension est égale au produit des dimensions de chacune des matrices, *i.e.* :

$$I_n \otimes I_m = I_{nm}$$

A.1.2 Somme Tensorielle

✍ **Notation**

$A \oplus B$	la somme tensorielle des matrices carrées A et B ;
$A + B$	la somme (traditionnelle) des matrices A et B ;
n_A	la dimension (nombre de lignes et de colonnes) de la matrice carrée A.

La somme tensorielle de deux matrices carrées[1] A et B est définie comme la somme de facteurs normaux de chacune des matrices selon la formule :

$$A \oplus B = (A \otimes I_{n_B}) + (I_{n_A} \otimes B) \tag{A.2}$$

Prenons par exemple les matrices A et B définies par :

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix}$$

La somme tensorielle définie par $C = A \oplus B$ est égal à :

$$C = \left(\begin{array}{ccc|ccc} a_{11} & 0 & 0 & a_{12} & 0 & 0 \\ 0 & a_{11} & 0 & 0 & a_{12} & 0 \\ 0 & 0 & a_{11} & 0 & 0 & a_{12} \\ \hline a_{21} & 0 & 0 & a_{22} & 0 & 0 \\ 0 & a_{21} & 0 & 0 & a_{22} & 0 \\ 0 & 0 & a_{21} & 0 & 0 & a_{22} \end{array}\right) + \left(\begin{array}{ccc|ccc} b_{11} & b_{12} & b_{13} & 0 & 0 & 0 \\ b_{21} & b_{22} & b_{23} & 0 & 0 & 0 \\ b_{31} & b_{32} & b_{33} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & b_{11} & b_{12} & b_{13} \\ 0 & 0 & 0 & b_{21} & b_{22} & b_{23} \\ 0 & 0 & 0 & b_{31} & b_{32} & b_{33} \end{array}\right)$$

$$C = \left(\begin{array}{ccc|ccc} a_{11}+b_{11} & b_{12} & b_{13} & a_{12} & 0 & 0 \\ b_{21} & a_{11}+b_{22} & b_{23} & 0 & a_{12} & 0 \\ b_{31} & b_{32} & a_{11}+b_{33} & 0 & 0 & a_{12} \\ \hline a_{21} & 0 & 0 & a_{22}+b_{11} & b_{12} & b_{13} \\ 0 & a_{21} & 0 & b_{21} & a_{22}+b_{22} & b_{23} \\ 0 & 0 & a_{21} & b_{31} & b_{32} & a_{22}+b_{33} \end{array}\right)$$

La somme tensorielle $C = A \oplus B$ est définie algébriquement par l'affectation de la valeur $a_{ij}\delta_{kl} + \delta_{ij}b_{kl}$ à l'élément dans la position (k, l) du bloc (i, j), *i.e.* :

$$\begin{aligned} c_{[ik],[jl]} &= a_{ij}\delta_{kl} + \delta_{ij}b_{kl} \\ &\qquad \text{avec } i, j \in [1..n_A] \text{ et } k, l \in [1..n_B], \end{aligned} \tag{A.3}$$

où δ_{ij} (delta de Kronecker) est l'élément dans la ligne i et la colonne j d'une matrice identité défini par :

$$\delta_{ij} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

[1] Bien que le produit tensoriel soit défini pour des matrices non carrées, la somme tensorielle est définie exclusivement pour les matrices carrées.

L'opérateur produit tensoriel ($\otimes$) est prioritaire sur l'opérateur somme tensorielle ($\oplus$) et les deux opérateurs tensoriels sont prioritaires sur les opérateurs traditionnels de multiplication et addition ($\times$ et $+$).

A.1.3 Propriétés

Ces propriétés ont été déjà prouvés dans le cadre de la définition des produits de Kronecker [7]. Pour ces propriétés aucune démonstration n'est présentée dans cette thèse. Le lecteur peut trouver dans les travaux de Davio, Deschamps et Thayse [42, 43] et dans le livre de Bellman [7] la preuve des propriétés suivantes :

- Associativité :
  $A \otimes (B \otimes C) = (A \otimes B) \otimes C$
  $A \oplus (B \oplus C) = (A \oplus B) \oplus C$

- Distributivité sur la somme :
  $(A + B) \otimes (C + D) = (A \otimes C) + (B \otimes C) + (A \otimes D) + (B \otimes D)$

- Compatibilité avec la multiplication :
  $(A \times B) \otimes (C \times D) = (A \otimes C) \times (B \otimes D)$

- Compatibilité avec la transposition des matrices :
  $(A \otimes B)^T = A^T \otimes B^T$

- Compatibilité avec l'inversion des matrices (si A et B sont des matrices carrées et inversibles) :
  $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$

Comme Davio [42] l'a déjà remarqué pour les premières applications des produits de Kronecker [87], la propriété de la *compatibilité avec la multiplication* est fondamentale, car elle établit la relation entre les produits traditionnels de matrices et les produits tensoriels. Cette propriété est à la base des démonstrations des deux propriétés proposées dans les travaux directement liées au formalisme SAN [110, 113, 112, 55] :

- Décomposition en Facteurs Normaux :
  $A \otimes B = (A \otimes I_{n_B}) \times (I_{n_A} \otimes B)$

- Distributivité sur la multiplication par l'identité :
  $(A \times B) \otimes I_n = (A \otimes I_n) \times (B \otimes I_n)$
  $I_n \otimes (A \times B) = (I_n \otimes A) \times (I_n \otimes B)$[2]

[2]Encore que cet propriété puisse être déduite par la propriété *Compatibilité avec la multiplication*, elle a été définie par Fernandes, Plateau et Stewart [55].

A.2 Algèbre Tensorielle Généralisée (GTA)

L'algèbre tensorielle généralisée est définie comme une extension de l'algèbre tensorielle classique. La différence fondamentale apportée par l'algèbre tensorielle généralisée est l'introduction du concept d'*éléments fonctionnels*. Une matrice peut désormais être composée d'éléments constants (appartenant à $\mathbb{R}$) ou d'éléments fonctionnels. Un *élément fonctionnel* est une fonction à valeur dans $\mathbb{R}$ selon un jeu de paramètres composé par les indices de ligne d'une ou de plusieurs matrices.

Un élément fonctionnel b qui possède l'indice de ligne de la matrice A dans son jeu de paramètres est dit *dépendant* de la matrice A. Par abus de langage, on appelle *paramètres* d'un élément fonctionnel toutes les matrices dont l'élément est dépendant. Une matrice contenant au moins un élément fonctionnel dépendant de la matrice A est dite *dépendante* de la matrice A. Les *paramètres* d'une matrice sont l'union des paramètres de tous ses éléments.

Dans le contexte du formalisme SAN, les matrices sont utilisées pour représenter des transitions entre les états d'un automate (l'élément i, j représente la transition de l'état a_i vers l'état a_j). Cette interprétation, particulière au formalisme SAN, nous permet de faire une correspondance bi-univoque entre l'état d'un automate dans un modèle SAN et l'indice de ligne d'une matrice. Notons que ceci ne représente en aucun cas une restriction d'utilisation de l'algèbre tensorielle généralisée exclusivement au formalisme SAN. Les définitions et propriétés présentées sont indépendantes de l'interprétation donnée à l'indice de ligne. Cependant, ceci est fait dans le cadre du formalisme SAN pour faciliter la compréhension sémantique des éléments fonctionnels.

À l'instar de l'algèbre tensorielle classique, l'algèbre tensorielle généralisée est définie par les deux opérateurs matriciels :

- les produits tensoriels généralisés ($\underset{g}{\otimes}$) ; et
- les sommes tensorielles généralisées ($\underset{g}{\oplus}$).

Les notations définies dans la section précédente sont toujours valides pour les matrices n'ayant pas d'éléments fonctionnels, appelées *matrices constantes*. Les matrices avec des éléments fonctionnels, appelées *matrices fonctionnelles*, seront décrites par la notation suivante :

✍ **Notation**

a_k	l'indice de la ligne k de la matrice A, utilisé dans le contexte du formalisme SAN aussi comme l'état de l'automate $\mathcal{A}$ auquel la matrice A correspond ;
$A(\mathcal{B}, \mathcal{C})$	la matrice fonctionnelle A qui possède comme paramètres les matrices B et C ;
$a_{ij}(\mathcal{B}, \mathcal{C})$	l'élément fonctionnel i, j de la matrice $A(\mathcal{B}, \mathcal{C})$;
$A(b_k, \mathcal{C})$	la matrice fonctionnelle $A(\mathcal{B}, \mathcal{C})$ où l'indice de ligne de la matrice B est déjà connu et égal à k (cette matrice peut être considéré comme dépendante de la matrice C seulement) ;
$a_{ij}(b_k, \mathcal{C})$	l'élément fonctionnel i, j de la matrice $A(b_k, \mathcal{C})$;
$A(b_k, c_l)$	la matrice fonctionnelle $A(\mathcal{B}, \mathcal{C})$ avec les éléments évalués pour les états correspondants aux lignes k et l des matrices B et C respectivement (cette matrice, ayant tous ses paramètres connus, est considérée comme constante) ;

$a_{ij}(b_k, c_l)$	l'élément constant (élément fonctionnel évalué) i, j de la matrice $A(b_k, c_l)$;
$\ell_k(A)$	la matrice avec tous les éléments mis à zéro, sauf ceux appartenant à la ligne k qui est égale à la ligne k de la matrice A ($A = \sum_{k=1}^{n_A} \ell_k(A)$) ;
$A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A})$	le produit tensoriel généralisé entre les matrices $A(\mathcal{B})$ et $B(\mathcal{A})$;
$A(\mathcal{B}) \underset{g}{\oplus} B(\mathcal{A})$	la somme tensorielle généralisée entre les matrices $A(\mathcal{B})$ et $B(\mathcal{A})$.

A.2.1 Produit Tensoriel Généralisé

Prenons par exemple deux matrices $A(\mathcal{B})$ et $B(\mathcal{A})$ définies par :

$$A(\mathcal{B}) = \begin{pmatrix} a_{11}[\mathcal{B}] & a_{12}[\mathcal{B}] \\ a_{21}[\mathcal{B}] & a_{22}[\mathcal{B}] \end{pmatrix} \qquad B(\mathcal{A}) = \begin{pmatrix} b_{11}[\mathcal{A}] & b_{12}[\mathcal{A}] & b_{13}[\mathcal{A}] \\ b_{21}[\mathcal{A}] & b_{22}[\mathcal{A}] & b_{23}[\mathcal{A}] \\ b_{31}[\mathcal{A}] & b_{32}[\mathcal{A}] & b_{33}[\mathcal{A}] \end{pmatrix}$$

Le produit tensoriel défini par $C = A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A})$ est égal à :

$$C = \left(\begin{array}{ccc|ccc} a_{11}(b_1)b_{11}(a_1) & a_{11}(b_1)b_{12}(a_1) & a_{11}(b_1)b_{13}(a_1) & a_{12}(b_1)b_{11}(a_1) & a_{12}(b_1)b_{12}(a_1) & a_{12}(b_1)b_{13}(a_1) \\ a_{11}(b_2)b_{21}(a_1) & a_{11}(b_2)b_{22}(a_1) & a_{11}(b_2)b_{23}(a_1) & a_{12}(b_2)b_{21}(a_1) & a_{12}(b_2)b_{22}(a_1) & a_{12}(b_2)b_{23}(a_1) \\ a_{11}(b_3)b_{31}(a_1) & a_{11}(b_3)b_{32}(a_1) & a_{11}(b_3)b_{33}(a_1) & a_{12}(b_3)b_{31}(a_1) & a_{12}(b_3)b_{32}(a_1) & a_{12}(b_3)b_{33}(a_1) \\ \hline a_{21}(b_1)b_{11}(a_2) & a_{21}(b_1)b_{12}(a_2) & a_{21}(b_1)b_{13}(a_2) & a_{22}(b_1)b_{11}(a_2) & a_{22}(b_1)b_{12}(a_2) & a_{22}(b_1)b_{13}(a_2) \\ a_{21}(b_2)b_{21}(a_2) & a_{21}(b_2)b_{22}(a_2) & a_{21}(b_2)b_{23}(a_2) & a_{22}(b_2)b_{21}(a_2) & a_{22}(b_2)b_{22}(a_2) & a_{22}(b_2)b_{23}(a_2) \\ a_{21}(b_3)b_{31}(a_2) & a_{21}(b_3)b_{32}(a_2) & a_{21}(b_3)b_{33}(a_2) & a_{22}(b_3)b_{31}(a_2) & a_{22}(b_3)b_{32}(a_2) & a_{22}(b_3)b_{33}(a_2) \end{array}\right)$$

La définition algébrique du produit tensoriel généralisé $C = A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A})$ est faite par l'affectation de la valeur $a_{ij}(b_k)b_{kl}(a_i)$ à l'élément $c_{[ik],[jl]}$, *i.e.* :

$$\begin{aligned} c_{[ik],[jl]} &= a_{ij}(b_k)b_{kl}(a_i) \\ &\text{avec } i, j \in [1..n_A] \text{ et } k, l \in [1..n_B] \end{aligned} \tag{A.4}$$

A.2.2 Somme Tensorielle Généralisée

La définition de la somme tensorielle généralisée est faite en utilisant des produits tensoriels généralisés sur l'équation (A.2) :

$$A \underset{g}{\oplus} B = (A \underset{g}{\otimes} I_{n_B}) + (I_{n_A} \underset{g}{\otimes} B) \tag{A.5}$$

Reprenons les matrices $A(\mathcal{B})$ et $B(\mathcal{A})$ utilisées pour décrire le produit tensoriel généralisé. La somme tensorielle défini par $C = A(\mathcal{B}) \underset{g}{\oplus} B(\mathcal{A})$ est égal à :

$$C = \left(\begin{array}{ccc|ccc} a_{11}(b_1)+b_{11}(a_1) & b_{12}(a_1) & b_{13}(a_1) & a_{12}(b_1) & 0 & 0 \\ b_{21}(a_1) & a_{11}(b_2)+b_{22}(a_1) & b_{23}(a_1) & 0 & a_{12}(b_2) & 0 \\ b_{31}(a_1) & b_{32}(a_1) & a_{11}(b_3)+b_{33}(a_1) & 0 & 0 & a_{12}(b_3) \\ \hline a_{21}(b_1) & 0 & 0 & a_{22}(b_1)+b_{11}(a_2) & b_{12}(a_2) & b_{13}(a_2) \\ 0 & a_{21}(b_2) & 0 & b_{21}(a_2) & a_{22}(b_2)+b_{22}(a_2) & b_{23}(a_2) \\ 0 & 0 & a_{21}(b_3) & b_{31}(a_2) & b_{32}(a_2) & a_{22}(b_3)+b_{33}(a_2) \end{array}\right)$$

La définition algébrique de la somme tensorielle généralisée $C = A(\mathcal{B}) \underset{g}{\oplus} B(\mathcal{A})$ est faite par l'affectation de la valeur $a_{ij}(b_k)\delta_{kl} + b_{kl}(a_i)\delta_{ij}$ à l'élément $c_{[ik],[jl]}$, *i.e.* :

$$\begin{aligned} c_{[ik],[jl]} &= a_{ij}(b_k)\delta_{kl} \ + \ b_{kl}(a_i)\delta_{ij} \\ &\text{avec } i,j \in [1..n_A] \text{ et } k,l \in [1..n_B] \end{aligned} \tag{A.6}$$

A.2.3 Propriétés

Dans ce section, on présente les propriétés de l'algèbre tensorielle généralisée définies par Fernandes, Plateau et Stewart [55] :

- Distributivité du produit tensoriel généralisé sur la somme traditionnelle de matrices :
  $[A(\mathcal{C},\mathcal{D}) + B(\mathcal{C},\mathcal{D})] \underset{g}{\otimes} [C(\mathcal{A},\mathcal{B}) + D(\mathcal{A},\mathcal{B})] = A(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B}) \ + \ A(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} D(\mathcal{A},\mathcal{B}) \ +$
  $B(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B}) \ + \ B(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} D(\mathcal{A},\mathcal{B})$
- Associativité du produit tensoriel généralisé et de la somme tensorielle généralisée :
  $[\ A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} B(\mathcal{A},\mathcal{C})\ ] \underset{g}{\otimes} C(\mathcal{A},\mathcal{B}) = A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} [\ B(\mathcal{A},\mathcal{C}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})\ ]$

  $[\ A(\mathcal{B},\mathcal{C}) \underset{g}{\oplus} B(\mathcal{A},\mathcal{C})\ ] \underset{g}{\oplus} C(\mathcal{A},\mathcal{B}) = A(\mathcal{B},\mathcal{C}) \underset{g}{\oplus} [\ B(\mathcal{A},\mathcal{C}) \underset{g}{\oplus} C(\mathcal{A},\mathcal{B})\ ]$
- Distributivité sur la multiplication par l'identité :
  $[\ A(\mathcal{C}) \times B(\mathcal{C})\ ] \underset{g}{\otimes} I_{n_C} = A(\mathcal{C}) \underset{g}{\otimes} I_{n_C} \ \times \ B(\mathcal{C}) \underset{g}{\otimes} I_{n_C}$

  $[\ I_{n_C} \underset{g}{\otimes} A(\mathcal{C})\ ] \times B(\mathcal{C}) = I_{n_C} \underset{g}{\otimes} A(\mathcal{C}) \ \times \ I_{n_C} \underset{g}{\otimes} B(\mathcal{C})$
- Décomposition en Facteurs Normaux I :
  $A \underset{g}{\otimes} B(\mathcal{A}) = I_{n_A} \underset{g}{\otimes} B(\mathcal{A}) \ \times \ A \underset{g}{\otimes} I_{n_B}$
- Décomposition en Facteurs Normaux II :
  $A(\mathcal{B}) \underset{g}{\otimes} B = A(\mathcal{B}) \underset{g}{\otimes} I_{n_B} \ \times \ I_{n_A} \underset{g}{\otimes} B$

– Pseudo-Commutativité :
$A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A}) = P_\sigma \times \ B(\mathcal{A}) \underset{g}{\otimes} A(\mathcal{B}) \ \times {P_\sigma}^T$

– Décomposition en Produits Tensoriels Classiques :
$A \underset{g}{\otimes} B(\mathcal{A}) = \sum_{k=1}^{n_A} \ell_k(A) \otimes B(a_k)$

Titre : Réseaux d'Automates Stochastiques : Génération de l'espace d'états atteignables et Multiplication vecteur-descripteur pour une sémantique en temps discret

Résumé : Cette thèse présente des méthodes et des algorithmes pour l'évaluation de performance de systèmes à grand espace d'états décrits par des formalismes de haut niveau. Parmi les différents formalismes de haut niveau normalement utilisés, nous nous sommes intéressés au formalisme des *Réseaux d'Automates Stochastiques* (SAN). Le formalisme SAN se caractérise par la modélisation de systèmes complexes, où un système est représenté par la composition de sous-systèmes (automates) qui interagissent entre eux. Cette interaction se réalise par l'occurrence des événements synchronisants ou des taux fonctionnels.
Lorsqu'on calcule l'espace d'états atteignables de systèmes complexes, le principal problème qui surgit est l'*explosion combinatoire de l'espace d'états* du modèle. Dans la première partie de cette thèse, nous proposons des méthodes pour la génération de l'espace d'états atteignables de modèles compositionnels qui utilisent des taux fonctionnels. Nous utilisons les *Diagrammes de Décision Multi-valués* (MDD) pour représenter et manipuler les espaces d'états et le formalisme SAN pour la modélisation de systèmes. Un MDD est une structure de donnée arborescente qui permet de représenter et de manipuler de façon performante un très grand espace d'états. L'avancée par rapport à l'état de l'art a été de proposer de méthodes qui prennent en compte ces fonctions qui expriment des relations entre les composants des modèles. Des études d'exemples sont présentées afin d'illustrer les apports de ces méthodes.
Dans la deuxième partie de cette thèse, nous nous sommes intéressés à la résolution d'un modèle SAN à temps discret dont la matrice de transition est représentée par une formule tensorielle (appelée *descripteur discret*). A cet effet, nous présentons l'*Algèbre Tensorielle compleXe* (ATX) adaptée à la composition parallèle des SAN à temps discret pour la représentation du descripteur et nous démontrons des propriétés qui servent de base aux méthodes itératives pour la résolution de la chaîne de Markov associée au modèle SAN. Un des avantages de représenter un modèle SAN par un descripteur est la façon compacte par laquelle on peut représenter les transitions du modèle : on remplace une description dans un espace produit par un unique produit tensoriel portant sur des facteurs qui décrivent ce qui se passe sur une seule dimension (une composante du modèle SAN). Afin de profiter de cette représentation, nous présentons une méthode de multiplication d'un vecteur de probabilité par un descripteur discret adaptée à cette algèbre. Cette méthode vise à exploiter des propriétés du produit tensoriel complexe de façon à ce que la multiplication par un opérateur sur l'espace produit soit remplacée par une suite d'opérations qui manipulent des données de la taille d'une composante (et pour toutes les composantes).

Mots clés : Évaluation de Performance, Réseaux d'Automates Stochastiques, Temps Continu et Discret, Génération de l'Espace d'États, Diagrammes de Décision, Algèbre Tensorielle, Multiplication Vecteur-Descripteur.

Title : Stochastic Automata Networks : Reachable state space generation and Vector-descriptor product for a semantic of discrete time models

Abstract : This thesis presents methods and algorithms for the performance evaluation of large state space models described by high-level formalisms. Among the various formalisms we use *Stochastic Automata Networks* (SAN) formalism. SAN formalism is dedicated to modeling very large systems by the composition of its subsystems (automata), where these automata interact with each other by synchronizing events or functional rates and probabilities.
The state space explosion of a model is a common problem when computing the reachable state space of complex systems. In the first part of this thesis, we propose reachable state space generation methods of structured models which use functional rates (general state dependant rates) and probabilities. We use *Multi-valued Decision Diagrams* (MDD) to store sets of reachable spaces and SAN formalism to describe structured models. MDD is a multilevel data structure which efficiently manipulates a extremely large state spaces. Regarding state-of-the-art generation methods, our methods allow to construct a huge reachable state space of a model which uses functional rates and probabilities. The methods are tested on some models in order to illustrate this contribution.
In the second part, we are interested in the solution of a discrete time SAN model whose transition matrix is represented by a tensor formula (called *discrete descriptor*). For this purpose, we present the *Complex Tensor Algebra* (XTA) adapted to the parallel composition of the discrete time SANs in order to represent the discrete descriptor and we prove some properties that are the basis for iterative methods for solving the Markov chain associated to the SAN model. Representing a SAN model by a descriptor is a compact way to describe the transitions among global states of the model : we replace a description in a product space by a single tensor product on factors that describe what happens on a single dimension (one automaton of the SAN model). In order to take advantage of this representation, we present a vector-descriptor product method which performs the multiplication of a probability vector and the discrete descriptor. This method aims at exploiting the properties of the complex tensor product so that the multiplication by an operator on the product space is replaced by a series of operations that manipulate data on the size of an automaton (and for all automata).

Keywords : Performance Evaluation, Stochastic Automata Networks, Continuous and Discrete Time Models, State Space Generation, Decision Diagrams, Tensor Algebra, Vector-Descriptor Product.