UNIVERSITE PARIS VI - PIERRE ET MARIE CURIE
LABORATOIRE D'INFORMATIQUE DE PARIS 6 - LIP6
SYSTEMES REPARTIS ET COOPERATIFS - SRC

Doctorat Spécialité Informatique

Exploitation des symétries partielles pour la vérification et l'évaluation de performances des systèmes finis.

Souheib Baarir

Thèse dirigée par le Professeur Serge Haddad
Soutenue le 16 Mai 2007

Jury :

M. Jean-Michel Couvreur	Professeur	Univ. d'Orléans	Rapporteur
M. François Vernadat	Professeur	INSA de Toulouse	Rapporteur
Mme. Giuliana Franceschinis	Professeur	Univ. de Piemonte Orientale	Examinatrice
M. Claude Jard	professeur	ENS Cachan	Examinateur
M. Fabrice Kordon	Professeur	Paris VI	Examinateur
M. Serge Haddad	Professeur	Paris-Dauphine	Examinateur
Mme. Claude Dutheillet	Maitre de Conférences	Paris VI	Co-encadrant
M. Jean-Michel Ilié	Maitre de Conférences	Paris V	Co-encadrant

Table des matières

[1] Modèle introduit pour la première fois dans [BHI04].

[2] Les idées développées dans ce chapitre ont fait l'objet d'un rapport technique ([BDL06]), et d'une publication dans ACSD 2007 ([BDL07]).

[3] Les idées développées dans ce chapitre ont fait l'objet de plusieurs publications [IBB+04, TMBDLK04, BIDL04, HTMK+04, TMIB06] .

[4] Les idées développées dans ce chapitre ont fait l'objet d'une publication [BDSH05].

Chapitre 1

Introduction Générale

1.1 Les systèmes concurrents

Un système concurrent à événements discrets est un système composé d'entités s'exécutant en parallèle et communiquer entre elles. Dans cette classe de systèmes, citons les systèmes informatiques, les systèmes de production, les réseaux de communication, etc.

Ces dernières décennies ont vu une évolution rapide de ces systèmes, liée essentiellement aux progrès technologiques du matériel, mais aussi à la conception des systèmes d'exploitations et des protocoles de communication qui régissent le fonctionnement et le parallélisme entre composantes des systèmes. La tendance actuelle est de bénéficier de ce parallélisme en particulier pour améliorer les temps de réponse ou encore partager l'espace mémoire. En contrepartie, les tâches de synchronisation entre composants s'alourdissent et deviennent très complexes à maîtriser par des moyens humains.

La mise en œuvre de tels systèmes est contrôlée par des méthodes classiques en génie logiciel, partant de la spécification en fonction de l'analyse des besoins jusqu'à la validation du produit final. Cependant, une caractéristique importante de ces systèmes est leur propension à avoir un grand nombre de comportements possibles. Cela est essentiellement dû à l'exécution concurrente des différents éléments et aux multiples interactions qu'un élément du système peut avoir avec son environnement. Ce fait rend leur conception, leur réalisation et leur mise en œuvre difficile.

Dans le cycle de développement d'un système concurrent, ces caractéristiques particulières rendent les phases de *vérification* et *d'évaluation de performances* essentielles et cruciales, car elles permettent de contrôler que la spécification du

système satisfait les propriétés attendues. Il en résulte que tout problème lié au comportement qualitatif (sûreté, équité, etc.), comme au comportement quantitatif (taux de perte, vitesse moyenne de transmission, etc.) est détecté et corrigé très tôt dans le cycle de vie.

Pour raisonner, sans risque d'ambiguïté, sur une spécification, il faut que cette dernière soit décrite de manière formelle. Le formalisme choisi doit répondre à deux objectifs contradictoires. D'une part, il doit être suffisamment expressif afin de décrire toutes les parties délicates et sujettes à des comportements concurrents complexes. D'autre part, cette expressivité est nécessairement limitée afin d'obtenir des algorithmes efficaces de vérification et d'évaluation de performances.

De nombreux formalismes dédiés à la concurrence ont été proposés, par exemple, les automates communicants [BZ83, GR85], les algèbres de processus CCS [BHR84a], CSP [Mil89a], Estelle [HE93], LOTOS [GS90, BSS95a], SDL [Bro91], Unity [ZE99], Promela [BMvE98], etc. Par ailleurs, des formalismes tels que les réseaux de files d'attente ont été développés pour l'évaluation de performances. Ces derniers disposent de techniques d'analyse éprouvées (exactes ou approchées) [BCMP75, Mar79, Bra85, BM89, BMT89, Dal90].

Parmi ces modèles, les *Réseaux de Petri (RdP)* [Pet62, Mur89] occupent une place de choix si l'on considère la diversité des techniques automatisées qui leur sont associées. Dans leur forme standard, ils offrent une structure très simple et constituent le support de nombreuses études théoriques des systèmes et de leurs comportements. De plus, pour mener une étude quantitative, plusieurs extensions temporelles des RdP ont été proposées, en particulier le modèle des RdP stochastiques [Mol81, FN85].

Cependant, le formalisme des RdPs favorise l'étude du contrôle des systèmes au détriment de la structuration des données, car il n'est pas facile d'intégrer dans les RdPs standard l'organisation des données transportées, échangées ou modifiées. Afin de palier à ce problème, différentes versions de plus haut niveau ont été proposées. Parmi les extensions les plus connues, on peut citer les réseaux colorés [Jen82], les réseaux à Prédicats/Transitions [GL81], les réseaux algébriques [Rei91], ou encore les réseaux bien formés [CDFH93].

Ces modèles ont pour but de permettre l'analyse des propriétés des systèmes avant leur réalisation.

1.2 Vérification qualitative

Elle consiste à définir les propriétés structurelles et comportementales du système, telles que l'absence de blocage (vivacité), les invariants du système, le comportement borné, etc. Par exemple, dans un réseau de communication, si une machine *A* attend un message d'une machine *B* pour pouvoir poursuivre son processus de communication et qu'en même temps la machine *B* attend un message de la machine *A*, le système est dans un état d'interblocage (en anglais : deadlock) et ne peut plus évoluer. L'étude qualitative nous renseignera sur l'éventualité d'un tel état.

Selon la nature des propriétés à vérifier, les méthodes de vérification peuvent être classées en trois grandes catégories :

– Les méthodes qui s'apparentent aux démonstrateurs de théorèmes. Ce sont des processus analytiques opérant par raffinement des propriétés. Ces outils aident à concevoir les pas intermediaries de la preuve de la propriété. Par contre, le processus de vérification n'est que semi-décidable [AMO99, BSS95b].

– Les méthodes liées à l'analyse de la structure d'un système. Elles sont généralement basées sur des méthodes algébriques, calcul de flots pour la recherche d'invariant [Cou90], preuves axiomatiques basées sur les algèbres de processus [BHR84b, Mil89b], opérations de réduction sur les réseaux de Petri spécifiant le système [Ber86, Ber87, Had87]. Ces méthodes ont l'avantage de ne pas explorer l'espace d'états, cependant leur exploitation concerne des propriétés d'ordre général, comme l'accessibilité d'un état, l'interblocage, etc. Par ailleurs, l'interprétation des résultats obtenus est souvent difficile (réservée aux experts).

– Les méthodes dites exploratoires. Elles sont basées sur l'exploration d'une partie ou de la totalité des comportements possibles d'un système. Ces comportements sont représentés par un *graphe d'accessibilité* dont chaque nœud représente un état accessible par le système et les arcs les différents changements qu'il peut réaliser. Pendant longtemps, les seules techniques proposées ont été justement la construction de ce graphe et la simulation, laissant à l'ingénieur le soin d'opérer la vérification lui-même. Ces techniques ont depuis, été largement étendues à la vérification automatique des propriétés [CES86, ECL93, Eme95]. On distingue les propriétés de vivacité attestant des fonctionnements souhaités, des propriétés de sûreté précisant les fonctionnements qui sont prohibés. Une contrainte (mineure pour les systèmes que nous considérons) à prendre en compte dans ces approches est que le graphe d'accessibilité (le nombre d'états) doit être fini. D'autre part, ce graphe ne peut être construit que lorsque l'état initial du système est fixé et les propriétés validées ne le seront que pour cet

état initial.

Le problème majeur de ce type de méthodes est la taille souvent excessive du graphe d'accessibilité. En effet, elle peut être exponentielle par rapport à la taille de la description du système. Cette *explosion combinatoire* du nombre d'états est due essentiellement à l'exécution concurrente des différents objets qui composent le système et à leur nécessaire synchronisation.

Pour juguler les effets néfastes de l'explosion combinatoire, différentes solutions ont été proposées et elles ont pour objectif, soit de réduire la représentation du graphe, soit de substituer au graphe un graphe plus petit. Les critères caractérisant leur qualité sont : (1) leur pouvoir de réduction ; (2) les propriétés préservées ; (3) leur capacité à être utilisées conjointement avec d'autres méthodes et (4) l'efficacité des algorithmes mis en œuvre. Deux catégories de méthodes sont à distinguer : les méthodes pour *gérer* et celles pour *combattre* l'explosion.

Gérer. Cela consiste à minimiser l'impact de l'explosion combinatoire en choisissant une représentation des états et des comportements particulièrement concise. Les diagrammes de décision binaires ordonnés (BDD : Binary Decision Diagram) sont une structure de données permettant la représentation de fonctions booléennes et sur laquelle les opérations booléennes classiques sont réalisées efficacement. En codant d'une part l'ensemble des états accessibles et d'autre part la relation de transition sous forme de BDD, il est alors possible de définir des outils de vérification pour un grand nombre de propriétés [BCM+90, CE81, McM93]. Plusieurs structures basées sur le même principe ont été définies, telles que les diagrammes de décision de données (DDD : Data Decision Diagram) [CEPA+02, TMIP04] ou encore les diagrammes de décision hiérarchiques (SDD : Set Decision Diagram) [CTM05].

Combattre. Ces méthodes s'attaquent aux racines du problème.

– Les méthodes basées sur l'exploitation de *la modularité* des systèmes [Val93, NM94, CP95]. Dans ces approches, on se base sur l'observation que certains systèmes sont constitués de modules. Dans ce cas, la vérification de certaines propriétés (e.g., la vivacité) du système global, peut être réduite à la vérification de propriétés locales à chaque module. Ainsi, au lieu de construire l'espace d'états du système global, on construit les graphes des modules (beaucoup moins gourmands en temps et en espace) qui seront compressés, puis combinés, pour obtenir une représentation du comportement du système originel.

– Lorsque le parallélisme est représenté par entrelacement des actions, les méthodes *d'ordre partiel* visent à supprimer les entrelacement superflus dans le graphe de comportement. En effet, les différents entrelacements possibles d'événements concurrents introduisent un grand nombre de séquences composées des mêmes transitions, partant et arrivant aux mêmes états et où seul l'ordre d'apparition diffère. Ces séquences "similaires" passent par un nombre important d'états différents. L'idée est de réduire le graphe de comportement grâce à l'équivalence de séquences. Il existe essentiellement deux types d'approches pour ces méthodes, donnant des relations de réduction différentes :
  – *Élimination de l'entrelacement* : cette approche cherche à obtenir un sous-graphe du graphe de comportement comportant le moins de séquences équivalentes possibles [WG93].
  – *Pas couvrant* : cette approche consiste à agglomérer des ensembles de transitions indépendantes. Ces ensembles d'actions sont des *pas*. Le graphe obtenu est un graphe de pas couvrants [VAM96].

– Le troisième type de techniques se base sur l'observation qu'un système concurrent est souvent constitué d'éléments ayant des comportements *fortement symétriques (globalement symétriques), i.e.*, identiques à une permutation près [Had88, **?**, Dut91]. La factorisation de la représentation de ces comportements similaires permet la construction de graphes réduits, qui peuvent être utilisés pour la vérification de propriétés dites, elles aussi, *fortement symétriques* [ES96, ES97, IA97].

  Cependant, la pratique montre qu'un grand nombre des systèmes étudiés sont constitués d'éléments *faiblement symétriques, i.e.*, leurs comportements alternent des comportements similaires sur certains éléments de séquences et différents (asymétriques) sur d'autres. Pour ce type de systèmes, dits *partiellement symétriques*, l'application des techniques basées sur les symétries globales sont inefficaces. En effet, dans ces approches, les asymétries, si locales soient elles seront considérées comme globales et par conséquent, leur effet va se répercuter sur toute la construction, engendrant ainsi un taux de réduction minimum.

  Plusieurs travaux ont étudié ce problème, soit en ramenant l'étude d'un système partiellement symétrique à celle d'un système symétrique [ET99], soit en développant de nouvelles approches profitant de la localité des asymétries [HITZ95, AHl98, Cap00, HIA00].

  Puisque nos travaux s'intègrent dans cette dernière catégorie de techniques, nous y reviendrons pour une étude détaillée dans le chapitre 3.

1.3 Évaluation de performances

La preuve des propriétés qualitatives d'un système n'est généralement pas suffisante pour assurer son bon fonctionnement. Par exemple, dans le domaine des réseaux de communication, il est indispensable d'évaluer le taux de perte de messages ou le temps moyen de réponse. On veut aussi tester l'effet de la variation de certains paramètres sur le fonctionnement du système (dimentionnement des ressources). Ce type d'études quantitatives est appelé évaluation de performances des systèmes.

L'un des formalismes utilisés pour la modélisation et l'évaluation de performances fut les réseaux de files d'attente [Kle75]. L'avantage de ce type de modèles est qu'ils possèdent des techniques d'analyse éprouvées, exactes [Jac63, BCMP75] ou approchées [Mar79, Bra85, Dal90], permettant d'obtenir la solution de certaines classes de systèmes, éventuellement infinis, sans la construction de l'ensemble des états accessibles du système. Cependant, leur faiblesse réside dans leur inadéquation avec les mécanismes de synchronisation, souvent nécessaires pour la modélisation des systèmes concurrents. L'ajout de ces mécanismes par des moyens *ad hoc* - les sémaphores [SF86] ou encore les mécanismes fork/join [DLT97] -, ne permet plus d'appliquer les méthodes analytiques classiques.

Dans une démarche opposée, les utilisateurs des RdPs, habitués à manipuler les mécanismes de concurrence à travers ce modèle, ont étendu leur formalismes en y ajoutant une temporisation [Mol81, FN85]. Ils se sont logiquement retrouvés confrontés au même problème que les utilisateurs de files d'attentes. Pour contourner ce problème, trois approches ont été envisagées :

– faire appel à la *simulation*. Elle consiste à reproduire l'évolution du système, *pas à pas,* en étudiant une réalisation particulière du modèle. L'avantage de la simulation est d'offrir une approche très générale permettant d'étudier n'importe quel modèle (dès l'instant où l'outil de simulation est adapté au modèle considéré). Son gros inconvénient est, cependant, d'être une technique extrêmement coûteuse en temps de calcul et de ne pas garantir une exploration exhaustive des comportements [GMKN00, Beu03].

– la deuxième approche est d'appliquer des méthodes approximatives, efficaces dans la pratique mais dont il est difficile de vérifier la validité des résultats [SE97, Val97].

– la troisième approche est de représenter le *processus stochastique*[1] sous forme de chaîne de Markov dont les états accessibles du modèle. Le problème ici est, bien sûr, *l'explosion combinatoire* de cet espace d'états.

Dans cette thèse, nous nous sommes intéressés plus particulièrement à cette troisième approche. Comme pour la vérification qualitative, plusieurs méthodes ont été développées pour contenir l'explosion combinatoire. Deux d'entre elles ont retenu notre attention, de par leur simplicité et leur efficacité, mais aussi parce qu'elles se combinent pour donner des approches hybrides intéressantes.

– La première opère au niveau du modèle (utilisé pour la spécification du système, e.g., les réseaux de Petri stochastiques, l'algèbre de processus stochastique). Elle consiste en la décomposition du modèle en plusieurs sous-modèles pour lesquels la génération de l'espace d'états est moins coûteuse (en espace et en temps). Le comportement du processus global est retrouvé (en utilisant l'algèbre de Kronecker) par une combinaison des comportements locaux [Don94, CDS99]. Le problème ici est justement de trouver la décomposition en sous-modèles pour laquelle les études globale et locale sont équivalentes.

– La deuxième approche repose sur les techniques d'agrégation markovienne pour la réduction de la taille du processus stochastique à étudier. En bénéficiant des symétries *fortes* exhibées par les comportements des systèmes distribués, il est possible de construire des graphes réduits isomorphes à des chaînes de Markov agrégées sur lesquels l'étude quantitative sera effectuée efficacement [DH89, Buc95].

  Cependant, comme pour la vérification qualitative, le problème des systèmes partiellement symétriques se pose encore une fois, et à notre connaissance, la seule approche qui tente de traiter ce genre de systèmes est [Cap00] (à laquelle nous consacrerons une partie du chapitre 8.3).

On note enfin l'existence de plusieurs approches hybrides qui tirent profit du meilleur des deux approches précédentes. Dans le cadre des réseaux de Petri de haut niveau [HM95, HM96] ou encore pour les algèbres de processus stochastiques [HR98].

1.4 Objectifs

Notre objectif est de proposer de nouvelles méthodes optimisées de vérification et d'évaluation de performances basées sur l'exploitation des symétries partielles.

[1] Un processus stochastique est un phénomène aléatoire dépendant du temps.

Ces méthodes réduisent le plus souvent de manière exponentielle les graphes d'états.

Pour la vérification qualitative, ces graphes réduits sont construits en fonction de la propriété à vérifier. En réalisant la vérification à la volée la construction du graphe réduit est stoppée quand la valeur de vérité de la propriété est établie. De plus, si la propriété est fausse une séquence d'exécution pourra être exhibée.

Pour l'évaluation de performance, nous utiliserons les techniques d'agrégation markovienne pour dériver des chaînes de Markov agrégées à partir des graphes réduits. Dans ce cadre, l'agrégation markovienne *exacte* va servir d'atout principal.

Notre travail est basé sur trois piliers majeurs par rapport aux techniques fondées sur l'exploitation des symétries :
– Automatiser la vérification de formules et l'évaluation de performances en exploitant les symétries partielles
– Reconcevoir les techniques définies pour aborder le cadre le plus large des systèmes partiellement symétriques.
– Modéliser les systèmes en réseau de Petri bien Formés et cela pour plusieurs raisons. D'une part, à cause de leur pouvoir d'expression, prouvé identique à celui des réseaux de Petri Colorés généraux. D'autre part, parce qu'ils offrent une approche symbolique fondée sur théorie mathématique très précise et qui a déjà prouvé son efficacité. Enfin, parce qu'ils intègrent les deux aspects qualitatifs et quantitatifs de manière élégante.

Pour l'aspect qualitatif, les développements de cette thèse seront présentés à partir de la logique temporelle à temps linéaire, puisque des prolongements vers les logiques arborescentes ont été démontrés. Nous utiliserons la syntaxe LTL de fait de sa facilité de comprehension et de sa large couverture des propriétés du système. Les proposions atomiques que nous considérons concernent des propriétés d'états sur lesquelles nous détecterons les symétries. L'interpretation sémantique des formules sera exploitées à partir des automates de Büchi et leurs variantes.

1.5 Organisation

Ce travail est organisé en deux parties : la première concerne la verification qualitative des systèmes concurrents fini et la deuxième est concentrée sur l'évaluation de performances de ces mêmes systèmes.

Dans la première partie, le chapitre 2 rappelle les principaux concepts qu'il convient de connaître dans le cadre de la vérification qualitative des systèmes concurrents suivant une approche comportementale.

Le chapitre 3 offre un état de l'art détaillé sur l'exploitation des symétries pour optimiser la vérification. Ce chapitre est le fruit d'un travail de synthèse des méthodes les plus intéressantes (relativement à notre travail), dans laquelle nous unifions les concepts et les notations.

Le chapitre 4 est consacré à la méthode de [HIA00] ainsi qu'aux améliorations que nous y apportons. Cette méthode a constitué le point de départ de cette thèse par le large potentiel d'optimisation non encore exploité jusqu'alors.

Le chapitre 5 présente une nouvelle optimisation de la vérification par l'exploitation des *tests d'inclusion* entre ensembles. Cette approche peut être vue comme la généralisation de l'approche du chapitre précédent.

Les chapitres 6 et 7 sont consacrés à l'application des méthodes précédentes au modèle des réseaux de Petri Bien Formés. Nous rappellerons les concepts de base de ce modèle, puis nous présenterons nos contributions pour la mise œuvres des techniques génériques introduites dans les chapitre 3, 4 et 5.

Dans la deuxième partie, le chapitre 8 fait un rapide rappel sur les chaînes de Markov et les méthodes d'agrégation markovienne, avec une application sur le modèle des réseaux de Petri bien Formés stochastiques.

Le chapitre 9 présente nos contributions pour l'évaluation de performances des systèmes partiellement symétriques. Il introduit le modèle des chaînes de Markov partiellement symétriques et leur dérivation à partir des réseaux de Petri bien Formés.

Enfin, le chapitre 10 conclut cette thèse avec une présentation des perspectives à court, moyen et long termes.

Première partie

Vérification qualitative des systèmes concurrents finis

Chapitre 2

Définitions préliminaires sur le model-checking

2.1 Introduction

Dans ce chapitre, nous rappelons les principaux concepts qu'il convient de connaître dans le cadre de la vérification qualitative des systèmes concurrents suivant une approche comportementale (*model-checking*).

La démarche classique pour opérer cette vérification peut être décomposée en deux phases. La première consiste en une modélisation et une spécification formelles du système et de la propriété à vérifier. La deuxième consiste en la confrontation de l'ensemble des comportements du système à la propriété, de façon à répondre à la question suivante : **existe-t-il un comportement dans le système qui ne vérifie pas la propriété ?**

Pour une présentation générique de ces concepts, nous allons nous placer au niveau de la sémantique, c'est-à-dire de la *Structure de Kripke Étiquetée*, engendrée par le modèle syntaxique quel qu'il soit.

Pour la spécification des propriétés, nous étudierons l'approche par la *logique temporelle linéaire (LTL)* et sa méthode d'évaluation basée sur les ω-*automates*. Ce choix est justifié d'une part par le fait que ce formalisme couvre la plupart des propriétés qu'on veut vérifier et d'autre part, parce qu'il s'adapte très bien aux techniques qui exploitent les symétries comportementales pour l'optimisation du procédé de vérification.

2.2 Sémantique associée à un système concurrent

A partir de son état initial, un système évolue et change d'état en réponse à des événements. Une manière naturelle de décrire les comportements possibles d'un tel système est de considérer l'ensemble des états accessibles et les transitions les liant les uns aux autres (sémantique par entrelacement)[1]. L'ensemble des comportements est représenté par un *Structure de Kripke Étiquetée*.

Définition 2.1 (Structure de Kripke Étiquetée)**.** *Une structure de Kripke étiquetée ($\mathcal{SKE}$) est un septuplet $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathcal{AP}, \mathcal{E}, \Pi, \Gamma \rangle$, défini par :*
- *$\mathcal{S}$ est un ensemble fini d'états,*
- *$\mathcal{S}_0 \subseteq S$ est l'ensemble des états initiaux,*
- *$\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ est la relation de transition.*
- *$\mathcal{AP}$ est un ensemble fini de propositions atomiques,*
- *$\mathcal{E}$ est un ensemble fini d'événements,*
- *$\Pi : \mathcal{S} \to 2^{\mathcal{AP}}$ est une application **injective**, qui associe à chaque état s l'ensemble des propositions atomiques qu'il satisfait.*
- *$\Gamma : \mathcal{R} \to \mathcal{E}$, est une application qui associe à chaque paire d'états un événement dans $\mathcal{E}$.*

On utilise la simplification $\langle s, e, s' \rangle \in \mathcal{R}$ (aussi, notée $s \xrightarrow{e} s'$) pour désigner la transition $\langle s, s' \rangle \in \mathcal{R}$ telle que $\Gamma(s, s') = e$.

Note 2.1. *L'injectivité de la fonction Π signifie qu'un état est entièrement caractérisé par la valeur des propositions atomiques. Les structures d'espace d'états issues des modèles les plus répandus vérifient toutes cette contrainte.*

Le point de départ pour l'étude d'une telle structure est de déterminer avec précision l'ensemble de tous les états qu'elle peut atteindre à partir de son état initial.

Définition 2.2 (États accessibles d'une $\mathcal{SKE}$)**.** *Soit $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathcal{AP}, \mathcal{E}, \Pi, \Gamma \rangle$ une $\mathcal{SKE}$. Un état s est* accessible *si $s \in \mathcal{S}_0$ ou s'il existe une séquence finie $\langle s_0, e_0, s_1 \rangle \langle s_1, e_1, s_2 \rangle \cdots \langle s_{n-1}, e_{n-1}, s_n \rangle$ de transitions de $\mathcal{R}$, commençant en un état initial $s_0 \in \mathcal{S}_0$, et se terminant par un état $s_n = s$. On note* $\mathrm{Reach}(\mathcal{K})$ *l'ensemble de tous les états accessibles de $\mathcal{K}$.*

Une exécution du système est formalisée par la définition suivante.

Définition 2.3 (Éxécution d'une $\mathcal{SKE}$)**.** *Soit $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathcal{AP}, \mathcal{E}, \Pi, \Gamma \rangle$ une $\mathcal{SKE}$. Une* exécution *de $\mathcal{K}$ est une séquence infinie $\langle s_0, e_0, s_1 \rangle \langle s_1, e_1, s_2 \rangle \cdots$*

[1] On considère un observateur qui est supposé "voir" un seul événement du système à la fois. La sémantique par entrelacement est donc donnée par toutes les séquences d'événements qui peuvent être observées à partir de l'état initial du système

de transitions de $\mathcal{R}$ commençant en un état initial $s_0 \in \mathcal{S}_0$. On note par $\mathrm{Run}(\mathcal{K})$ *l'ensemble de toutes les exécutions de $\mathcal{K}$. Pour une exécution $\sigma = \langle s_0, e_0, s_1\rangle\langle s_1, e_1, s_2\rangle \cdots$, on note $\sigma = \sigma(0)\sigma(1)\sigma(2)\cdots \in \mathrm{Run}(\mathcal{K})$, tel que $\sigma(i) = \langle \sigma_{\mathrm{in}}(i), \sigma_e(i), \sigma_{\mathrm{out}}(i)\rangle$. $\sigma_{\mathrm{in}}(i)$, $\sigma_e(i)$, et $\sigma_{\mathrm{out}}(i)$, sont : la source, l'événement, et la destination de la $i^{ème}$ transition de σ. Le suffixe de σ commençant en l'état s_i est noté $\sigma^i = \langle s_i, e_i, s_{i+1}\rangle \cdots$.*

Note 2.2. *La cas d'une séquence finie ne pose aucun problème théorique, car il est toujours possible de rendre la séquence infinie en "bégayant" sur le dernier état de la séquence, i.e., rajouter une transition de cet état vers lui-même.*

Si l'on est intéressé uniquement par l'enchaînement des états du système sans l'observation des événements, alors on parle d'une *structure de Kripke (non étiquetée)* $\mathcal{SK}$. La définition formelle d'une $\mathcal{SK}$ découle directement de celle de $\mathcal{SKE}$: $\mathcal{SK}$ est un quintuplet $\langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathcal{AP}, \Pi\rangle$. Ainsi, les définitions précédentes des états accessibles et des exécutions d'un système sont modifiées en conséquence (en ignorant les événements). Dans tout ce qui suit, les discussions portant sur le système se réfèrent à la $\mathcal{SK}$ et non la $\mathcal{SKE}$, sauf si cela est explicitement mentionné.

2.3 Spécification des propriétés d'un système concurrent

Vérifier qu'un système satisfait les propriétés attendues nécessite une représentation formelle de celles-ci. Le choix d'une représentation particulière dépend de la sémantique définissant les comportements possibles du système. Deux sémantiques sont généralement considérées. Dans le cadre de la sémantique de *temps arborescent*, l'ensemble des exécutions du système est considéré comme un arbre où les différents successeurs d'un état sont obtenus par les instances d'événements possibles pour cet état. Pour la sémantique de *temps linéaire*, on considère l'ensemble des exécutions comme des séquences indépendantes.

Les deux sémantiques présentent des avantages et des inconvénients largement discutés dans la littérature. Les arguments en faveur du temps linéaire, présentés par ses supportèrs, sont sa meilleure expressivité en particulier dans le domaine des propriétés de vivacité et d'équité. Les propriétés pouvant être exprimées par la sémantique de temps arborescent, par exemple "la possibilité", sont qualifiées par ces supportèrs comme sans intérêt dans le cadre des systèmes concurrents. Or, les arguments avancés pour la sémantique arborescente sont, justement, cette

capacité à exprimer “la possibilité”, mais aussi la plus grande efficacité, en terme de complexité, des algorithmes de vérification.

De par son expressivité et sa capacité à supporter les méthodes de vérification que nous allons mettre en œuvre, notre choix s’est fixé sur la sémantique de temps linéaire.

Dans ce contexte, nous présenterons le formalisme de la logique temporelle linéaire (LTL), dont l’évaluation est réalisée *via* la théorie des automates finis. LTL a été proposée par Lamport en 1977. Elle est très couramment utilisée.

2.3.1 Logique temporelle linéaire (LTL)

2.3.1.1 Syntaxe et sémantique

Les spécifications (formules) LTL sont construites à partir de propositions atomiques, de connecteurs booléens ($\wedge, \neg$) et d’opérateurs temporels (X "suivant", U "jusqu’à").

Définition 2.4 (Syntaxe de LTL)**.** *Soit $\mathcal{AP}$ un ensemble de propositions atomiques, alors les formules de LTL sont définies inductivement par les règles suivantes :*
– *chaque proposition atomique $p \in \mathcal{AP}$ est une formule,*
– *si f et g sont des formules alors $f \wedge g$, $\neg f$, $X f$ et $f U g$ en sont aussi.*

Les abréviations habituellement utilisées sont :

(ou)	$f \vee g$	$\Leftrightarrow$	$\neg(\neg f \wedge \neg g)$
(vrai)	True	$\Leftrightarrow$	$f \vee \neg f$
(faux)	False	$\Leftrightarrow$	$\neg$ True
(finalement)	$\mathbf{F}g$	$\Leftrightarrow$	True $U g$
(toujours)	$\mathbf{G}g$	$\Leftrightarrow$	$\neg\mathbf{F}\neg g$

La sémantique de LTL est formalisée en introduisant les notions de modèle et de satisfaction d’une formule LTL par un modèle.

Définition 2.5 (Modèle de LTL)**.** *Un modèle de LTL est une $\mathcal{SK}$, $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathcal{AP}, \Pi \rangle$ telle que $\mathcal{R}$ est une relation binaire totale, i.e., $\forall s \in \mathcal{S}, \exists s' \in \mathcal{S}$ tel que $s \rightarrow s'$.*

Note 2.3. *Traditionnellement, la logique temporelle raisonne sur des séquences infinies. En effet, celle-ci s’intéresse particulièrement à des propriétés d’équité qui n’ont de signification que dans ce contexte. Ceci explique la contrainte sur la*

relation $\mathcal{R}$.

Le problème de la satisfaction d'une formule par une exécution du système est défini comme suit.

Définition 2.6 (Sémantique de LTL)**.** *Soit $\mathcal{K}$ un modèle, s un état de $\mathcal{K}$ et $\sigma = \langle s_0, s_1\rangle\langle s_1, s_2\rangle\ldots$ une exécution de $\mathcal{K}$, alors pour une proposition atomique p et deux formules LTL quelconques f et g nous avons :*
- $\sigma \models p \quad \Leftrightarrow \quad p \in \Pi(\sigma_{in}(0))$,
- $\sigma \models f \wedge g \quad \Leftrightarrow \quad (\sigma \models f) \wedge (\sigma \models g)$,
- $\sigma \models \neg f \quad \Leftrightarrow \quad \neg(\sigma \models f)$,
- $\sigma \models X f \quad \Leftrightarrow \quad \sigma^1 \models f$,
- $\sigma \models fUg \quad \Leftrightarrow \quad \exists i$ *tel que* $(\forall j < i, \sigma^j \models f) \wedge (\sigma^i \models g)$.

Nous définissons alors la satisfaction de f par $\mathcal{K}$ comme suit :
- $\mathcal{K} \models f \quad \Leftrightarrow \quad \forall \sigma \in \mathrm{Run}(\mathcal{K}), \sigma \models f$

2.3.1.2 Méthode d'évaluation

Les ω-automates sont des automates finis acceptant des mots de longueur infinie (on parle alors d'ω-mots) [GTW02]. Dans l'approche automate du model-checking (Fig. 2.1), le comportement d'un système S à vérifier est vu comme un ω-automate $\mathcal{A}_S$. Les lettres de ces ω-mots correspondent chacune à des configurations du système. Un ω-mot représente alors une séquence d'exécution du système qui traverse chacune de ces configurations. Le langage de l'automate, noté $\mathcal{L}(\mathcal{A}_S)$, est l'ensemble des ω-mots qu'il reconnaît ; il représente l'ensemble des comportements possibles du système.[2]

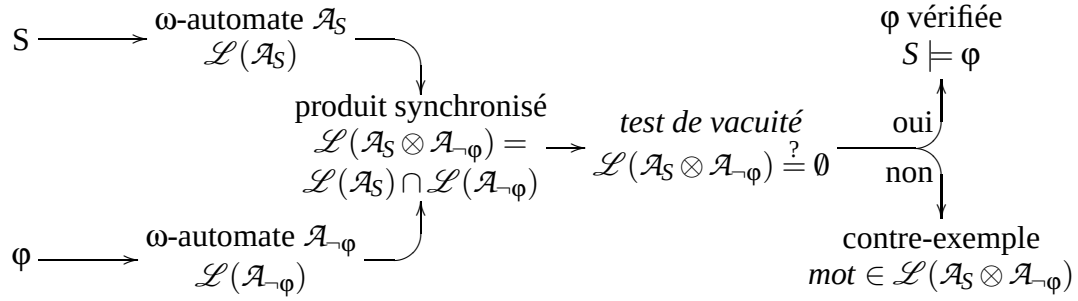


FIG. 2.1 – Approche automate du model-checking.

[2]En fait, $\mathcal{A}_s$ fait référence à la structure de Kripke (étiquetée ou non) représentant l'ensemble des exécutions du système. Les ω-mots sont construits à partir des étiquettes des états traversés par les exécutions du système.

Par ailleurs, la propriété comportementale φ que l'on veut vérifier sur *S* est elle-même exprimée par un ω-automate $\mathcal{A}_{\neg\varphi}$ dont le langage est l'ensemble des comportements qui *invalident* la propriété φ.

Un enchaînement de deux opérations permet de déterminer si $\mathcal{L}(\mathcal{A}_S)$ et $\mathcal{L}(\mathcal{A}_{\neg\varphi})$ partagent un ω-mot, c'est-à-dire, s'il existe une exécution de *S* qui ne vérifie pas φ. D'abord le produit synchronisé des deux automates est construit, il s'agit d'un ω-automate reconnaissant l'intersection des deux langages. Puis un *test de vacuité (emptiness-check)* de ce produit est effectué, cette opération détermine si le langage du système reconnu par un automate est vide.

Le langage du produit est vide lorsqu'il n'existe aucune séquence d'exécution de *S* qui invalide φ. Dans le cas contraire un *contre-exemple*, c'est-à-dire un ω-mot représentant une exécution du système interdite par φ, est retourné.

Cette approche permet de manipuler des ensembles infinis (les langages) en les représentant par des structures finies (les ω-automates). Plusieurs types d'ω-automates existent et se distinguent, essentiellement, par la forme de leur *conditions d'acceptation*, c'est-à-dire la façon d'indiquer quels chemins infinis de l'automate correspondent à des ω-mots acceptés.

Dans ce cadre, deux types d'ω-automates ont été largement utilisés : *les automates de Büchi* ($\mathcal{BA}$) [Büc62] et leur variante appelée *automates de Büchi généralisés basés sur les transitions* ($\mathcal{TGBA}$) [Cou99, HV03]. Le pouvoir d'expression des $\mathcal{BA}$ et des $\mathcal{TGBA}$ est strictement équivalent. Cependant, en comparant les tailles d'un $\mathcal{BA}$ et d'un $\mathcal{TGBA}$ (en termes de nombre d'états et de transitions) représentant une même propriété, celle du $\mathcal{TGBA}$ est toujours plus petite que celle du $\mathcal{BA}$. On note aussi que la conversion d'un $\mathcal{TGBA}$ en un $\mathcal{BA}$ est toujours possible [GL02] et par conséquent, tous les résultats obtenus sur les $\mathcal{BA}$ restent valables pour les $\mathcal{TGBA}$. De plus (et c'est le plus important), l'un comme l'autre permet de représenter toute formule *LTL* [Var96] (l'inverse n'est pas vrai [Wol83]).

Nous formalisons ces concepts dans les paragraphes suivants.

Automates de Büchi. Les automates de Büchi sont une classe d'ω-automates pour lesquels le test de vacuité peut être réalisé efficacement.

Définition 2.7 (Automate de Büchi ($\mathcal{BA}$))**.** *Un automate de Büchi* $\mathcal{A} = \langle Q, Q_0, Q_{\mathcal{F}}, \mathcal{R}, \mathcal{AP}, \Pi \rangle$ *est défini par :*
– Q *est un ensemble fini d'états,*

- $Q_0 \subseteq Q$ *est l'ensemble des états initiaux,*
- $Q_\mathcal{F} \subseteq Q$ *est l'ensemble des états d'acceptation,*
- $\mathcal{R} \subseteq Q \times Q$ *est la relation de transition. On note la transition* $(q, q') \in \mathcal{R}$ *par* $q \rightarrow q'$*,*
- $\mathcal{AP}$ *est un ensemble de propositions atomiques,*
- $\Pi : Q \rightarrow 2^{\mathcal{AP}}$ *est une application qui associe à chaque état un ensemble de propositions atomiques.*

Les notions de chemin et de chemin acceptant necessaires à décider de la vacuité d'un automate de Büchi sont définis dans ce qui suit.

Définition 2.8 (Chemin et Chemin acceptant d'un $\mathcal{BA}$)**.** *Soit* $\mathcal{A}$ *un* $\mathcal{BA}$*.*
- *Un* chemin *de* $\mathcal{A}$ *est une séquence infinie* $\langle q_0, q_1\rangle\langle q_1, q_2\rangle \cdots$ *de transitions de* $\mathcal{R}$ *commençant dans un état initial* $q_0 \in Q_0$*.*
- *Un chemin est acceptant si, partant d'un état initial, il passe une infinité de fois par un état d'acceptation :* $\exists f \in Q_\mathcal{F}, \forall i \geqslant 0, \exists j \geqslant i$*, tel que* $q_j = f$*.*

Pour savoir si une exécution du système est reconnue par l'automate, il faut qu'à chaque transition effectuée dans cette exécution corresponde une transition dans l'automate. Cette correspondance est formalisée par le concept du *produit synchronisé*.

Définition 2.9 (Produit synchronisé entre une $\mathcal{SK}$ et un $\mathcal{BA}$)**.** *Soit* $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}_\mathcal{K}, \mathcal{AP}, \Pi_\mathcal{K}\rangle$ *un* $\mathcal{SK}$ *et* $\mathcal{A} = \langle Q, Q_0, Q_\mathcal{F}, \mathcal{R}_a, \mathcal{AP}, \Pi_a\rangle$ *un* $\mathcal{BA}$*. Le produit synchronisé entre* $\mathcal{K}$ *et* $\mathcal{A}$ *est un* $\mathcal{BA}$*,* $\mathcal{K} \otimes \mathcal{A} = \langle \mathcal{V}, \mathcal{V}_0, \mathcal{V}_\mathcal{F}, \mathcal{R}, \mathcal{AP}, \Pi\rangle$ *défini par :*
- $\mathcal{V} = \{(s, q) \in \mathcal{S} \times Q \mid \Pi_a(q) \subseteq \Pi_\mathcal{K}(s)\}$ *est l'ensemble des états,*
- $\mathcal{V}_0 = (\mathcal{S}_0 \times Q_0) \cap \mathcal{V}$ *est l'ensemble des états initiaux,*
- $\mathcal{V}_\mathcal{F} = (\mathcal{S} \times Q_\mathcal{F}) \cap \mathcal{V}$ *est l'ensemble des états d'acceptation,*
- $\mathcal{R} = \{((s, q), (s', q')) \in \mathcal{V} \times \mathcal{V} \mid s \rightarrow s' \wedge q \rightarrow q'\}$ *est une relation de transition.*
- $\Pi : \mathcal{V} \rightarrow 2^{\mathcal{AP}}$ *tel que* $\Pi(s, q) = \Pi_\mathcal{K}(s)$*.*

Ainsi, le test de vacuité revient à la recherche d'un chemin dans le produit synchronisé qui passe une infinité de fois par un état d'acceptation (*i.e.*, un chemin acceptant).

Automates de Büchi généralisés basés sur les transitions. Un $\mathcal{TGBA}$ se distingue d'un $\mathcal{BA}$ par le fait que les propositions atomiques et les conditions d'acceptation se réfèrent *aux transitions* plutôt qu'aux états.

Définition 2.10 (Automate de Büchi généralisé basé sur les transitions ($\mathcal{TGBA}$))**.** *Un* $\mathcal{TGBA}$ *est un automate de Büchi dont les propositions atomiques et les*

conditions d'acceptation sont sur les transitions. Il est défini par un septuplet $\mathcal{A} = \langle Q, Q_0, \mathcal{R}, \mathcal{AP}, \mathcal{F} \rangle$*, où*
- Q *est un ensemble d'états fini,*
- $Q_0 \subseteq Q$ *est un ensemble fini d'états initiaux,*
- $\mathcal{AP}$ *est un ensemble fini de propositions atomiques,*
- $\mathcal{F}$ *est un ensemble fini d'éléments appelés* conditions d'acceptation*,*
- $\mathcal{R} \subseteq Q \times 2^{\mathcal{AP}} \times 2^{\mathcal{F}} \times Q$ *est la relation de transition.*

Nous utilisons la notation $q \xrightarrow{P,F} q'$ *pour désigner la transition* $\langle q, P, F, q' \rangle \in \mathcal{R}$*.*

Les notions de chemins et chemins acceptant sont redéfinis en conséquence.

Définition 2.11 (Chemin et Chemin acceptant d'un $\mathcal{TGBA}$)**.** *Soit* $\mathcal{A}$ *un* $\mathcal{TGBA}$*.*
- *Un* chemin *de* $\mathcal{A}$ *est une séquence infinie* $\langle s_0, P_0, F_0, s_1 \rangle \langle s_1, P_1, F_1, s_2 \rangle \cdots$ *de transitions de* $\mathcal{R}$ *commençant dans un état initial* $s_0 \in \mathcal{V}_0$*.*
- *Un chemin est acceptant si* $\forall f \in \mathcal{F}, \forall i \geqslant 0, \exists j \geqslant i$*, tel que* $f \in F_j$*, c'est-à-dire, si ses transitions sont étiquetées infiniment souvent par chaque condition d'acception.*

Aussi, la définition du produit synchronisé entre une $\mathcal{SK}$ et un $\mathcal{TGBA}$ est donnée par ce qui suit.

Définition 2.12 (Produit synchronisé entre une $\mathcal{SK}$ et un $\mathcal{TGBA}$)**.** *Soit* $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}_\mathcal{K}, \mathcal{AP}, \Pi_\mathcal{K} \rangle$ *une* $\mathcal{SK}$ *et* $\mathcal{A} = \langle Q, Q_0, \mathcal{R}_a, \mathcal{AP}, \mathcal{F} \rangle$ *un* $\mathcal{TGBA}$*. Le produit synchronisé entre* $\mathcal{K}$ *et* $\mathcal{A}$ *est un* $\mathcal{TGBA}$*,* $\mathcal{K} \otimes \mathcal{A} = \langle \mathcal{V}, \mathcal{V}_0, \mathcal{R}, \mathcal{AP}, \mathcal{F} \rangle$*, définit par :*
- $\mathcal{V} = \mathcal{S} \times Q$ *est l'ensemble des états,*
- $\mathcal{V}_0 = \mathcal{S}_0 \times Q_0$ *est l'ensemble des états initiaux,*
- $\mathcal{R} = \{((s,q), P, F, (s',q')) \in \mathcal{V} \times 2^{\mathcal{AP}} \times 2^{\mathcal{F}} \times \mathcal{V} \mid s \rightarrow s' \wedge q \xrightarrow{P,F} q' \wedge P \subseteq \Pi_\mathcal{K}(s)\}$ *est la relation de transition.*

2.4 Conclusion

La présentation des deux types d'automates (bien qu'ils soient équivalents) est justifiée par le fait que - presque - toutes les méthodes existantes, en particulier celles que nous présenterons au prochain chapitre, utilisent les $\mathcal{BA}$, tandis que notre propre travail utilise les $\mathcal{TGBA}$. Nous avons gardé les $\mathcal{BA}$ pour les approches existantes afin de ne pas les dénaturer et nous avons utilisé les $\mathcal{TGBA}$ pour nos approches car ils permettent une plus grande optimisation du processus de vérification.

En effet, la recherche d'un chemin acceptant sur le produit synchronisé d'une structure de Kripke, $\mathcal{K}$, et un automate $\mathcal{A}$ ($\mathcal{BA}$ ou $\mathcal{TGBA}$) est réalisée en $O(|\mathcal{K}| \times |\mathcal{A}|)$ tel que $|\mathcal{K}| = |\mathcal{S}| + |\mathcal{R}|$ (de même pour $|\mathcal{A}|$). Par conséquent, on a intérêt à choisir la structure la plus réduite pour l'expression de la propriété.

En outre, pour l'efficacité de la vérification, le problème de la taille de $\mathcal{K}$ est, bien évidemment, plus crucial que celui de la taille de $\mathcal{A}$. Dans la continuité des techniques qui ont été présentées dans l'introduction, nous allons consacrer le chapitre suivant aux méthodes de réduction de la taille de $\mathcal{K}$.

Chapitre 3

Exploitation des symétries pour le model-checking

La technique de vérification présentée dans le chapitre précédent a l'avantage d'être automatisable. Néanmoins elle nécessite de parcourir l'espace d'états du système et est donc confrontée au problème connu de *l'explosion combinatoire* en espace et en temps. Ceci rend sa gestion par les outils de vérification basés sur une représentation explicite pratiquement impossible. Il est donc primordial de développer des techniques efficaces pour réduire la taille du graphe à construire afin de minimiser l'effet de cette explosion sur les algorithmes de vérification.

3.1 Introduction

Parmi les solutions formelles qui tendent à optimiser le processus de vérification et à alléger le poids de l'explosion combinatoire, notre travail s'est focalisé sur l'exploitation des symétries présentes dans les systèmes et les propriétés. Dans un système concurrent, on rencontre souvent des architectures symétriques où les processus ont des comportements largement équivalents et ont des environnements de communication similaires. En effet, les conceptions symétriques ont été particulièrement utilisées pour renforcer la tolérance aux fautes dans les systèmes.

Deux points cruciaux soulignent l'extrême difficulté de la mise en application de telles approches :

– le choix d'une relation de symétrie est essentiel. Plus cette relation est fine, plus son expression est aisée, mais plus l'intérêt de son application à des cas concrets est limité.

– La mise en œuvre d'une réduction d'espace d'états en exploitant une relation de symétrie n'a de sens que si la construction de l'espace réduit est directe, c'est-à-dire qu'elle ne passe pas par la représentation explicite de tous les états.

Les méthodes qui traitent du sujet sont nombreuses et de diverses origines. Ceci rend leur compréhension par les non spécialistes (du domaine dont elles sont issues) toujours difficile. Ce constat nous a amené à élaborer une synthèse des méthodes les plus intéressantes (relativement à notre travail), dans laquelle nous unifions les concepts et les notations, au risque de diverger des présentations originelles des méthodes. Cependant, nous respectons toujours leurs idées principales.

Notons enfin que cette synthèse est basée essentiellement sur les travaux d'Emerson & al [ES96, ET99], d'Ajami & al [AHl98, IA97], d'Haddad & al [HITZ95], et ceux de Capra & al [Cap00].

3.2 Les symétries globales d'un système concurrent

D'un point de vue formel, il est communément admis de supposer la connaissance de l'espace des états potentiels du système pour définir la notion de symétrie. Dans notre cas, cet espace d'états est représenté par une structure de Kripke (étiquetée ou non) (définition 2.1). Les symétries d'une telle structure sont généralement manipulées par la théorie des groupes. Nous en rappelons ici quelques notions élémentaires.

Définition 3.1 (Groupe et groupe opérant sur un ensemble)**.**
– *Un groupe* $(G,\bullet,e)$*, noté* $\mathcal{G}$*, est un ensemble G muni d'un élément neutre nécessairement unique e, et d'une loi de composition interne* $\bullet$*, qui satisfait les axiomes suivants :*
  – *identité :* $\forall x \in G, x \bullet e = e \bullet x = x$ *;*
  – *inverse :* $\forall x \in G, \exists y \in G, x \bullet y = y \bullet x = e$*. y est dit inverse de x et on le note aussi* x^{-1} *;*
  – *associativité :* $\forall x,y,z \in G, x \bullet (y \bullet z) = (x \bullet y) \bullet z$*.*

  Lorsque G est un ensemble fini, on dit que $\mathcal{G}$ *est un groupe fini ; sinon c'est un groupe infini. Pour un groupe fini, l'ordre (cardinal) de ce groupe est :* $|G|$*. Aussi, pour simplifier on utilise la notation :* $g \in \mathcal{G}$ *pour* $g \in G$*.*

– *Soit H un sous-ensemble de G. On dit que* $(H,\star,e')$*, noté* $\mathcal{H}$*, est un sous-groupe de* $\mathcal{G}$ *(*$\mathcal{H} \subseteq \mathcal{G}$*) si* $\mathcal{H}$ *est un groupe dont la loi* $\star$ *s'obtient par restriction de* $\bullet$ *à* $H \times H$ *et* $e' = e$*. Dans la pratique, on note la loi interne du sous-groupe avec*

le même symbole que celui de la loi interne du groupe, c'est-à-dire $\bullet$.

- *L'action d'un groupe* $\mathcal{G}$ *sur un ensemble* E *est une application de* $G \times E$ *dans* E*, notée "*$.$*", qui associe au couple* $(g,x) \in G \times E$*, l'élément* $g.x$ *tel que :* $\forall x \in E\,, e.x = x$ *et* $\forall g, g' \in \mathcal{G}$, $(g \bullet g').x = g.(g'.x)$.

- *Le* stabilisateur *(sous-groupe d'isotropie) du sous-ensemble* $O \subseteq E$*, noté* $\mathcal{G}_O$*, est défini par :* $\mathcal{G}_O = \{g \in \mathcal{G} \mid \forall x \in O, g.x \in O\}$.

- *Une action* g *of* $\mathcal{G}$ *peut être trivialement étendue à tout ensemble* $E' \subseteq E$ *par :* $g.E' = \{g.x \mid x \in E'\}$.

- *Soit* $\mathcal{H} \subseteq \mathcal{G}$. L'orbite *de* $x \in E$ *par* $\mathcal{H}$*, notée* $\mathcal{H}.x$*, est défini par :* $\mathcal{H}.x = \{g.x \mid g \in \mathcal{H}\}$. *L'ensemble des orbites par* $\mathcal{H}$ *définit une partition de* E. *L'orbite d'un élément* $x \in E$ *est la classe d'équivalence de* x *par rapport à la relation d'équivalence* $\sim$ *définie par :* $x \sim x' \Leftrightarrow \exists g \in \mathcal{H}$ *tel que* $x' = g.x$.

Dans le contexte actuel, un groupe particulier retient notre attention, c'est le *groupe de permutations* d'un ensemble E, noté $\mathfrak{S}(E)$: *le groupe de permutations d'un ensemble* E *est le groupe de toutes les bijections de* E *sur lui-même, muni de la loi de composition.* [1]

Intuitivement, les symétries d'une structure de Kripke sont exprimées par des applications bijectives (permutations) sur les états de la structure, qui préservent la relation de transition. Plus précisément, les permutations sont définies sur l'ensemble des étiquettes des états (l'ensemble $\mathcal{AP}$). On dit que deux états sont permutables si en appliquant une permutation sur l'étiquette de l'un on obtient l'étiquette de l'autre. De plus, pour qu'une permutation préserve la relation de transition dans la structure il faut que deux conditions soient vérifiées :
- son application sur un état du graphe donne un autre état du graphe ;
- son application sur n'importe quel successeur du premier état donne un successeur du deuxième état.

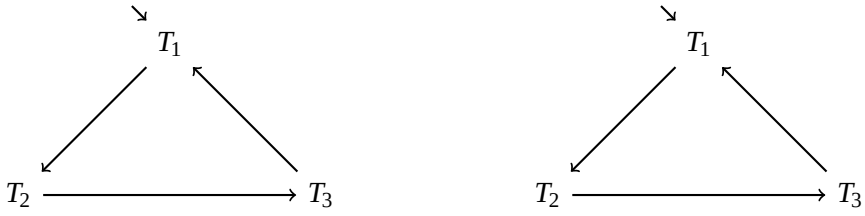
Définition 3.2 (Structure de Kripke symétrique par rapport à $\mathcal{G}$)**.** *Soit* $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathcal{AP}, \Pi \rangle$ *une* $\mathcal{SK}$ *et* $\mathcal{G} \subseteq \mathfrak{S}(\mathcal{AP})$. $\mathcal{K}$ *est dit symétrique par rapport à* $\mathcal{G}$ *ssi :*
- *Chaque état a un symétrique par rapport à chaque élément de* $\mathcal{G}$ *:* $\forall s \in \mathcal{S}, \forall g \in \mathcal{G}, \exists s' \in \mathcal{S}, \Pi(s') = g.\Pi(s)$. *L'action du groupe est étendue trivialement sur l'ensemble* $\mathcal{S}$ *et en notant* $g.s$*, l'unique état* s'.

[1] L'élément neutre est la fonction identité, habituellement notée "id".

– *L'ensemble des états initiaux est globalement invariant sous l'action de* $\mathcal{G}$ *:* $\mathcal{G}.\mathcal{S}_0 = \mathcal{S}_0$.
– *L'action de* $\mathcal{G}$ *est congruente par rapport à la relation de transition :* $\forall s, s' \in \mathcal{S}, \forall g \in \mathcal{G}, s \to s' \Leftrightarrow g.s \to g.s'$.

Note 3.1. *Le cas des structures de Kripke étiquetée est simplement plus général. Ainsi, quand il est question d'une structure* $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathcal{AP}, \mathcal{E}, \Pi, \Gamma \rangle$*, alors* $\mathcal{G}$ *agit sur* $\mathcal{E}$*,* $\mathcal{G} \subseteq \mathfrak{S}(\mathcal{AP} \uplus \mathcal{E})$*, et la congruence par rapport à la relation de transition doit s'étendre aux événements :* $\forall s, s' \in \mathcal{S}, e \in \mathcal{E}, \forall g \in \mathcal{G}, s \xrightarrow{e} s' \Leftrightarrow g.s \xrightarrow{g.e} g.s'$.

Exemple 3.1. *Soit un système concurrent composé deux processus* p_1 *et* p_2 *exécutant, indéfiniment, les trois tâches successives* T_1*,* T_2 *et* T_3*, en commençant par* T_1 *(Fig. 3.1).*



Comportement p_1 Comportement p_2

FIG. 3.1 – Exemple d'un système concurrent.

Le comportement global de ce système est représenté par la structure de Kripke de la Fig. 3.2. L'ensemble des nœuds de la structure représentent l'ensemble $\mathcal{S}$*. L'ensemble* $\mathcal{AP}$ *est défini par* $\mathcal{AP} = \{T_{i,j} \mid i \in \{1,2,3\} \wedge j \in \{1,2\}\}$ *tel que* $T_{i,j}$ *signifie que le processus* p_j *est en train d'accomplir la tâche* T_i*.*

Nous définissons le sous-groupe $\mathcal{G} \subseteq \mathfrak{S}(\mathcal{AP})$ *tel que* $\mathcal{G} = \{id, g\}$ *où* g *est défini par :* $g.T_{i,1} = T_{i,2}$ *et* $g.T_{i,2} = T_{i,1}$*,* $i \in \{1,2,3\}$*.*[2]

Par application de la définition 3.2, nous constatons que la structure $\mathcal{K}$ *est symétrique par rapport au groupe* $\mathcal{G}$ *: chaque état de* $\mathcal{K}$ *a un symétrique par rapport à un élément de* $\mathcal{G}$ *; l'ensemble des états initiaux est invariant par ce groupe ; l'action de* $\mathcal{G}$ *est congruente par rapport à la relation de transition.*

[2]Nous pouvons facilement vérifier que $\mathcal{G}$ est un groupe sur $\mathcal{AP}$.

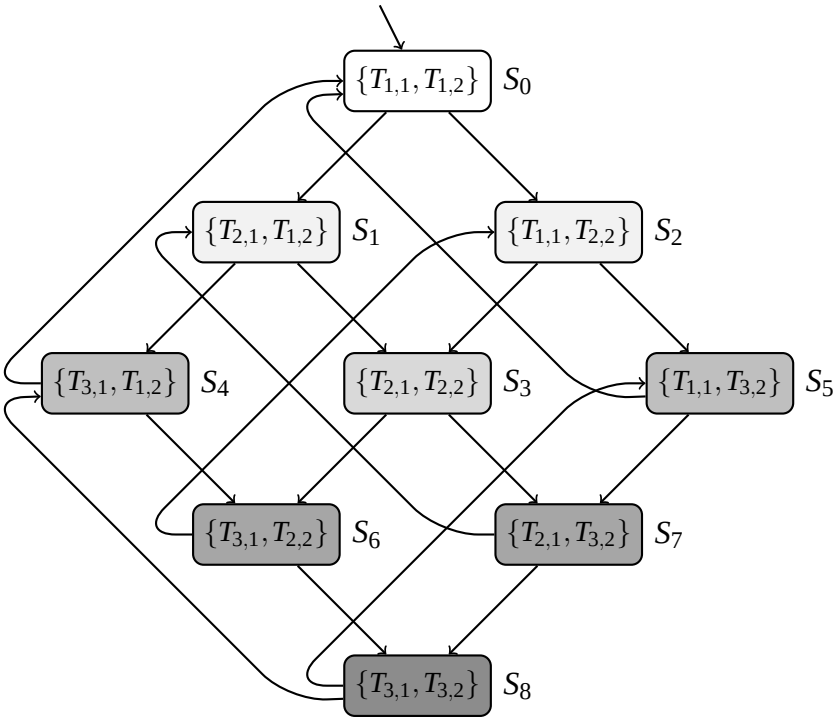


FIG. 3.2 – Structure de Kripke, $\mathcal{K}$, du système de la Fig.3.1.

La relation d'équivalence $\sim$, qu'on appelle "est symétrique de", définie par les éléments du sous-groupe $\mathcal{G}$ (voir définition 3.1) permet la construction d'un graphe réduit (modulo $\mathcal{G}$ ou l'un de ses sous-groupes $\mathcal{H} \subseteq \mathcal{G}$) d'une structure de Kripke.

Ce graphe, appelé *graphe quotient*, est une structure de Kripke qui vérifie les deux points suivants :
- ses nœuds sont des classes d'équivalence de la relation "est symétrique de".
- il existe un arc d'un nœud A vers un nœud B ssi il existe un arc d'un état de A vers un état de B.

Définition 3.3 (Graphe quotient d'une structure de Kripke)**.** *Soit* $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathcal{AP}, \Pi \rangle$ *une* $\mathcal{SK}$ *symétrique p.r.a.* $\mathcal{G} \subseteq \mathfrak{S}(\mathcal{AP})$*. Le graphe quotient* $\overline{\mathcal{K}}$ *de* $\mathcal{K}$ *p.r.a.* $\mathcal{H} \subseteq \mathcal{G}$ *est une structure de Kripke définie par* $\overline{\mathcal{K}} = \langle \overline{\mathcal{S}}, \overline{\mathcal{S}_0}, \overline{\mathcal{R}}, \mathcal{AP}, \Pi \rangle$ *:*
- $\overline{\mathcal{S}} = \{\overline{s} \in \mathcal{S} \mid \overline{s}$ *est un unique représentant de la classe d'équivalence* $\mathcal{H}.\overline{s}\}$,
- $\overline{\mathcal{S}_0} = \overline{\mathcal{S}} \cap \mathcal{S}_0$,
- $\overline{\mathcal{R}} = \{(\overline{s}, \overline{t}) \in \overline{\mathcal{S}} \times \overline{\mathcal{S}} \mid \exists\, s \in \mathcal{H}.\overline{s}, t \in \mathcal{H}.\overline{t},\ (s,t) \in \mathcal{R}\}$*. On note* $(\overline{s}, \overline{t}) \in \overline{\mathcal{R}}$ *par* $\overline{s} \rightarrow \overline{t}$.

Exemple 3.2. *Dans la structure de Kripke de l'exemple 3.1, les nœuds ayant le même niveau d'ombrage graphique appartiennent à la même classe d'équivalence par rapport à la relation "est symétrique de", induite par le sous-groupe $\mathcal{G}$ de l'exemple précédent. Nous avons donc les six classes suivantes :* $\{S_0\},\{S_1,S_2\},\{S_3\},\{S_4,S_5\},\{S_6,S_7\},\{S_8\}$.

La structure $\overline{\mathcal{K}}$, construite en prenant $\mathcal{H}=\mathcal{G}$ et en choisissant un représentant pour chaque classe, est présentée dans la Fig. 3.3 : $\overline{S}_0=S_0$, $\overline{S}_1=S_2$, $\overline{S}_2=S_3$, $\overline{S}_3=S_4$, $\overline{S}_4=S_6$ *et* $\overline{S}_5=S_8$.

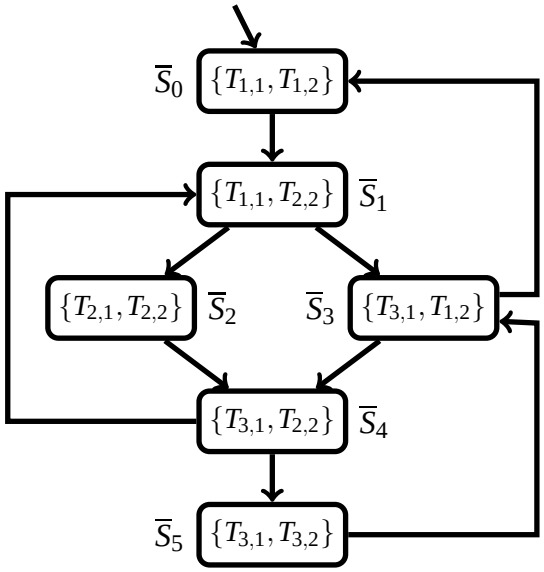


FIG. 3.3 – Structure quotient, $\overline{\mathcal{K}}$, de $\mathcal{K}$ (de la Fig. 3.2).

Le graphe quotient ainsi construit est une représentation compacte du graphe d'états et peut être utilisé pour la résolution de problèmes d'accessibilité d'états. Dans ce cadre, nous pouvons citer les travaux de Jensen [HJJK86] pour les réseaux de Petri Colorés, ceux de Haddad [Had87] pour les réseaux de Petri Réguliers, ou encore ceux de Dutheillet [Dut91] pour les réseaux de Petri Bien Formés. A la différence du premier formalisme, les deux autres permettent une construction automatique du graphe quotient, en partant d'une représentation générique des classes d'équivalence (des états et des arcs).[3]

[3]Nous allons reprendre, en détail, le formalisme des réseaux de Petri Bien formés dans le chapitre 6.

3.3 Exploitation des symétries globales pour la vérification de propriétés temporelles

Sans contrainte supplémentaire, le graphe quotient défini précédemment ne permet pas de vérifier des propriétés complexes telles que celles exprimées par LTL. En effet, cette logique étant définie pour la vérification de propriétés de chemins, elle nécessite une connaissance parfaite (ou du moins assez fine) sur les enchaînements des états du système. Or, dans le graphe quotient certaines transitions entre états sont masquées par la représentation abstraite des transitions entre les classes d'états.

En observant la structure quotient de la Fig.3.3, nous somme tentés de croire que le système peut passer immédiatement d'un état où le processus p_1 est en train d'exécuter la tâche T_1 à celui où il exécute la tâche T_3, à cause de la transition entre $\overline{S}_1$ et $\overline{S}_3$. Or, la structure (ordinaire) de la Fig.3.2 nous informe que ce passage est impossible.

Pour résoudre ce problème, Emerson & al [ES96] proposent de ne conserver que les symétries qui laissent la propriété temporelle à vérifier *invariante*. Ce sont ensuite ces symétries qui sont utilisées pour construire la structure quotient. De cette manière, ils garantissent que la vérification de la propriété peut s'effectuer *indifféremment* sur la structure de Kripke ou sur son graphe quotient. Deux approches ont été proposées pour mettre en œuvre cette idée : l'approche syntaxique et l'approche par automate.

3.3.1 Approche syntaxique

L'approche syntaxique consiste à détecter l'insensibilité de la propriété au niveau de son expression syntaxique (*i.e.*, sous la forme d'une formule LTL). Ceci revient à rechercher le sous-groupe de permutations $\mathcal{G}_f$ agissant sur les propositions atomiques composant la formule tel que l'application de n'importe quel éléments de $\mathcal{G}_f$ sur f ne change pas sa valeur de vérité.

Définition 3.4 (Groupe de symétries d'une formule LTL)**.** *Soit f une formule LTL, construite autour d'un ensemble de propositions atomiques $\mathcal{AP}$. Le sous-groupe de permutations de f, noté $\mathcal{G}_f$, est défini récursivement par :*
– *Si f est une formule proportionnelle, alors $\mathcal{G}_f = \{g \in \mathfrak{S}(\mathcal{AP}) \mid g(f) = f\}$.*
– *Si $f = X\,f'$ alors, $\mathcal{G}_f = \mathcal{G}_{f'}$.*
– *Si $f = f'U\,f''$ alors, $\mathcal{G}_f = \mathcal{G}_{f'} \cap \mathcal{G}_{f''}$.*

– *Autres cas : Si* $f = b(e_1, \ldots, e_k, f_1, \ldots, f_l)$ *où b est une formule booléenne sur les propositions atomiques* $e_1, \ldots, e_k$ *et les sous-formules* $f_1, \ldots, f_l$*, alors nous remplaçons* f_i *par* F_i *et* $\mathcal{G}_f = \mathcal{G}_{b(e_1,\ldots,e_k,F_1,\ldots,F_l)} \cap \mathcal{G}_{f_1} \cap \ldots \cap \mathcal{G}_{f_l}$.
Où $f_i = X f'$ *ou* $f_i = f' U f''$ *et* $b(e_1, \ldots, e_k, F_1, \ldots, F_l)$ *est une formule propositionnelle.*

Note 3.2. *L'introduction de* F_i *à la place de* f_i *est un renommage qui permet de faire disparaître tous les opérateurs temporels et de transformer la formule temporelle initiale en une formule propositionnelle.*

Ainsi, Emerson à prouvé que si $\mathcal{K}$ est une $\mathcal{SK}$ symétrique par rapport à un sous-groupe $\mathcal{G}$, et f une formule LTL alors la vérification de f peut être effectuée sur une structure quotient $\overline{\mathcal{K}}$ construite par rapport à un sous-groupe $\mathcal{H} \subseteq \mathcal{G} \cap \mathcal{G}_f$ (*i.e.,* $\overline{\mathcal{K}}$ est construite sur la base des permutations communes au système et à la formule).

Exemple 3.3. *Considérons les deux propriétés suivantes (à vérifier sur le système de la Fig. 3.1) : (1) le système passe infiniment souvent par un état où l'un des deux processus* p_1 *ou* p_2 *est en train d'exécuter la tâche* T_3 *; (2) immédiatement après l'exécution de la tâche* T_1 *le processus* p_1 *exécute la tâche* T_3.

Ces propriétés sont exprimées respectivement par les formules LTL : $f = GF(T_{3,1} \vee T_{3,2})$ *et* $f' = F(T_{1,1} \wedge X T_{3,1})$*. En appliquant les règles de construction de la définition 3.4, nous obtenons* $\mathcal{G}_f = \{id, g\}$ *tel que* $g.T_{3,1} = T_{3,2}$ *et* $g.T_{3,2} = T_{3,1}$ $(g^{-1} = g)$ *et* $\mathcal{G}_{f'} = \{id\}$ *(aucune symétrie n'est possible pour* f'*).* [4]

La vérification de f *peut donc s'opérer sur le graphe quotient construit en prenant* $\mathcal{H} = \mathcal{G} \cap \mathcal{G}_f = \mathcal{G}$*, c'est à dire, le graphe de la Fig.3.3. Celle de* f' *ne peut s'effectuer que sur le graphe ordinaire de la Fig. 3.2, car* $\mathcal{H} = \mathcal{G} \cap \mathcal{G}_{f'} = \{id\}$.

L'approche syntaxique offre un cadre formel très précis pour l'exploitation des symétries pour la vérification des propriétés temporelles. Cependant, elle reste inefficace devant des formules du type $f = \bigwedge_i f_i$ ou $f = \bigvee_i f_i$ où les f_i sont des formules LTL identiques à l'indice *i* près.

Par exemple, le groupe de permutations (calculé suivant l'approche syntaxique) de la formule $f = FG(T_{2,1}) \vee FG(T_{2,2})$ est réduit à l'identité : $\mathcal{G}_f = \{id\}$. Or, nous

[4] Notons que nous avons utilisé les simplifications G et F pour exprimer les propriétés : $GF(T_{3,1} \vee T_{3,2}) = \neg(True\, U\, \neg(True\, U\, (T_{3,1} \vee T_{3,2})))$ et $F(T_{1,1} \wedge X T_{3,1}) = True\, U\, (T_{1,1} \wedge X T_{3,1})$

pouvons aisément comprendre que les exécutions satisfaisants cette formule sont identiques à la permutation $[1 \rightarrow 2, 2 \rightarrow 1]$ près sur l'indice du processus.

Une première solution, proposée par Emerson, consiste à opérer la vérification de chaque sous-formule f_i séparément. Pour chacune d'elles, un graphe quotient est construit en séparant les comportements de "l'objet" concerné par l'indice *i* des comportements des autres objets.

Considérons que le système concurrent présenté précédemment est augmenté à *n* processus. Pour vérifier la formule $f = \bigvee_{i \in \{1 \ldots n\}} FG(T_{2,i})$, nous construisons un graphe quotient $\overline{\mathcal{K}_i}$ pour chaque sous-formule $f_i = FG(T_{2,i})$. Dans ce graphe, seules les exécutions du processus *i* sont identifiées avec exactitude (à la transition entres états ordinaires près). Nous obtenons alors une structure de taille intermédiaire, entre celles obtenue en considérant $\mathcal{G}_f = \{id\}$, et la structure ordinaire. On précise que cette manière de procéder n'est valable que dans le cas où tous les objets ont des comportements identiques, *i.e.*, la vérification du comportement de l'un suffit pour tous.

L'autre solution consiste à détecter les symétries directement au niveau de l'automate de Büchi représentant la formule à vérifier. Ceci nous amène à *l'approche par automate* de la section suivante.

3.3.2 Approche par automate

Une approche alternative pour vérifier les propriétés temporelles à temps linéaire sur une structure quotient (qui, entre autres, permet de gérer le problème soulevé précédemment), consiste à exploiter les symétries qui apparaissent sur l'automate de Büchi (représentant la propriété), en association avec celles inhérentes au système.

En effet, il est possible de définir un sous-groupe de permutations sur un automate de Büchi en utilisant l'assertion suivante : *deux chemins de l'automate sont équivalents s'ils reconnaissent les mêmes* ω*-mots à une permutation des indices près*.

Définition 3.5 (Automate de Büchi symétrique p.r.a. un groupe $\mathcal{G}$)**.** *Soit un automate de Büchi* $\mathcal{A} = \langle Q, Q_0, Q_\mathcal{F}, \mathcal{R}, \mathcal{AP}, \Pi \rangle$*, et* $\mathcal{G} \subseteq \mathfrak{S}(\mathcal{AP})$*.* $\mathcal{A}$ *est symétrique par rapport à* $\mathcal{G}$ *ssi :*

- *Chaque état a un symétrique p.r.a. chaque élément de* $\mathcal{G}$:
  $\forall q \in Q, \forall g \in \mathcal{G}, \exists q' \in Q, \Pi(q') = g.\Pi(q)$.
- $\forall q, q' \in Q$ *tels que* $\exists g \in \mathcal{G}, \Pi(q') = g.\Pi(q)$ *:*

- $q \in Q_0 \Leftrightarrow q' \in Q_0$ *et* $q \in Q_{\mathcal{F}} \Leftrightarrow q' \in Q_{\mathcal{F}}$.
- $\forall q_d \in Q, \exists q'_d \in Q$ *tels que* $q \rightarrow q_d \Rightarrow q' \rightarrow q'_d \wedge \Pi(q'_d) = g.\Pi(q_d)$.

Par la suite, nous noterons $\mathcal{G}_{\mathcal{A}}$, le plus grand sous-groupe vérifiant cette définition.

Exemple 3.4. *Considérons à nouveau la formule $f = FG(T_{2,1}) \vee FG(T_{2,2})$. L'automate de Büchi $\mathcal{A}_f$ représentant cette formule est représenté dans la Fig. 3.4.*

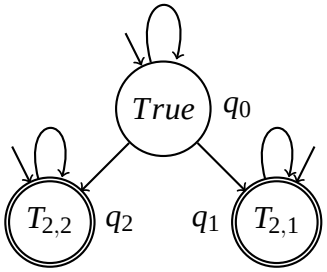


FIG. 3.4 – Automate de Büchi $\mathcal{A}_f$

En appliquant la définition 3.5, nous trouvons que cet automate est symétrique par rapport au groupe $\mathcal{G}_{\mathcal{A}} = \{id, g\}$ où g est défini par $g.T_{i,1} = T_{i,2}$ et $g.T_{i,2} = T_{i,1}$, $i \in \{1,2,3\}$. En effet, $g.\Pi(q_0) = \Pi(q_0)$, $g.\Pi(q_1) = \Pi(q_2)$,$g.\Pi(q_2) = \Pi(q_1)$ et le symétrique de la transition $q_0 \rightarrow q_1$ est $q_0 \rightarrow q_2$.

Grâce à cette propriété de symétrie, Emerson [ES96] prouve qu'il est possible de s'autoriser certaines "confusions" dans la représentation compacte des exécutions du système (par le biais du graphe quotient), à condition cependant de conserver dans le graphe une représentation explicite des transitions d'un état représentant d'une classe vers tous ses états successeurs dans une autre classe. Cette confusion ne gênera pas la vérification de la propriété. Cette représentation est réalisée en augmentant le graphe quotient de la définition 3.3 par une annotation ad-hoc sur les transitions entre les nœuds.

Considérons la transition $\overline{S}_0 \rightarrow \overline{S}_1$ de la Fig.3.3. Le fait que le représentant $\overline{S}_0$ atteint les deux états $\{S_1, S_2\}$ (représentés par l'état $\overline{S}_1$) est signifié par $\overline{S}_0 \xrightarrow{\{id,g\}} \overline{S}_1$ tel que $id.\overline{S}_1 = \overline{S}_1 = S_1$ et $g.\overline{S}_1 = S_2$. [5] Le graphe ainsi construit est appelé *graphe quotient annoté* et noté $\overline{\mathcal{K}}_{Ann}$. Le $\overline{\mathcal{K}}_{Ann}$ de la structure $\mathcal{K}$ de la Fig.3.2 est présenté dans la Fig.3.5.

[5] $\{id, g\} \subseteq \mathcal{G}$, et $\mathcal{K}$ est symétrique par rapport à $\mathcal{G}$.

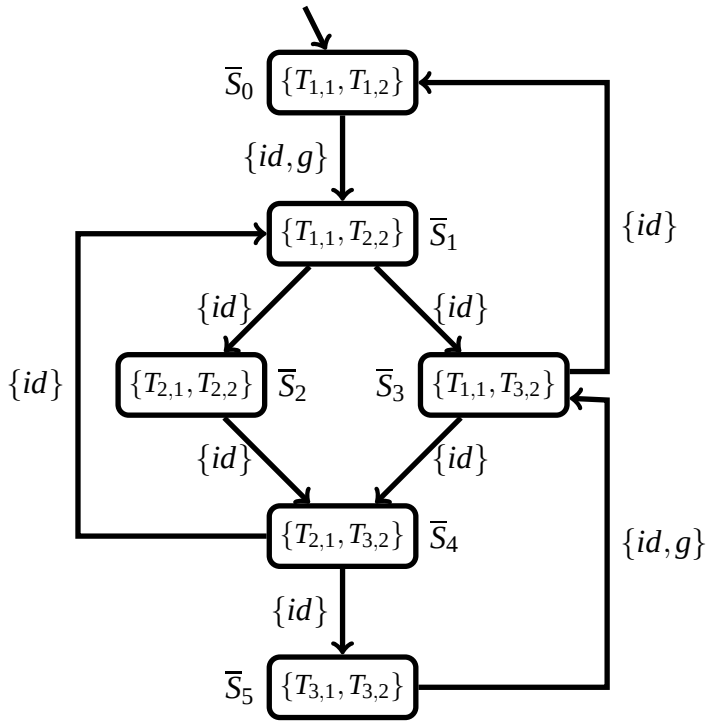


FIG. 3.5 – Structure quotient annotée, $\overline{\mathcal{K}}_{Ann}$, de $\mathcal{K}$ (de la Fig. 3.2).

Ainsi, si $\mathcal{G}$ est le sous-groupe de permutations d'une structure de Kripke $\mathcal{K}$ et $\mathcal{A}$ est un automate de Büchi symétrique par rapport à $\mathcal{G}_{\mathcal{A}}$, alors la structure $\overline{\mathcal{K}}_{Ann}$ à utiliser pour la vérification, peut être construite en utilisant tout sous-groupe $\mathcal{H}$ tel que : $\mathcal{H} \subseteq \mathcal{G} \cap \mathcal{G}_{\mathcal{A}}$.

Il suffit alors de construire le produit synchronisé de $\overline{\mathcal{K}}_{Ann}$ et $\mathcal{A}$, suivant la définition 3.6, puis exécuter n'importe quel algorithme de test de vacuité [SE05, CDLP05, GV05] sur ce produit pour vérifier la propriété.

Définition 3.6 (Produit synchronisé de $\overline{\mathcal{K}}_{Ann}$ et $\mathcal{A}$)**.** *Le produit synchronisé* $\overline{\mathcal{K}}_{Ann} \otimes \mathcal{A} = (\mathcal{V}, \mathcal{V}_0, \mathcal{V}_{\mathcal{F}}, \mathcal{R})$ *est un graphe défini par :*
- $\mathcal{V} = \{(\overline{s}, q) \in \overline{\mathcal{S}} \times Q \mid \Pi(q) \subseteq \Pi(\overline{s})\}$ *est l'ensemble des nœuds,*
- $\mathcal{V}_0 = (\overline{\mathcal{S}}_0 \times Q_0) \cap \mathcal{V}$ *est l'ensemble des nœuds initiaux,*
- $\mathcal{V}_{\mathcal{F}} = (\overline{\mathcal{S}}, Q_{\mathcal{F}}) \cap \mathcal{V}$ *est l'ensemble des nœuds acceptants,*
- $\mathcal{R} \subseteq \mathcal{V} \times \mathcal{V}$ *est la relation de transition telle que* $(\overline{s}, q) \rightarrow (\overline{t}, r)$ *ssi :* $\overline{s} \xrightarrow{g} \overline{t}$ *et* $\Pi(r) = g^{-1}.\Pi(q')$ *où* $q \rightarrow q'$.

Note 3.3. *On remarque que la définition de la relation* $\mathcal{R}$ *autorise une transition entre les états* $(\overline{s}, q)$ *et* $(\overline{t}, r)$ *même dans le cas où* $q \rightarrow r$ *n'est pas une transition de*

l'automate de Büchi $\mathcal{A}$. En fait, ceci n'est autre que le représentant de la transition $(\overline{s},q) \rightarrow (g.\overline{t},q')$ avec $q \rightarrow q'$ et $g.\Pi(r) = \Pi(q')$. Cette astuce de représentation est primordiale dans l'approche pour ne garder dans le produit synchronisé que les états représentants et pas leurs symétriques.

Exemple 3.5. *Pour la vérification de la propriété exprimée par l'automate $\mathcal{A}$ de la Fig.3.4 sur le système concurrent de la Fig.3.1, il suffit de construire le produit synchronisé (suivant la définition précédente) entre la structure $\overline{\mathcal{K}}_{Ann}$ de la Fig. 3.5 et $\mathcal{A}$ car, dans ce cas particulier, nous avons $\mathcal{G}_{\mathcal{A}} = \mathcal{G}$ et donc $\mathcal{H} = \mathcal{G} \cap \mathcal{G}_{\mathcal{A}} = \mathcal{G}$.*

Bilan. Les approches présentées ici proposent un cadre formel, très précis, définissant l'exploitation des symétries pour la vérification de propriétés temporelles. Cependant, la pratique montre les limites de leur applicabilité dans un cadre général.

En effet, les cas où les systèmes et les propriétés étudiés présentent une large symétrie (*i.e.*, $|\mathcal{G}| \gg 1$, $|\mathcal{G}_f| \gg 1$ et $|\mathcal{G}_{\mathcal{A}}| \gg 1$) sont très rares. De plus, il faut que ces groupes soient "compatibles" entre eux (*i.e.*, présentent des éléments communs : $|\mathcal{H}| \gg 1$) pour obtenir une structure quotient de taille raisonnablement réduite.

La question qui reste donc posée est : que faire quand ces groupes sont proches de l'identité et que la réduction qu'ils apportent n'est pas significative ?

3.4 Exploitation des symétries faibles pour la vérification de propriétés temporelles

A la lumière de la section précédente, les deux approches que nous présentons dans cette section tentent d'optimiser l'exploitation des symétries en allégeant certaines contraintes. La première approche opère au niveau du système, en levant une contrainte sur la congruence des symétries entres les états les transitions. La deuxième approche s'attaque au problème de l'explosion combinatoire des états du point de vue de l'automate de Büchi. Elle apporte une vision locale des symétries de l'automate plutôt que de les considérer au niveau global (*i.e.*, celles de la définition 3.5).

3.4.1 Exploitation des symétries faibles du système

Les systèmes auxquels nous nous intéressons ici sont ceux dont le comportement est globalement symétrique (au sens de la définition 3.2) *excepté sur certains états*. Ces états ont la particularité d'être complètement symétriques, *i.e.*, l'application de n'importe quelle permutation sur les propositions atomiques étiquetant l'un de ces états donne le même état. En d'autres termes, la congruence de la relation d'équivalence, qui devait s'étendre vers les transitions est *partiellement levée*.

Exemple 3.6.

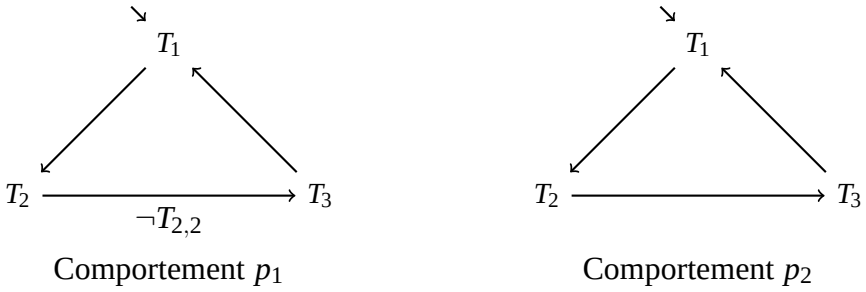


FIG. 3.6 – Exemple d'un système concurrent approximativement symétrique

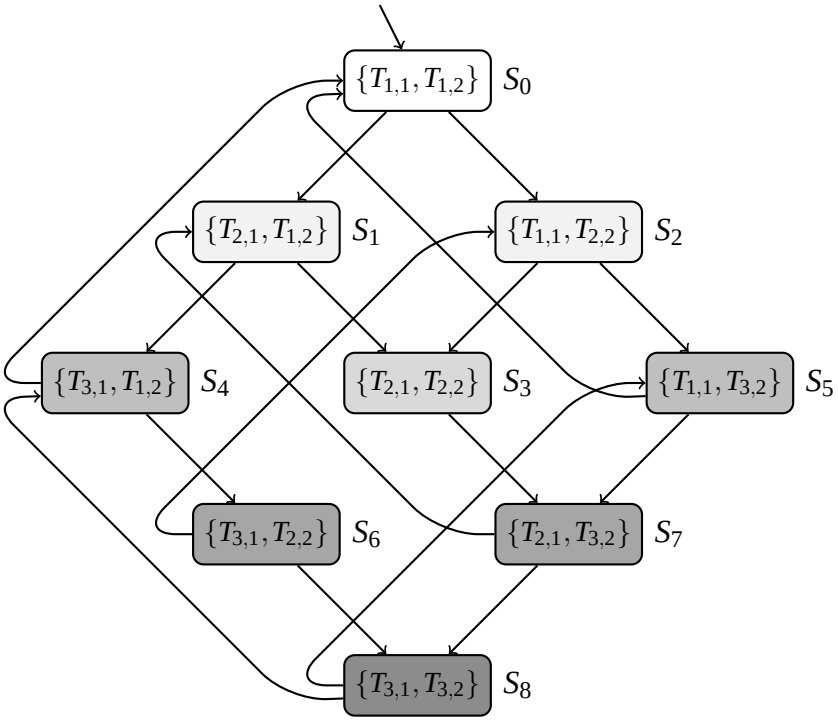


FIG. 3.7 – Structure de Kripke ($\mathcal{K}$) du système de la Fig.3.6.

Considérons le système de la Fig.3.6. Il est identique à celui présenté précédemment à l'exception du passage du processus p_1 de la tâche T_2 à T_3. En effet, pour effectuer cette transition il faut que le processus p_2 ne soit pas en train d'exécuter la tâche T_2. Le comportement global de ce système est représenté par la structure de Kripke de la Fig.3.7. Par rapport à l'exemple de la Fig. 3.1, nous remarquons qu'il manque la transition entre les états S_3 et S_6.

Ces systèmes sont dits *approximativement symétriques* et ils sont formalisés par la définition suivante.

Définition 3.7 (Structure de Kripke approximativement symétrique p.r.a $\mathcal{G}$)**.** *Soient $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathcal{AP}, \Pi \rangle$ une $\mathcal{SK}$ et $\mathcal{G} \subseteq \mathfrak{S}(\mathcal{AP})$. $\mathcal{K}$ est approximativement symétrique par rapport à $\mathcal{G}$ ssi :*
- *$\mathcal{G}.\mathcal{S} = \mathcal{S}$,*
- *Pour tout $s \in \mathcal{S}$, une des deux conditions suivantes est vérifiée :*
  - *Soit $\forall g \in \mathcal{G}$, $g.s = s$. Dans ce cas, s est dit totalement symétrique.*
  - *Soit $\forall g \in \mathcal{G}, \forall (s,t) \in \mathcal{R}, (g.s, g.t) \in \mathcal{R}$.*

Emerson et Trefler [ET99] prouvent qu'en exploitant les définitions 3.3 et 3.7, il est possible de construire un graphe quotient $\overline{\mathcal{K}}$ pour un système approximativement symétrique. Sur ce graphe, des formules de *LTL symétrique* (*SLTL*) peuvent être vérifiées. La syntaxe de *SLTL* est la même que celle de *LTL* sauf que les formules propositionnelles employées ne doivent pas faire référence à un objet (processus, processeur,...) particulier. Par exemple, la formule $G(\bigvee_i T_{3,i} \implies F(\bigvee_i T_{1,i}))$ peut être vérifiée sur une telle structure alors que $G(T_{3,1} \implies FT_{1,1})$ ne l'est pas.[6]

En pratique, le champ d'application de cette méthode est très réduit et ne dépasse pas les cas les plus élémentaires. Par exemple, on peut l'utiliser pour la vérification des algorithmes de lecture-écriture avec un seul lecteur et un seul écrivain (l'écrivain ayant une priorité par rapport au lecteur), mais elle est inapplicable dans les cas plus généraux (plus d'un lecteur, ou plus d'un écrivain), car ils ne vérifient pas les conditions de la définition 3.7. De plus, si la propriété à vérifier est asymétrique, on se retrouve dans la même situation que celle qui a amené cette approche, c'est-à-dire la construction d'une structure $\overline{\mathcal{K}}$ d'une taille très proche à celle de $\mathcal{K}$.

[6] On note qu'il est possible de généraliser cette méthode à LTL en la combinant avec l'approche syntaxique ou l'approche par automate.

3.4.2 Exploitation des symétries faibles d'un automate de Büchi

Les auteurs de [AHl98] se sont intéressés à la réduction apportée par l'exploitation des symétries faibles de l'automate de Büchi (représentant la propriété à vérifier). Ces symétries sont capturées au niveau *des suffixes* des chemins de l'automate de Büchi.

Considérons l'automate de la Fig. 3.8. Bien qu'il soit globalement asymétrique (par rapport à la définition 3.5 : $\mathcal{G}_{\mathcal{A}} = \{id\}$), nous pouvons constater que les suffixes $(q_2)^+(q_4)^+$ et $(q_3)^+(q_4)^+$ sont identiques à une permutation près (la permutation entre les processus p_1 et p_2).

Grâce à cette similarité, nous pouvons prédire la reconnaissance des exécutions du système en utilisant un seul représentant. En effet, soient deux états symétriques du système (ici la symétrie est au sens de la définition 3.2), alors si le futur de l'un de ces états est (non-)reconnu par un de ces suffixes, alors forcément le futur de l'autre état sera (non-)reconnu par l'autre chemin. Par conséquent, il suffit de représenter l'évolution à partir d'un des deux états.

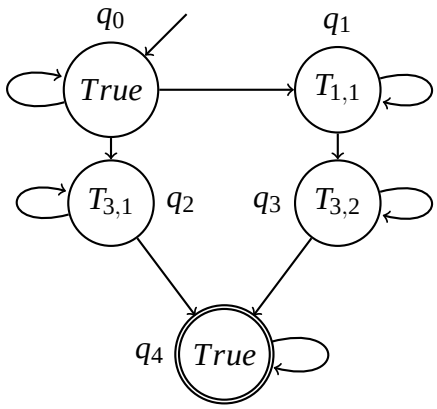


FIG. 3.8 – Automate de Büchi faiblement symétrique.

Cette idée a été formalisée par la définition d'une relation d'équivalence entre les états de l'automate comme suit.

Définition 3.8 (Équivalence entre deux états de l'automate de Büchi)**.** *Soit $\mathcal{A} = \langle Q, Q_0, Q_{\mathcal{F}}, \mathcal{R}, \mathcal{AP}, \Pi \rangle$, g une permutation sur $\mathcal{AP}$. $\sim_g$ est une relation d'équivalence entre $q, q' \in Q$, noté $q \sim_g q'$ défini par :*

– $q \in Q_{\mathcal{F}} \Leftrightarrow q' \in Q_{\mathcal{F}}$,
– $\Pi(q) = g.\Pi(q')$,
– $\forall q_d, q'_d \in Q$ *tels que* $q \rightarrow q_d$ *et* $q' \rightarrow q'_d$ *alors,* $q_d \sim_g q'_d$.

Soit une structure de Kripke $\mathcal{K}$ symétrique par rapport à un groupe $\mathcal{G}$. Il est alors possible de définir une relation de symétrie E_q sur le produit synchronisé $\mathcal{K} \otimes \mathcal{A}$ comme suit.

Définition 3.9 (Relation de symétrie, E_q, sur $\mathcal{K} \otimes \mathcal{A}$)**.** *Soit* $(s,q),(s',q') \in \mathcal{V}$. $(s,q)E_q(s',q')$ *ssi* $\exists g \in \mathcal{G}$ *tel que* $s = g.s'$ *et* $q \sim_g q'$.

Ici, E_q est une relation d'équivalence et par conséquent on peut définir une structure $\overline{\mathcal{K} \otimes \mathcal{A}}$ quotient de $\mathcal{K} \otimes \mathcal{A}$, sur laquelle la vérification peut être menée.

Définition 3.10 (Structure quotient $\overline{\mathcal{K} \otimes \mathcal{A}}$)**.** *La structure quotient* $\overline{\mathcal{K} \otimes \mathcal{A}}$ *est définie au moyen des représentants des classes d'équivalence des états de* $\mathcal{S}\mathcal{K} \otimes \mathcal{A}$. *La classe d'équivalence d'un état* $(s,q) \in \mathcal{V}$ *est définie par l'ensemble :*

$$[(s,q)] = \{(s',q') \in \mathcal{V} \mid (s,q)E_q(s',q')\}.$$

La structure quotient ainsi définie est la plus petite structure sur laquelle la vérification peut être effectuée. Dans le pire des cas, la construction nécessite un temps exponentiel. Les auteurs de cette approche proposent de réduire la complexité en construisant une structure intermédiaire, appelé *graphe consistant*, moins réduite que la structure quotient, mais qui ne nécessite qu'un temps polynomial pour sa construction.

Bilan. Les méthodes exploitant les symétries faibles offrent une alternative à celles par symétries globales pour la résolution du problème de l'explosion combinatoire. Cependant, leur applicabilité reste discutable. En effet, les systèmes faiblement symétriques se limitent à des cas d'école et le principe de ces méthodes est difficilement généralisable. En outre, l'utilisation des automates faiblement symétriques, bien qu'atténuant la complexité du problème, reste insuffisante pour le cas où le système est globalement asymétrique.

3.5 Exploitation des symétries partielles

Contrairement aux symétries globales, les symétries partielles ne sont valides que sur des parties du système ou de la propriété à vérifier. Mais c'est souvent largement suffisant pour construire des structures quotient de taille acceptable et sur

lesquelles la vérification peut s'effectuer efficacement.

Nous présenterons ici l'une des premières techniques basées sur le principe de l'exploitation des symétries partielles. Elle a été développée dans [HITZ95, Cap00], dans le cadre du formalisme des réseaux de Petri bien-formés : *Le graphe de marquages symboliques étendus (ESRG)*. Elle permet la vérification des propriétés de base telles que l'accessibilité, la présence de verrous ou l'existence de chemins infinis. [7]

3.5.1 Symétries partielles et accessibilité

Dans la pratique, on constate que les comportements de certains systèmes concurrents sont caractérisables par une association de comportements *totalement symétriques* et de comportements *asymétriques occasionnels*. L'existence de ces comportements asymétriques fait que les symétries globales de la $\mathcal{SK}$ (représentant le comportement du système) sont souvent très réduites (*i.e.*, $|\mathcal{G}| \backsimeq 1$).[8] Cependant, la représentation des comportements symétriques peut être factorisée d'une manière optimale.

Exemple 3.7.

Comportement p_1 Comportement p_2

FIG. 3.9 – Exemple d'un système partiellement symétrique.

La Fig.3.9 montre un exemple d'un système globalement asymétrique. Cette asymétrie est due à l'exécution de la tâche T_4 par le processus p_2 uniquement, tandis que pour le reste du système, les deux processus se comportent d'une manière identique. La structure de Kripke de ce système est représentée dans la Fig.3.10.

[7] À notre connaissance, aucune tentative n'a été faite pour utiliser cette structure pour la vérification de propriétés temporelles.

[8] Les symétries ici sont comprises au sens de la définition 3.2.

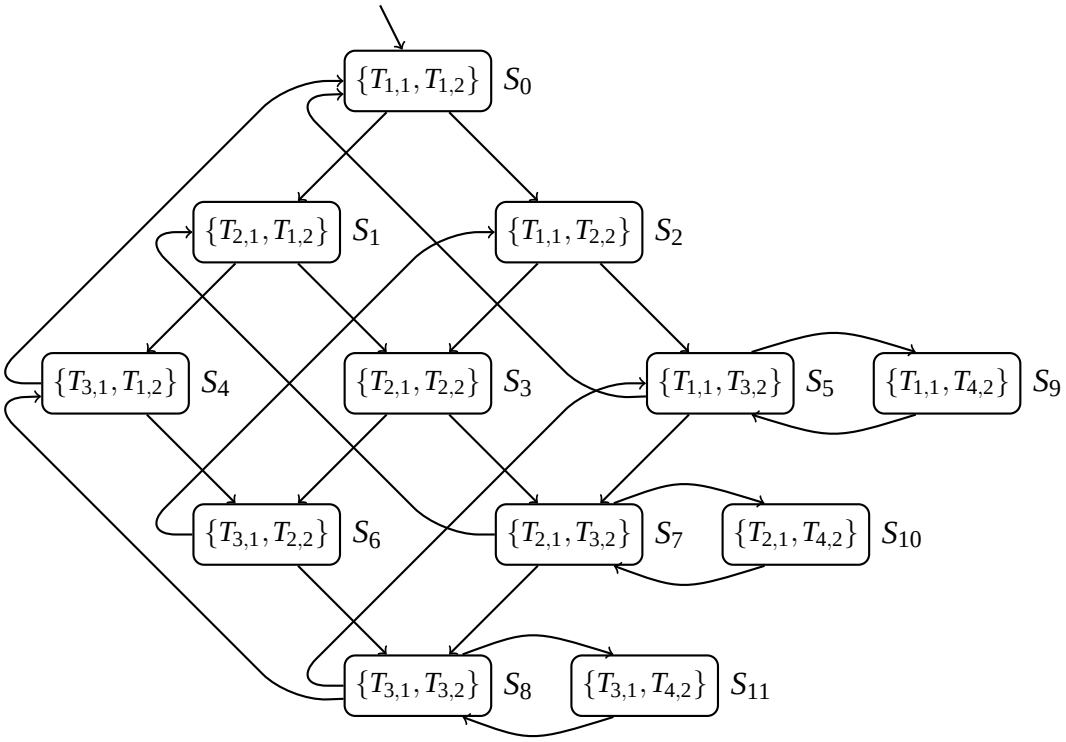


FIG. 3.10 – Structure de Kripke, $\mathcal{K}$, du système de la Fig.3.9.

Les asymétries occasionnelles sont formalisées par la notion de structure de Kripke *partiellement symétrique*.

Définition 3.11 (Structure de Kripke partiellement symétrique)**.** *Soient* $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathcal{AP}, \Pi \rangle$ *une* $\mathcal{SK}$ *et* $\mathcal{G}_s \subseteq \mathfrak{S}(\mathcal{AP})$. $\mathcal{K}$ *est dite partiellement symétrique par rapport à* $\mathcal{G}_s$ *ssi il existe une sous structure de Kripke,* $\mathcal{K}_s = \langle \mathcal{S}^s, \mathcal{S}_0^s, \mathcal{R}^s, \mathcal{AP}, \Pi \rangle$*, qui vérifie :*
– $\mathcal{S}^s \subseteq \mathcal{S}$, $\mathcal{S}_0^s \subseteq \mathcal{S}^s \cap \mathcal{S}_0$, $\mathcal{R}^s \subseteq \mathcal{R}$ *et*
– $\mathcal{K}_s$ *est symétrique par rapport à* $\mathcal{G}_s$.

Exemple 3.8. *La structure* $\mathcal{K}$ *de la Fig.3.10 est partiellement symétrique par rapport au sous-groupe* $\mathcal{G}_s \subseteq \mathfrak{S}(\mathcal{AP})$ *tel que* $\mathcal{G}_s = \{id, g\}$ *où* g *est défini par* $g.T_{i,1} = T_{i,2}$ *et* $g.T_{i,2} = T_{i,1}$, $i \in \{1,2,3\}$.

En effet, il suffit d'ignorer les états S_9, S_{10} *et* S_{11}, *ainsi que leurs arcs adjacents, pour découvrir une structure* $\mathcal{K}_s$ *symétrique par rapport à* $\mathcal{G}_s$.

Ainsi, pour une structure $\mathcal{K}$, le sous-groupe $\mathcal{G}_s$ permet de réduire au maximum la représentation de $\mathcal{K}_s$. Le graphe quotient $\overline{\mathcal{K}_s}$ de $\mathcal{K}_s$ est construit suivant la définition 3.3, en prenant $\mathcal{H} = \mathcal{G}_s$. Dans ce cas, le sous-groupe $\mathcal{G}$ est utilisé uniquement pour réduire la représentation des parties asymétriques (il est applicable partout).[9] Ceci est formalisé par la définition de la structure quotient suivante.

Définition 3.12 (Structure quotient d'une $\mathcal{SK}$ partiellement symétrique)**.** *Soit $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathcal{AP}, \Pi \rangle$ une $\mathcal{SK}$ symétrique par rapport à $\mathcal{G}$ et partiellement symétrique par rapport à $\mathcal{G}_s$ et $\mathcal{K}_s = \langle \mathcal{S}^s, \mathcal{S}_0^s, \mathcal{R}^s, \mathcal{AP}, \Pi \rangle$ sa sous-structure. La structure quotient de $\mathcal{K}$ est un graphe $\overline{\mathcal{K}} = \langle \overline{\mathcal{S}}, \overline{\mathcal{R}} \rangle$ définie par :*

$$
\begin{aligned}
-\ \overline{\mathcal{S}} = \ & \{(\mathcal{G}_s, \overline{s}) \mid \overline{s} \in \mathcal{S}^s, \overline{s} \textit{ est un représentant unique de la classe d'équivalence } \mathcal{G}_s.\overline{s}\} \cup \\
& \{(\mathcal{G}, \overline{s}) \mid \overline{s} \in \mathcal{S}, \overline{s} \textit{ est un représentant unique de la classe d'équivalence } \mathcal{G}.\overline{s}\} \setminus \\
& \{(\mathcal{G}, \overline{s}) \mid \overline{s} \in \mathcal{S}, \exists \overline{s}' \in \mathcal{S}^s, \mathcal{G}_s.\overline{s}' = \mathcal{G}.\overline{s}\} \\
-\ \overline{\mathcal{R}} = \ & \{((\mathcal{G}_s, \overline{s}_1), (\mathcal{G}_s, \overline{s}_2)) \mid \exists s_1 \in \mathcal{G}_s.\overline{s}_1, \exists s_2 \in \mathcal{G}_s.\overline{s}_2, (s_1, s_2) \in \mathcal{R}^s\} \cup \\
& \{((\mathcal{G}, \overline{s}_1), (\mathcal{G}, \overline{s}_2)) \mid \exists s_1 \in \mathcal{G}.\overline{s}_1, \exists s_2 \in \mathcal{G}.\overline{s}_2, (s_1, s_2) \in (\mathcal{R} \setminus \mathcal{R}^s)\} \cup \\
& \{((\mathcal{G}, \overline{s}_1), (\mathcal{G}_s, \overline{s}_2)) \mid (\mathcal{G}_s.\overline{s}_2 = \mathcal{G}.\overline{s}_2) \wedge \\
& (\exists s_1 \in \mathcal{G}.\overline{s}_1, \exists s_2 \in \mathcal{G}.\overline{s}_2, (s_1, s_2) \in (\mathcal{R} \setminus \mathcal{R}^s))\}.
\end{aligned}
$$

Dans la construction de la structure quotient, il faut porter une attention particulière aux classes d'équivalence construites à partir de $\mathcal{G}_s$ ou de $\mathcal{G}$ quand elles représentent exactement le même ensemble d'états. Dans ce cas, on ne garde que la classe d'équivalence construite par $\mathcal{G}_s$ (dernière clause dans la définition de $\overline{\mathcal{S}}$). Par conséquent, les arcs pointant vers les nœuds retirés doivent être redirigés vers les nœuds sauvegardés (dernière clause de la définition de $\overline{\mathcal{R}}$).

Exemple 3.9. *Le graphe quotient $\overline{\mathcal{K}_s}$ de $\mathcal{K}_s$ est représenté en gras dans la Fig. 3.11. Les nœuds (resp. les arcs) de $\overline{\mathcal{K}_s}$ sont des classes d'équivalence d'états (resp. d'arcs) par rapport à $\mathcal{G}_s$. Nous avons les classes d'équivalences suivantes : $\{S_0\}, \{S_1, S_2\}, \{S_3\}, \{S_4, S_5\}, \{S_6, S_7\}, \{S_8\}$. Aussi, nous avons choisi les représentant suivants : $\overline{S}_0 = S_0$, $\overline{S}_1 = S_2$, $\overline{S}_2 = S_3$, $\overline{S}_3 = S_4$, $\overline{S}_4 = S_6$ et $\overline{S}_5 = S_8$.*

Pour retrouver l'ensemble des comportements du système, nous devons réintroduire les états et les franchissements précédemment ignorés. Ceci est effectué en isolant, à l'intérieur de chaque classe d'états de $\overline{\mathcal{K}_s}$, les états prédécesseurs et successeurs des états ignorés (considérer les traits en pointillé).

Ceci concerne tous les états dont le sous-groupe est $\mathcal{G}$ et leurs arcs adjacents : les nœuds (resp. les arcs) sont des classes d'équivalence de nœuds (resp. d'arcs) par

[9] $\mathcal{K}$ est symétrique par rapport à $\mathcal{G}$.

rapport à $\mathcal{G}$ (ici, $\mathcal{G}$ est réduit à l'identité) : $\overline{S}'_3 = S_5$ et $\overline{S}'_4 = S_7$.

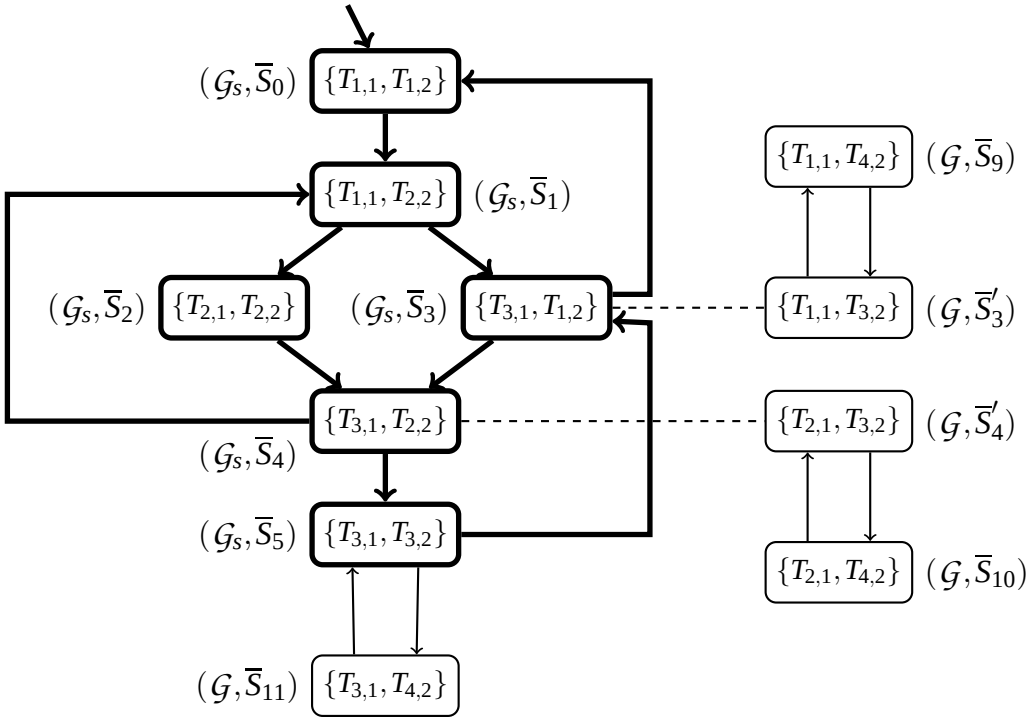


FIG. 3.11 – Structure quotient, $\overline{\mathcal{K}}$, de $\mathcal{K}$ (de la Fig.3.10).

Nous remarquons ici, que les classes d'équivalence associées à l'état S_8 et calculées par rapport à $\mathcal{G}_s$ et $\mathcal{G}$ sont les mêmes : $\mathcal{G}_s.S_8 = \mathcal{G}.S_8 = \{S_8\}$. Ceci explique les arcs entre les nœuds $(\mathcal{G}_s, \overline{S}_5)$ et $(\mathcal{G}, \overline{S}_{11})$.

Ainsi, nous avons construit un graphe quotient $\overline{\mathcal{K}}$ de $\mathcal{K}$ dont les nœuds et les arcs sont divisés en deux ensembles :
- *le premier ensemble représente d'une manière optimisée l'ensemble des comportements symétriques moyennant le sous-groupe de permutations $\mathcal{G}_s$.*
- *le deuxième ensemble représente l'ensemble des comportements asymétriques moyennant le sous-groupe de permutations restrictif $\mathcal{G}$.*

Bilan. La technique par symétries partielles, développée ici, permet de traiter les systèmes globalement asymétriques de manière efficace. Néanmoins, cette efficacité dépend fortement de la taille du sous-système symétrique $\mathcal{K}_s$: plus $\mathcal{K}_s$ est grand, plus la réduction est importante. Aussi, cette technique ne traite pas des

propriétés temporelles et son application est réduite au cas de l'accessibilité.

Cependant, elle est l'initiatrice des idées qui vont suivre pour l'exploitation des symétries partielles dans un cadre plus général. Ceci nous conduit à l'approche [HIA00], destinée cette fois à la vérification des propriétés temporelles. Elle a été le point de départ de cette thèse car elle offrait un large potentiel d'optimisation non encore exploité. Vue son importance, nous lui consacrons le prochain chapitre ainsi qu'aux améliorations que nous y apportons.

Chapitre 4

Symétries locales et vérification de propriétés temporelles

Les techniques décrites précédemment, exploitant les symétries des systèmes, proposent un schéma dans lequel la propriété est structurellement symétrique.

Les auteurs de [HIA00] proposent une technique dans laquelle il est possible non seulement de traiter ces cas, mais aussi ceux où les propriétés sont *asymétriques*. De plus, et moyennant une représentation particulière des asymétries du système, cette méthode reste applicable pour les systèmes *asymétriques*.

4.1 Présentation de la méthode originelle

L'approche présentée dans [HIA00] est basée sur la théorie des automates et opère au niveau du produit synchronisé entre la $\mathcal{SK}$ du système et le $\mathcal{BA}$ représentant la propriété à vérifier.

Durant le processus de vérification, il arrive souvent que des exécutions équivalentes soient reconnues *partiellement* par le même chemin de l'automate. En d'autres termes, *des préfixes* de ces exécutions sont reconnus par un *préfixe* du chemin. Pour ce chemin, les représentations de ces exécutions peuvent être factorisées *au niveau de ces préfixes,* et la distinction ne sera utile que lors de la divergence dans la reconnaissance.

Exemple 4.1. *Considérons l'automate de Büchi de la formule LTL,* $f = FG(T_{2,1} \wedge T_{3,2})$ *(Fig. 4.1). Deux chemins du produit synchronisé entre la* $\mathcal{SK}$ *de la Fig. 3.2, représentant les états du système symétrique à deux processus, et l'automate* $\mathcal{A}_f$ *sont représentés dans la Fig.4.2.*

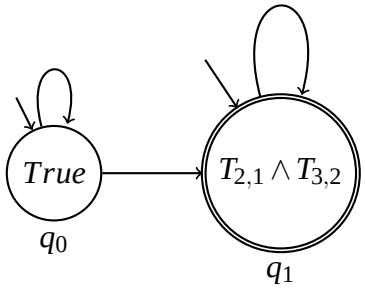


FIG. 4.1 – Automate de Büchi $\mathcal{A}_f$ de la formule LTL : $f = FG(T_{2,1} \wedge T_{3,2})$.

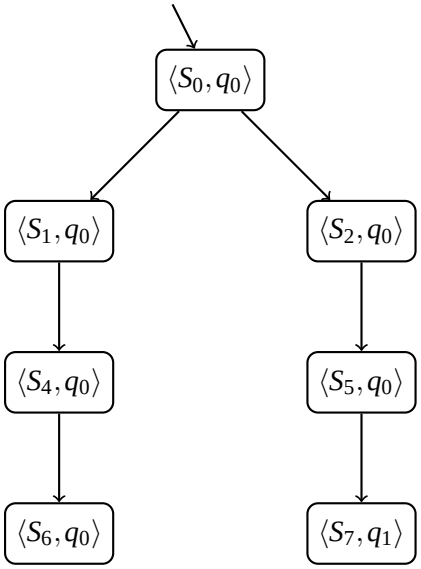


FIG. 4.2 – Une partie du produit synchronisé entre la $\mathcal{SK}$, $\mathcal{K}$, de la Fig. 3.2 et $\mathcal{A}_f$

Nous remarquons que dans les exécutions $\sigma = \langle S_0, S_1 \rangle \langle S_1, S_4 \rangle \langle S_4, S_6 \rangle \ldots$ *et* $\sigma' = \langle S_0, S_2 \rangle \langle S_2, S_5 \rangle \langle S_5, S_7 \rangle \ldots$*, les préfixes* $\langle S_0, S_1 \rangle \langle S_1, S_4 \rangle$ *de* σ *et* $\langle S_0, S_2 \rangle \langle S_2, S_5 \rangle$ *de* σ' *sont reconnus par le même préfixe* $\langle q_0, q_0 \rangle \langle q_0, q_0 \rangle$ *de l'automate. La divergence dans la reconnaissance ne commence que lorsque les états* S_6 *dans* σ *et* S_7 *dans* σ' *sont rencontrés.*

Ainsi, puisque les exécutions σ *et* σ' *sont symétriques par rapport à une permutation de* $\mathcal{G}$[1]*, il est possible de factoriser la représentation des préfixes des deux chemins.*

[1] $\mathcal{G}$ est un groupe de symétries sur $\mathcal{K}$.

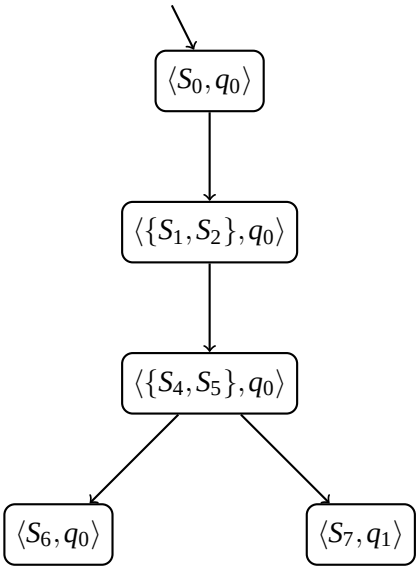


FIG. 4.3 – Factorisation des représentations.

Cette idée est exploitée pour la définition d'un produit synchronisé quotient, $\overline{\mathcal{K} \otimes \mathcal{A}}$, représentant le produit ordinaire $\mathcal{K} \otimes \mathcal{A}$ ($\mathcal{K}$ étant la $\mathcal{SK}$ représentant le comportement du système et $\mathcal{A}$, le $\mathcal{BA}$ representant la propriété à verifier). Du point de vue du test de vacuité, les deux structures sont prouvées équivalentes. En effet, l'existence d'un chemin acceptant dans la structure ordinaire est une condition nécessaire et suffisante pour l'existence d'chemin acceptant dans la structure quotient.

Définition 4.1 (Produit synchronisé quotient)**.** *Soit $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}_\mathcal{K}, \mathcal{AP}, \Pi_\mathcal{K} \rangle$ une $\mathcal{SK}$ symétrique p.r.a un groupe $\mathcal{G} \subseteq \mathfrak{S}(\mathcal{AP})$, et $\mathcal{A} = \langle Q, Q_0, Q_\mathcal{F}, \mathcal{R}_\mathcal{A}, \overline{\mathcal{AP}}, \Pi_\mathcal{A} \rangle$ un $\mathcal{BA}$. Le* produit synchronisé quotient *de $\mathcal{K}$ et $\mathcal{A}$ est un $\mathcal{BA}$, $\overline{\mathcal{K} \otimes \mathcal{A}} = \langle \mathcal{V}, \mathcal{V}_0, \mathcal{V}_\mathcal{F}, \mathcal{R}, \mathcal{AP}, \Pi \rangle$ défini par :*

- $\mathcal{V} = \{\langle \mathcal{H}, O, q \rangle | \mathcal{H} \subseteq \mathcal{G}, O \subseteq \text{Reach}(\mathcal{K}), q \in Q \wedge \mathcal{H}.O = O\}$,
- $\mathcal{V}_0 = \{\langle \mathcal{G}, \mathcal{G}.s, q \rangle \mid s \in \mathcal{S}_0 \wedge q \in Q_0\}$,
- $\mathcal{V}_\mathcal{F} = \{\langle \mathcal{H}, O, q \rangle \in \mathcal{V} \mid q \in Q_\mathcal{F}\}$,
- $\mathcal{R}$ *est défini par construction comme suit :* $\big\langle \langle \mathcal{H}, O, q \rangle, \langle \mathcal{H}', O', q' \rangle \big\rangle \in \mathcal{R}$ *ssi* $\exists (s, s') \in O \times \mathcal{S}$ *tel que* $\langle s, s' \rangle \in \mathcal{R}_\mathcal{K} \wedge \langle q, q' \rangle \in \mathcal{R}_\mathcal{A} \wedge \Pi_\mathcal{A}(q) \subseteq \Pi_\mathcal{K}(s)$. *Alors* $O' = (\mathcal{H} \cap \mathcal{G}_q).s'$ *et* $\mathcal{H}' \subseteq \mathcal{G}_{O'}$. [2]
- $\Pi : \mathcal{V} \to 2^{\mathcal{AP}}$, *telle que* $\Pi(\langle \mathcal{H}, O, q \rangle) = \Pi_\mathcal{A}(q)$.

[2] $\mathcal{G}_q$ est une simplification d'écriture pour $\mathcal{G}_{\Pi_\mathcal{A}(q)}$.

Note 4.1. *Pour rendre la relation successeur opérationnelle, l'utilisateur de l'approche doit spécifier comment choisir le sous-groupe $\mathcal{H}'$. Il est a priori intéressant de maximiser $\mathcal{H}'$, i.e., prendre $\mathcal{H}' = \mathcal{G}_{O'}$, mais ce choix peut s'avérer extrêmement coûteux si on ne peut pas trouver $\mathcal{G}_{O'}$ directement à partir de la syntaxe du modèle utilisé. Dans tous les cas, $\mathcal{H}'$ peut être choisi comme étant le sous-groupe $\mathcal{H} \cap \mathcal{G}_q \subseteq \mathcal{G}_{O'}$.*

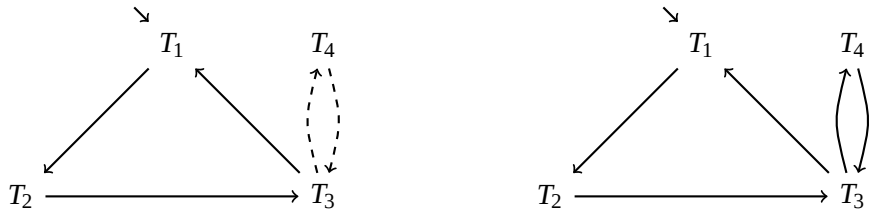
Le constat important ici est que la définition précédente ne pose aucune contrainte sur les symétries globales de l'automate de Büchi, ce qui permet d'accepter n'importe quel automate, quel que soit son degré d'asymétrie. Cependant, la réduction apportée par l'approche dépend fortement de la taille du sous-groupe $\mathcal{G}$, car chaque nœud de la structure quotient référence un groupe $\mathcal{H}$ qui est lui-même un sous-groupe de $\mathcal{G}$ ($|\mathcal{G}| \simeq 1 \implies |\mathcal{H}| \simeq 1$).

Ainsi, pour espérer une réduction importante (même dans le cas des systèmes globalement asymétriques), les auteurs de l'approche proposent de procéder de la manière suivante : d'abord il faut augmenter les différents composants (de même type), du système asymétrique, par des actions et des tâches, de manière à unifier leurs comportements ; ensuite, un $\mathcal{BA}$ particulier, appelé *automate de contrôle*, est utilisé pour restreindre les comportements du système, ainsi modélisé, à ceux réellement exécutés par le système originel.

Note 4.2. *Ici, il n'est pas question de traiter n'importe quel système asymétrique (bien que cela soit théoriquement possible), mais seulement ceux dont les composants exhibent une certaine forme de symétrie, que nous qualifierons de systèmes partiellement symétriques.*[3] *Cette propriété est courante dans les systèmes distribués et un bon exemple est celui des processus concurrents qui veulent acceder à une section critique distribuée. La gestion des conflits est faite, par exemple, en attribuant une priorité à chaque processus. Dans ce cas, tous les processus se comportent d'une manière symétrique sauf pour l'accès à la section.*

Exemple 4.2. *Considérons à nouveau le système de la Fig. 3.9. Pour faire correspondre le comportement de p_1 à celui de p_2, nous rajoutons la tâche T_4 à p_1 comme illustré dans la Fig.4.4. Cet ajout transforme le comportement global du système en celui représenté par la structure de Kripke, $\mathcal{K}$, de la Fig.4.5.*

[3] Dans le reste du document, nous utiliserons indifféremment les termes *globalement asymétriques* et *partiellement symétriques*.



Comportement p_1 Comportement p_2

FIG. 4.4 – Ajout de la tâche T_4 à p_1.

En comparant $\mathcal{K}$ à la structure représentée dans la Fig. 3.10, nous remarquons l'ajout des trois états : S_{12}, S_{13} et S_{14}. Ces états, ainsi que leurs arcs adjacents, garantissent la symétrie de $\mathcal{K}$ par rapport au sous-groupe $\mathcal{G} \subseteq \mathfrak{S}(\mathcal{AP})$ tel que $\mathcal{G} = \{id, g\}$ où g est défini par $g.T_{i,1} = T_{i,2}$ et $g.T_{i,2} = T_{i,1}$, $i \in \{1,2,3\}$.

L'automate de contrôle $\mathcal{A}_c$ est utilisé pour interdire le passage du système d'un état où le processus p_1 exécutait la tâche T_3 vers celui où il exécute la tâche T_4. Ainsi, le comportement réel du système est obtenu par la synchronisation de la structure $\mathcal{K}$ avec $\mathcal{A}_c$: $\mathcal{K} \otimes \mathcal{A}_c$.

En réalité, ce procédé introduit une nouvelle approche de modélisation des systèmes globalement asymétriques, qui se trouve être adaptée à la gestion *locale et dynamique* des asymétries. Dans cette modélisation, les asymétries du système sont séparées du comportement *nominal* par le biais de l'automate de contrôle. Par conséquent, le problème de la vérification d'un système (globalement asymétrique) est ramené à la vérification d'un *système symétrique contrôlé* : $\mathcal{K} \otimes \mathcal{A}_c$ étant le représentant du comportement du système asymétrique et $\mathcal{A}_f$, l'automate de Büchi représentant la propriété à vérifier, alors le produit synchronisé sur lequel le test de vacuité va s'exécuter est défini par : $(\mathcal{K} \otimes \mathcal{A}_c) \otimes \mathcal{A}_f$. Puisque l'opération $\otimes$ est associative, $(\mathcal{K} \otimes \mathcal{A}_c) \otimes \mathcal{A}_f = \mathcal{K} \otimes (\mathcal{A}_c \otimes \mathcal{A}_f)$. En notant $\mathcal{A}_{c,f} = \mathcal{A}_c \otimes \mathcal{A}_f$ alors $\mathcal{K} \otimes (\mathcal{A}_c \otimes \mathcal{A}_f) = \mathcal{K} \otimes \mathcal{A}_{c,f}$, et le problème est ramené à la vérification d'un système symétrique par rapport à un automate quelconque.

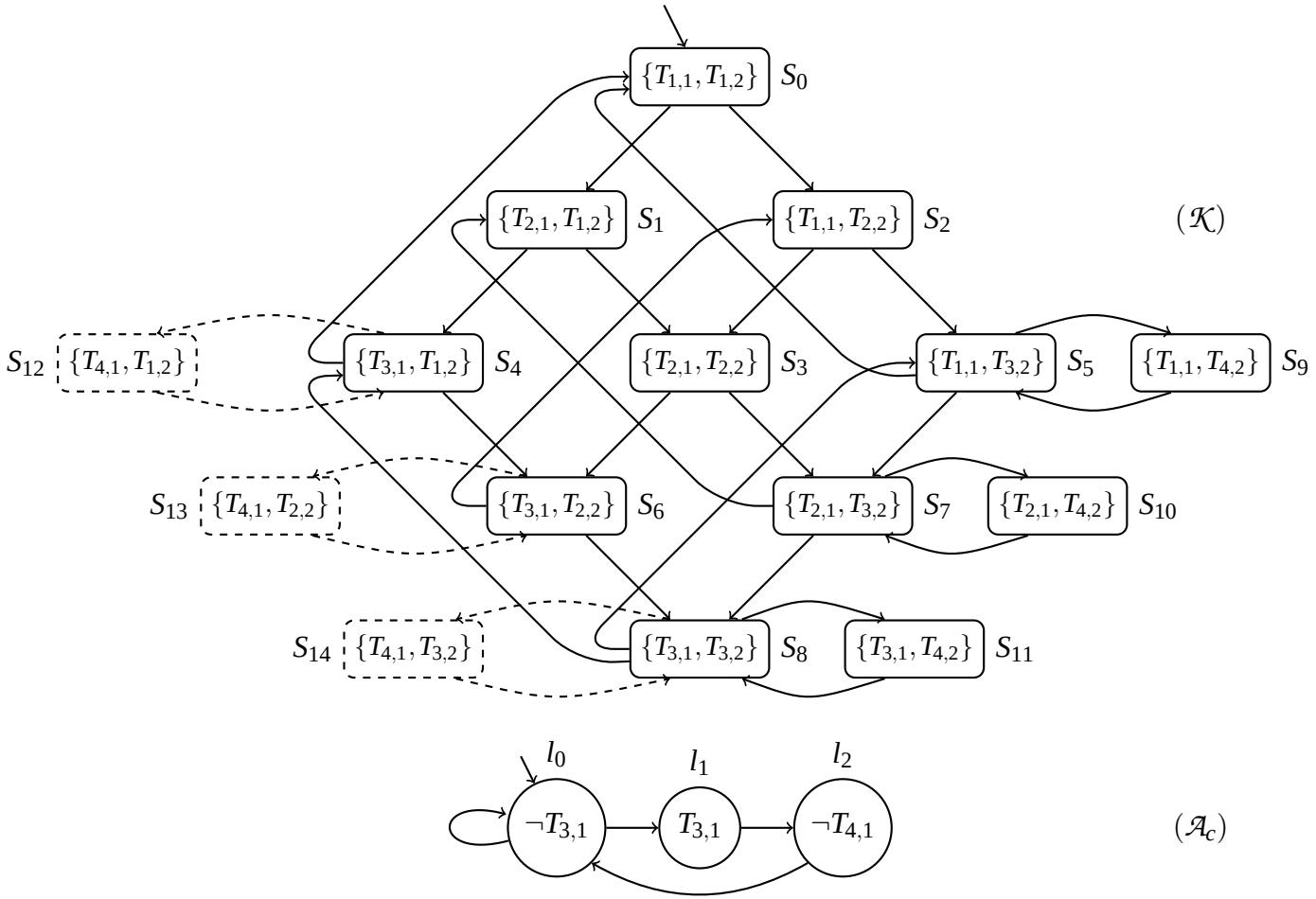


FIG. 4.5 – Structure de Kripke, $\mathcal{K}$, du système de la Fig.4.4 et son automate de contrôle ($\mathcal{A}_c$).

4.2 Critiques de la méthode originelle

L'avantage de l'approche présentée est, bien sûr, sa capacité à gérer localement et dynamiquement les (a)symétries du système et de la propriété à vérifier, même pour les cas les plus désavantageux (*i.e.*, système et propriété globalement asymétriques). Elle offre ainsi la possibilité d'une optimisation du processus de vérification. Cependant, cette même localité de traitement conduit à une complexité[4], en espace, du produit synchronisé quotient de $2^{(|\mathcal{K}|\times|A_c|\times|A_f|)}$. En d'autres termes, il est possible de construire une structure quotient plus grande que le produit ordinaire !

Ainsi, et bénéficiant de l'idée de base, nous nous proposons d'améliorer la méthode afin de minimiser l'impact de cette complexité. Ceci est réalisé en deux points :

– le premier point concerne la modélisation. Nous proposons une modélisation utilisant les *automates de contrôle basés sur les événements* au lieu de ceux utilisés dans la méthode originelle : d'une part, parce que le raisonnement sur les événements, pour contrôler les actions du système, est plus intuitif que celui basé sur l'énumération des états ; et d'autre part (et c'est le plus important), parce que l'automate généré par un tel raisonnement est (en pratique) plus compact. Pour l'expression de la propriété, nous utiliserons les automates de Büchi généralisés basés sur les transitions ($\mathcal{TGBA}$), car il sont plus compacts que les $\mathcal{BA}$ classiques.

– Le deuxième point est plus fondamental car il concerne l'algorithme de test de vacuité. Nous développerons une approche qui bénéficie de la structuration en ensembles des états du produit quotient pour optimiser la vérification. En effet, les états de la structure quotient étant des ensembles d'états du produit ordinaire, ils peuvent être comparés par des *tests d'inclusion* en plus des tests d'égalité classiques. Cette possibilité permet de réduire la taille globale de la structure. L'approche sera décrite dans le chapitre suivant dans le cadre général des *automates de Büchi ensemblistes*.

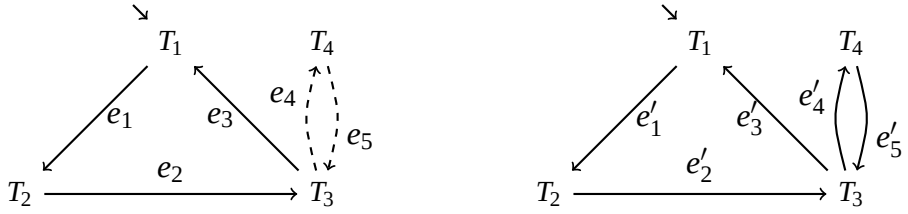
[4] Complexité dans le pire des cas.

4.3 Modèle de système partiellement symétrique [5]

Compte tenu des critiques précédentes, nous proposons d'améliorer l'approche en conservant l'idée du contrôle d'un système symétrique, mais en utilisant cette fois un automate de contrôle basé sur les événements. En effet, il est plus facile et plus naturel d'interdire au système (symétrique) d'atteindre un état particulier moyennant la "suppression" de certaines actions, que d'exprimer l'énumération de toutes les exécutions admises. En plus d'être source d'erreurs de modélisation, un automate de contrôle basé sur l'énumération des sequences d'états non admises comporte souvent un nombre important d'états et ceci pose un autre problème de complexité pour la vérification.

Exemple 4.3. *Considérons à nouveau le système de la Fig. 4.4. Nous augmentons sa représentation par l'ensemble d'événements* $\mathcal{E} = \{e_i\}_{i=1,\ldots,5} \cup \{e'_i\}_{i=1,\ldots,5}$. *Chaque événement est interprété par la terminaison d'une tâche et le déclenchement de l'exécution d'une autre.*

Pour retrouver les comportements du système originel (i.e., sans l'exécution de T_4 *par* p_1*), il suffit de contrôler l'occurrence de l'événement* e_4. *Ceci est facilement réalisable par l'automate de contrôle de la Fig.4.7. L'étiquette* $\neg e_4$ *est définie par* $\neg e_4 = \mathcal{E} \setminus \{e_4\}$. *Cet automate accepte toutes les séquences événements sauf celles incluant* e_4.



Comportement p_1 Comportement p_2

FIG. 4.6 – Exemple d'un système étiqueté.

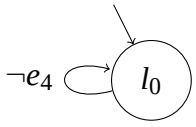


FIG. 4.7 – Automate de contrôle, $\mathcal{A}_c$, basé sur les événements.

[5] Modèle introduit pour la première fois dans [BHI04].

On note immédiatement que cet automate compte moins d'états et d'arcs que celui de la Fig. 4.5 (1 état contre 3).

Avant de formaliser l'idée précédente par la notion de *structure de Kripke contrôlée*, nous introduisons la définition d'un automate de contrôle basé sur les événements.

Définition 4.2 (Automate de contrôle ($\mathcal{CA}$))**.** *Un automate de contrôle* $\mathcal{A}_c = \langle L, l_0, \mathcal{R}, \mathcal{E}, \Delta \rangle$ *est défini par :*
- L*, est l'ensemble des états,*
- $l_0 \in L$*, est l'état initial,*
- $\mathcal{R} \subseteq L \times L$ *est un ensemble de transitions.*
- $\mathcal{E}$ *est un ensemble fini d'événements.*
- $\Delta : L \times L \rightarrow 2^{\mathcal{E}}$ *est une application qui associe à chaque couple d'états un ensemble d'événements.*

On note $(l, \gamma, l') \in \mathcal{R}$ *(ou encore* $l \xrightarrow{\gamma} l'$*), la transition* $(l, l') \in \mathcal{R}$ *telle que* $\Delta(l, l') = \gamma$*.*

Ainsi, le contrôle d'une structure de Kripke étiquetée par un automate consiste à synchroniser chacune de ses transitions avec une transition de l'automate. Autrement dit, le passage du système d'un état vers un autre ne sera autorisé que si l'événement qui le provoque est validé par une transition de l'automate.

Définition 4.3 ($\mathcal{SKE}$ contrôlée par un $\mathcal{CA}$)**.** *Soit* $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathcal{AP}, \mathcal{E}, \Pi, \Gamma \rangle$ *une* $\mathcal{SKE}$ *et* $\mathcal{A}_c = \langle L, l_0, \mathcal{R}_{\mathcal{A}_c}, \mathcal{E}, \Delta \rangle$ *un* $\mathcal{CA}$*. Le contrôle de* $\mathcal{K}$ *par* $\mathcal{A}_c$ *est défini par une* $\mathcal{SKE}$*,* $\mathcal{K}_{\mathcal{A}_c} = \langle \mathcal{S}_{\mathcal{A}_c}, \mathcal{S}^0_{\mathcal{A}_c}, \mathcal{R}^{\mathcal{K}}_{\mathcal{A}_c}, \mathcal{AP}, \mathcal{E}, \Pi^{\mathcal{K}}_{\mathcal{A}_c}, \Gamma^{\mathcal{K}}_{\mathcal{A}_c} \rangle$*, notée* $\mathcal{K}_{\mathcal{A}_c} = \mathcal{K} \overline{\otimes} \mathcal{A}_c$*, tel que :*
- $\mathcal{S}_{\mathcal{A}_c} = \mathcal{S} \times L$ *est l'ensemble des états,*
- $\mathcal{S}^0_{\mathcal{A}_c} = \mathcal{S}_0 \times \{l_0\}$ *est l'ensemble des états initiaux,*
- $\mathcal{R}^{\mathcal{K}}_{\mathcal{A}_c} = \{\langle (s,l), (s',l') \rangle \in \mathcal{S}_{\mathcal{A}_c} \times \mathcal{S}_{\mathcal{A}_c} \mid \langle s, s' \rangle \in \mathcal{R} \wedge \langle l, l' \rangle \in \mathcal{R}_{\mathcal{A}_c} \wedge \Gamma(s,s') \in \Delta(l,l')\}$ *est la relation de transition.*
- $\Pi^{\mathcal{K}}_{\mathcal{A}_c} : \mathcal{S}_{\mathcal{A}_c} \rightarrow 2^{\mathcal{AP}}$ *telle que* $\Pi^{\mathcal{K}}_{\mathcal{A}_c}(s,l) = \Pi(s)$*,*
- $\Gamma^{\mathcal{K}}_{\mathcal{A}_c} : \mathcal{S}_{\mathcal{A}_c} \times \mathcal{S}_{\mathcal{A}_c} \rightarrow 2^{\mathcal{E}}$ *telle que* $\Gamma^{\mathcal{K}}_{\mathcal{A}_c}((s,l),(s',l')) = \Gamma(s,s')$*.*

La vérification d'une propriété temporelle à temps linéaire, exprimée par une formule LTL f', sur un tel système est réalisée par un test de vacuité sur la structure

$\mathcal{K}\overline{\otimes}\mathcal{A}_c \otimes \mathcal{A}_f$, où $\mathcal{A}_f$ est un $\mathcal{TGBA}$[6] représentant la négation de la formule f' ($f = \neg f'$). Ainsi, le produit synchronisé quotient de cette nouvelle structure sera défini par ce qui suit.

Définition 4.4 (Produit synchronisé quotient)**.** *Soient* $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{R}_\mathcal{K}, \mathcal{AP}, \mathcal{E}, \Pi_\mathcal{K}, \Gamma \rangle$ *une* $\mathcal{SKE}$ *symétrique p.r.a un groupe* $\mathcal{G} \subseteq \mathfrak{S}(\mathcal{AP} \cup \mathcal{E})$, $\mathcal{A}_c = \langle L, l_0, \mathcal{R}_{\mathcal{A}_c}, \mathcal{E}, \Delta \rangle$ *un* $\mathcal{CA}$ *de* $\mathcal{K}$ *et un* $\mathcal{TGBA}$, $\mathcal{A}_f = \langle Q_{\mathcal{A}_f}, Q^0_{\mathcal{A}_f}, \mathcal{R}_{\mathcal{A}_f}, \mathcal{AP}, \mathcal{F} \rangle$. *Le* produit synchronisé quotient *de* $\mathcal{K}\overline{\otimes}\mathcal{A}_c \otimes \mathcal{A}_f$ *est un* $\mathcal{TGBA}$, $\overline{\mathcal{K}\overline{\otimes}\mathcal{A}_c \otimes \mathcal{A}_f} = \langle Q, Q_0, \mathcal{R}, \mathcal{AP}, \mathcal{F} \rangle$ *défini par :*

- $Q = \{\langle \mathcal{H}, O, l, q \rangle | \mathcal{H} \subseteq \mathcal{G}, O \subseteq \text{Reach}(\mathcal{K}), l \in L, q \in Q_{\mathcal{A}_f}, \wedge \mathcal{H}.O = O\}$,
- $Q_0 = \{\langle \mathcal{G}, \mathcal{G}.s, l_0, q \rangle \mid s \in \mathcal{S}_0 \wedge q \in Q^0_{\mathcal{A}_f}\}$,
- $\mathcal{R}$ *est défini par construction comme suit :* $\big\langle \langle \mathcal{H}, O, l, q \rangle, \langle \mathcal{H}', O', l', q' \rangle \big\rangle \in \mathcal{R}$ *ssi* $\exists (s, s') \in O \times \mathcal{S}$ *tel que,* $\langle s, s' \rangle \in \mathcal{R}$, $\langle l, l' \rangle \in \mathcal{R}_{\mathcal{A}_c}$, $\langle q, P, F, q' \rangle \in \mathcal{R}_{\mathcal{A}_f}$, *avec* $P \subseteq \Pi_\mathcal{K}(s)$, $\Gamma(s, s') \in \Delta(l, l')$. *Alors,* $O' = (\mathcal{H} \cap \mathcal{G}_\gamma \cap \mathcal{G}_P).s'$, *où* $\gamma = \Delta(l, l')$ *et* $P = \Pi_{\mathcal{A}_f}(q, q')$, $\mathcal{H}' \subseteq \mathcal{G}_{O'}$.

Note 4.3. *La preuve de la validité de cette construction et par conséquent l'équivalence entre le résultat de test de vacuité sur la structure quotient* $\overline{\mathcal{K}\overline{\otimes}\mathcal{A}_c \otimes \mathcal{A}_f}$ *et la structure ordinaire* $\mathcal{K}\overline{\otimes}\mathcal{A}_c \otimes \mathcal{A}_f$ *sera présentée dans la section 5.6.*

4.4 Conclusion

Dans ce chapitre, nous avons présenté la méthode de Haddad & al [HIA00] pour la vérification des système adaptée pour la gestion dynamique et locales des asymétries. Globalement, cette méthode modélise le système en utilisant deux composantes : une structure de Kripke symétrique et un automate de contrôle. La structure de Kripke décrit les comportements nominaux du système et l'automate de contrôle restreint ces comportements à ceux réellement produit. Ainsi, les asymétries du système sont transférées vers l'automate de contrôle, ce qui qui permet leur gestion de manière dynamique.

Nous avons ensuite proposé une amélioration de ce modèle, qui considère des automates de contrôle basés sur les événements au lieu de ceux basées sur les états. Cette amélioration rend d'une part, la modélisation plus intuitive, et d'autre part, elle produit des automates de contrôle plus compacts. Le modèle, ainsi décrit, permet la construction d'un structure quotient sur laquelle la vérification de propriétés temporelles peut être effectuée de manière équivalente à celle réalisée

[6] Définition 2.10

en utilisant la structure ordinaire. On opère alors cette vérification à l'aide d'un des algorithmes de test de vacuité proposés dans la littérature.

Cependant, la gestion dynamique des asymétries pose un problème de taille : les états de la structure quotient (qui sont, au fait, des ensembles d'états de la structure ordinaire) *n'ont pas forcément des intersections vides et peuvent aussi être inclus les uns dans les autres, i.e., les états de la structure quotient ne définissent pas nécessairement une partition des états de la structure ordinaire*. Par conséquent, il est possible que la taille de la structure quotient soit plus grande que la structure ordinaire.

Ce problème est étudié en détail dans le chapitre suivant. L'idée est de développer des algorithmes de test de vacuité, opérant à la volée (on-the-fly) et qui bénéficient, justement, des propriétés d'intersections non vides et d'inclusions des états de la structure quotient pour réduire sa taille, et par conséquent permettent l'optimisation du processus de vérification.

Chapitre 5

Optimisation de la vérification des automates ensemblistes[1]

5.1 Introduction

Comme nous l'avons déjà mentionné dans le chapitre précédent, l'avantage de l'approche par symétries locales constitue aussi son inconvénient. En effet, les états du produit synchronisé quotient ne forment pas une partition des états du produit synchronisé ordinaire.

Considérons les automates de la Fig. 5.1. Cette figure montre deux automates $\mathcal{A}$ et $\mathcal{B}$ tels que les états de $\mathcal{B}$ sont des ensembles d'états de $\mathcal{A}$. L'automate $\mathcal{A}$ correspond au produit synchronisé $\mathcal{K}\overline{\otimes}\mathcal{A}_c \otimes \mathcal{A}_f$ et $\mathcal{B}$ correspond à $\overline{\mathcal{K}\overline{\otimes}\mathcal{A}_c \otimes \mathcal{A}_f}$.

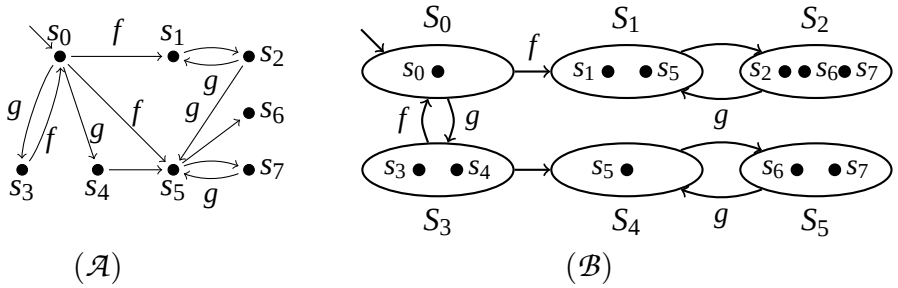


FIG. 5.1 – Un exemple d'automate ensembliste $\mathcal{B}$ de l'automate $\mathcal{A}$, avec $\mathcal{F}_\mathcal{A} = \mathcal{F}_\mathcal{B} = \{f, g\}$.

[1] Les idées développées dans ce chapitre ont fait l'objet d'un rapport technique ([BDL06]), et d'une publication dans ACSD 2007 ([BDL07]).

On remarque ici que certains états de $\mathcal{A}$ apparaissent dans plusieurs états de $\mathcal{B}$ (e.g., s_7 apparaît dans S_2 et S_5). Plus important encore, certains états de $\mathcal{B}$ sont inclus dans d'autres (e.g., $S_4 \subseteq S_1$).

Ainsi, dans son état actuel, la méthode décrite dans le chapitre 4 peut aboutir à la construction d'un automate, dit "quotient", qui contient plus d'états que l'automate originel ! Si nous considérons que l'ensemble des états de l'automate originel est $Q_{\mathcal{A}}$, alors la technique peut construire un automate $\mathcal{B}$ tel que $Q_{\mathcal{B}} \subseteq 2^{Q_{\mathcal{A}}}$. Donc, dans le pire des cas $Q_{\mathcal{B}}$ est égal à $2^{|Q_{\mathcal{A}}|}$ (ceci n'empêche pas que dans plusieurs cas pratiques on a $|Q_{\mathcal{B}}| \ll |Q_{\mathcal{A}}|$).

Le constat important est que cette méthode ne tire pas profit des inclusions entre les états de $\mathcal{B}$, ce qui pourrait être très bénéfique pour la réduction de la taille de $\mathcal{B}$, et donc l'optimisation de la vérification. Sur notre exemple, il est possible d'éviter de construire (et stocker) les états S_4 et S_5 car l'information qu'ils intègrent est déjà représentée par les états S_1 et S_2.

Il important de noter que les problèmes de l'inclusion et l'intersection non-vide n'est pas propre à la méthode présentée dans le chapitre précédent. En effet, nous observons les mêmes problèmes sur *les graphes de dépliage* [CGP00], ou encore sur *les graphes d'observation* [HIK04], que la pratique montre comme de bonnes techniques de réduction.

Ainsi, et pour être applicable dans le cas général, nous proposons ici des solutions attenuant l'effet de ces problèmes dans le cadre des général des *automates ensemblistes*, avec la création de nouveaux algorithmes de test de vacuité, spécialement adapté pour ce type d'automates.

Dans la section 5.2 nous définissons formellement les *automates ensemblistes*, et nous proposons un ensemble de cinq propriétés liant $\mathcal{A}$ à $\mathcal{B}$, qui permettent d'établir une équivalence entre les deux automates du point de vue du test de vacuité. La section 5.3 présente un algorithme de test de vacuité adapté pour ce type d'automates. Dans la section 5.4 nous opérons une modification de l'algorithme précédent qui le rend plus rapide mais *semi-décisionnel* : il répond à la question de la vacuité par "oui" ou "je ne sais pas". La section 5.5 discute de la génération des *contre-exemples*. Aussi, les différentes notions introduites dans ce chapitre nous permettent de prouver la correction de la construction du produit synchronisé quotient introduit dans le chapitre précédent. Cette preuve est présentée dans la section 5.6.

5.2 Définitions des Automates de Büchi Ensemblistes ($\wp$-$\mathcal{UTGBA}$)

Pour une présentation générique de notre méthode, nous considérons des automates dont la structure ressemble à celle des $\mathcal{TGBA}$ mais pour laquelle *on ne définit pas de propositions atomiques*. En effet, les propositions atomiques ne sont nécessaires que pour le calcul du produit synchronisé et peuvent être ignorées après cette opération : l'algorithme de test de vacuité n'a pas besoin des propositions atomiques pour s'accomplir.

Définition 5.1 (Un automate de Büchi généralisé non étiqueté avec transitions d'acceptation)**.** *Un automate de Büchi généralisé non étiqueté avec transitions d'acceptation ($\mathcal{UTGBA}$) est un $\mathcal{TGBA}$ sans proposition atomique. C'est un tuple $\mathcal{A} = \langle Q, Q^0, \mathcal{R}, \mathcal{F} \rangle$ tel que*
- *Q est un ensemble fini d'éléments, appelés* états,
- *$Q^0 \subseteq Q$ est l'ensemble des états initiaux,*
- *$\mathcal{F}$ est un ensemble fini d'éléments, appelés* conditions d'acceptation,
- *$\mathcal{R} \subseteq Q \times 2^{\mathcal{F}} \times Q$ est la relation de transition.*

Pour éviter toute confusion par rapport aux notions présentées dans le chapitre 2, nous reprenons les notions d'états accessibles, de chemin et de chemin acceptants pour le cas d'un $\mathcal{UTGBA}$.

Définition 5.2 (États accessibles d'un $\mathcal{UTGBA}$)**.** *Soit $\mathcal{A} = \langle Q, Q^0, \mathcal{R}, \mathcal{F} \rangle$ un $\mathcal{UTGBA}$. Un état $s \in Q$ est* accessible *si $s \in Q^0$ ou s'il existe une séquence $\langle s_0, F_0, s_1 \rangle \langle s_1, F_1, s_2 \rangle \cdots \langle s_{n-1}, F_{n-1}, s_n \rangle$ de transitions de $\mathcal{R}$ commençant dans un état initial $s_0 \in Q^0$ et se terminant par un état $s_n = s$. On note* $\mathrm{Reach}(\mathcal{A})$ *l'ensemble de tous les états accessibles de $\mathcal{A}$.*

Définition 5.3 (Chemin et Chemin acceptant d'un $\mathcal{UTGBA}$)**.** *Soit $\mathcal{A}$ un $\mathcal{UTGBA}$. Un* chemin *de $\mathcal{A}$ est une séquence infinie $\langle s_0, F_0, s_1 \rangle \langle s_1, F_1, s_2 \rangle \cdots$ de transitions de $\mathcal{R}$ commençant dans un état initial $s_0 \in Q^0$. Un chemin est acceptant si $\forall f \in \mathcal{F}, \forall i \geqslant 0, \exists j \geqslant i$, tel que $f \in F_j$, c'est-à-dire, si ses transitions sont étiquetées infiniment souvent par chaque condition d'acceptation. On note l'ensemble de tous les chemins de $\mathcal{A}$ par* $\mathrm{Run}(\mathcal{A})$ *et l'ensemble de ses chemins acceptants par* $\mathrm{Acc}(\mathcal{A})$*. Pour $\sigma = \sigma(0)\sigma(1)\sigma(2)\cdots \in \mathrm{Run}(\mathcal{A})$, on note $\sigma_{\mathrm{in}}(i)$, $\sigma_{\mathrm{acc}}(i)$, et $\sigma_{\mathrm{out}}(i)$ la source, les conditions d'acceptation, et la destination de la $i^{ème}$ transition de σ, autrement dit, $\sigma(i) = \langle \sigma_{\mathrm{in}}(i), \sigma_{\mathrm{acc}}(i), \sigma_{\mathrm{out}}(i) \rangle$. Finalement, on note σ^i*

le suffixe de σ *commençant par la* $i^{ème}$ *transition :* $\sigma^i = \sigma(i)\sigma(i+1)\sigma(i+2)\cdots$.

Tel que nous l'avons introduit dans le chapitre 2, l'objectif d'un algorithme de test de vacuité est de répondre à la question de la vacuité de l'ensemble Acc. Ici, nous voulons définir une relation d'équivalence entre deux $\mathcal{UTGBA}$ $\mathcal{A}$ et $\mathcal{B}$, basée sur le résultat du test de vacuité sur chacun des deux automates.

Définition 5.4 (Équivalence de vacuité)**.** *Deux* $\mathcal{UTGBA}$*,* $\mathcal{A}$ *et* $\mathcal{B}$ *sont* $\wp$*-équivalents ssi tous deux ont au moins une exécution acceptante ou aucun n'en a.*

$$\mathcal{A} \overset{\wp}{\equiv} \mathcal{B} \quad ssi \quad \mathrm{Acc}(\mathcal{A}) = \emptyset \iff \mathrm{Acc}(\mathcal{B}) = \emptyset$$

Nous proposons maintenant un ensemble de 5 propriétés qui lient deux $\mathcal{UTGBA}$ $\mathcal{A}$ et $\mathcal{B}$ où les états de $\mathcal{B}$ sont des ensembles d'états de $\mathcal{A}$ et $\mathcal{B}$ est $\wp$-équivalent à $\mathcal{A}$. L'idée qui sera poursuivie ultérieurement consistera à construire un automate $\mathcal{B}$ vérifiant ces propriétés et sur lequel le test de vacuité sera exécuté en lieu et place de $\mathcal{A}$.

Définition 5.5 (**$\wp$-$\mathcal{UTGBA}$**)**.** *Soient deux* $\mathcal{UTGBA}$*,* $\mathcal{A} = \langle Q_\mathcal{A}, Q^0_\mathcal{A}, \mathcal{R}_\mathcal{A}, \mathcal{F}_\mathcal{A}\rangle$ *et* $\mathcal{B} = \langle Q_\mathcal{B}, Q^0_\mathcal{B}, \mathcal{R}_\mathcal{B}, \mathcal{F}_\mathcal{B}\rangle$ *.* $\mathcal{B}$ *est un* $\wp$*-*$\mathcal{UTGBA}$ *sur* $\mathcal{A}$ *s'il satisfait les propriétés suivantes :*

$$Q_\mathcal{B} \subseteq 2^{Q_\mathcal{A}} \setminus \{\emptyset\} \tag{5.1}$$

$$\mathcal{F}_\mathcal{B} = \mathcal{F}_\mathcal{A} \tag{5.2}$$

$$\bigcup_{S \in Q^0_\mathcal{B}} S = Q^0_\mathcal{A} \tag{5.3}$$

$$\forall S \in \mathrm{Reach}(\mathcal{B}), \forall \langle s, F, s'\rangle \in \mathcal{R}_\mathcal{A}, \textit{ avec } s \in S, \exists S' \in Q_\mathcal{B} \textit{ tel que } s' \in S', \textit{et } \langle S, F, S'\rangle \in \mathcal{R}_\mathcal{B} \tag{5.4}$$

$$\forall \langle S, F, S'\rangle \in \mathcal{R}_\mathcal{B}, \forall s' \in S', \exists s \in S, \textit{ tel que } \langle s, F, s'\rangle \in \mathcal{R}_\mathcal{A} \tag{5.5}$$

Proposition 5.1. *Soient* $\mathcal{A} = \langle Q_\mathcal{A}, Q^0_\mathcal{A}, \mathcal{R}_\mathcal{A}, \mathcal{F}_\mathcal{A}\rangle$ *et* $\mathcal{B} = \langle Q_\mathcal{B}, Q^0_\mathcal{B}, \mathcal{R}_\mathcal{B}, \mathcal{F}_\mathcal{B}\rangle$ *deux* $\mathcal{UTGBA}$ *tels que* $\mathcal{B}$ *est un* $\wp$*-UTGBA sur* $\mathcal{A}$*. Alors* $\mathcal{A} \overset{\wp}{\equiv} \mathcal{B}$*.*

Démonstration. Nous voulons montrer que $\exists \sigma \in \mathrm{Acc}(\mathcal{A}) \iff \exists \sigma' \in \mathrm{Acc}(\mathcal{B})$.

($\Longrightarrow$) Soit $\sigma = \langle s_0, F_0, s_1\rangle\langle s_1, F_1, s_2\rangle \cdots \in \mathrm{Acc}(\mathcal{A})$. Partant du fait que $s_o \in Q^0_\mathcal{A}$ nous pouvons utiliser (5.3) et trouver un $S_0 \in Q^0_\mathcal{B}$ tel que $s_0 \in S_0$. Sachant que S_o est accessible dans $\mathcal{B}$ et contient s_0, on peut utiliser (5.4) pour trouver un $S_1 \in Q_\mathcal{B}$ tel

que $s_1 \in S_1$ et $\langle S_0, F_0, S_1 \rangle \in \mathcal{R}_\mathcal{B}$. Sachant que S_1 est accessible dans $\mathcal{B}$ et contient s_1 par construction, on utilise (5.4) encore une fois pour trouver un $S_2 \in Q_\mathcal{B}$ tel que $s_2 \in S_1$ et $\langle S_1, F_1, S_2 \rangle \in \mathcal{R}_\mathcal{B}$. Par itération sur la propriété (5.4) on peut construire une séquence $\sigma' = \langle S_0, F_0, S_1 \rangle \langle S_1, F_1, S_2 \rangle \cdots \in \mathrm{Run}(\mathcal{B})$ tel que $s_i \in S_i$ pour tout i. Puisque $\mathcal{F}_\mathcal{B} = \mathcal{F}_\mathcal{A}$ (5.2) et σ' visite chaque condition d'acceptation aussi souvent que σ, alors $\sigma' \in \mathrm{Acc}(\mathcal{B})$.

($\Longleftarrow$) Soit $\sigma' = \langle S_0, F_0, S_1 \rangle \langle S_1, F_1, S_2 \rangle \cdots \in \mathrm{Acc}(\mathcal{B})$. Construisons un arbre dont les nœuds (exceptée la racine), sont des états de $\mathcal{A}$. Appelons $\perp$ la racine de l'arbre à la profondeur 0. Les nœuds de profondeur $n > 0$ sont exactement les états appartenant à S_{n-1}. A la profondeur $n > 1$, le père s d'un état s' est choisi parmi les nœuds de profondeur $n-1$ tel que $\langle s, F_{n-1}, s' \rangle \in \mathcal{R}_\mathcal{A}$; (5.5) garantit l'existence de s. Le père de tout nœud à la profondeur 1 est $\perp$. Tous les arcs de cet arbre, excepté ceux sortant de la racine, correspondent à des transitions de $\mathcal{R}_\mathcal{A}$.

L'ensemble des nœuds à la profondeur $n > 0$ est un sous-ensemble de $Q_\mathcal{A}$, qui est fini, par conséquent même si cet arbre est infini, il a un degré fini. Par le lemme de König, il contient une branche infinie. La séquence construite en suivant les arcs de cette branche infinie, en ignorant l'arc quittant $\perp$, $\langle s_0, F_0, s_1 \rangle \langle s_1, F_1, s_2 \rangle \cdots$ est un un chemin de $\mathcal{A}$ ($s_0 \in Q^0_\mathcal{A}$) qui visite chaque condition d'acceptation aussi souvent que σ'. Donc, c'est une exécution acceptante de $\mathcal{A}$.

□

A présent, nous développons deux propositions qui introduisent notre nouvel algorithme de test de vacuité sur les $\wp$-$\mathcal{UTGBA}$. Les deux propositions utilisent la notation suivante.

Définition 5.6 (Substitution des états initiaux)**.** *Soit $\mathcal{A} = \langle Q, Q^0, \mathcal{R}, \mathcal{F} \rangle$ un $\mathcal{UTGBA}$, et $T \subseteq Q$ un ensemble d'états de $\mathcal{A}$. On note $\mathcal{A}[T]$ l'automate partageant la même structure que $\mathcal{A}$ mais ayant T comme ensemble d'états initiaux. Autrement dit, $\mathcal{A}[T] = \langle Q, T, \mathcal{R}, \mathcal{F} \rangle$.*

La première proposition tire profit du fait que certains états sont inclus dans d'autres, pour l'optimisation du test de vacuité : considérant l'exemple de la Fig.5.1, dans le $\mathcal{UTGBA}$ $\mathcal{B}$ nous avons $S_4 \subseteq S_1$. Puis qu'il n'existe pas de chemin acceptant qui traverse S_1 (qui passe infiniment souvent par f et g) alors, il est inutile de parcourir les successeurs de S_4 car il n'y aura *forcément* pas de chemin acceptant qui traverse cet état.

Proposition 5.2. *Soit* $\mathcal{B} = \langle Q_\mathcal{B}, Q^0_\mathcal{B}, \mathcal{R}_\mathcal{B}, \mathcal{F} \rangle$ *un* $\wp$*-UTGBA sur* $\mathcal{A} = \langle Q_\mathcal{A}, Q^0_\mathcal{A}, \mathcal{R}_\mathcal{A}, \mathcal{F} \rangle$ *et soient deux états* T *et* D *de* $Q_\mathcal{B}$ *tels que* $D \subseteq T$. *On a*

$$\mathrm{Acc}(\mathcal{B}[\{T\}]) = \emptyset \implies \mathrm{Acc}(\mathcal{B}[\{D\}]) = \emptyset$$

Démonstration. On prouve la contraposée :

$$\mathrm{Acc}(\mathcal{B}[\{D\}]) \neq \emptyset \implies \mathrm{Acc}(\mathcal{B}[\{T\}]) \neq \emptyset.$$

Considérons $\sigma = \langle D_0, F_0, D_1 \rangle \langle D_1, F_1, D_2 \rangle \cdots \in \mathrm{Acc}(\mathcal{B}[\{D\}])$. En utilisant (5.5) de la même façon que pour la preuve de la partie ($\Longleftarrow$) de la proposition 5.1, on peut trouver une séquence $\sigma' = \langle d_0, F_0, d_1 \rangle \langle d_1, F_1, d_2 \rangle \cdots$ telle que $\forall i \geqslant 0, d_i \in D_i$. Considérons la première transition de σ' : $\langle d_0, F_0, d_1 \rangle$. Puisque $D \subseteq T$, on a $d_0 \in T$, par conséquent, on peut appliquer la proposition (5.4) pour trouver une ensemble T_1 tel que $d_1 \in T_1$ et $\langle T, F_0, T_1 \rangle \in \mathcal{R}_\mathcal{B}$. Puisque $d_1 \in T_1$ on peut appliquer la proposition (5.4) encore une fois pour trouver $\langle T_1, F_1, T_2 \rangle \in \mathcal{R}_\mathcal{B}$. En répétant cette opération, on construit une exécution acceptée $\sigma'' = \langle T, F_0, T_1 \rangle \langle T_1, F_1, T_2 \rangle \cdots$ de $\mathcal{B}[\{T\}]$.

□

La proposition suivante nous permet la décomposition d'une transition $\langle R, F, T \rangle$ en un ensemble de transitions $\langle R, F, T_1 \rangle, \ldots, \langle R, F, T_n \rangle$ avec $T_1 \cup \cdots \cup T_n = T$, tout en respectant la $\wp$-équivalence. Une telle opération nécessite l'ajout d'états et de transitions à l'automate initial.

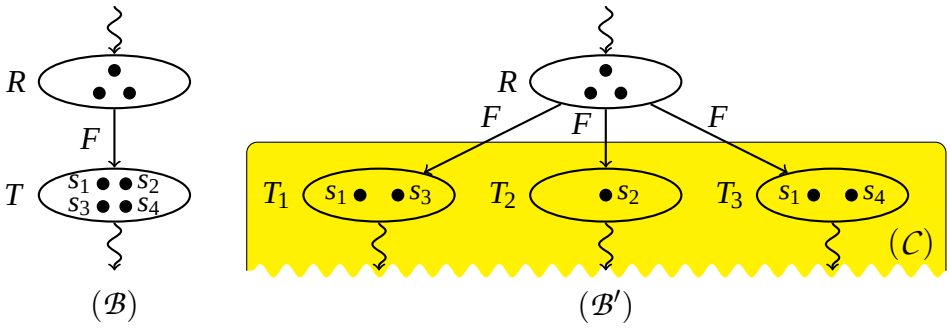


FIG. 5.2 – Exemple de décomposition d'une transition $\langle R, F, T \rangle$.

Autrement dit, on veut substituer T par un automate $\mathcal{C}$ qui a $\{T_1, \ldots, T_n\}$ comme ensemble d'états initiaux, et qui est $\wp$-equivalent à $\mathcal{A}[T]$. La Fig. 5.2 illustre cette opération. Cette décomposition s'avère importante si quelques uns des états T_i ont déjà été visités par l'algorithme de test de vacuité, et dans ce cas il est inutile de les visiter à nouveau.

Proposition 5.3 (Décomposition d'une transition dans un $\wp$-$\mathcal{UTGBA}$)**.** *Soit $\mathcal{B} = \langle Q_\mathcal{B}, Q^0_\mathcal{B}, \mathcal{R}_\mathcal{B}, \mathcal{F} \rangle$ un $\wp$-UTGBA sur $\mathcal{A} = \langle Q_\mathcal{A}, Q^0_\mathcal{A}, \mathcal{R}_\mathcal{A}, \mathcal{F} \rangle$. Soient une transition $\langle R, F, T \rangle \in \mathcal{R}_\mathcal{B}$ et $\mathcal{C} = \langle Q_\mathcal{C}, Q^0_\mathcal{C}, \mathcal{R}_\mathcal{C}, \mathcal{F} \rangle$ un $\wp$-UTGBA sur $\mathcal{A}[T]$. L'automate $\mathcal{B}' = \langle Q_\mathcal{B} \cup Q_\mathcal{C}, Q^0_{\mathcal{B}'}, \mathcal{R}_{\mathcal{B}'}, \mathcal{F} \rangle$ tel que,*

$$Q^0_{\mathcal{B}'} = \begin{cases} \left(Q^0_\mathcal{B} \setminus \{T\}\right) \cup Q^0_\mathcal{C} & \text{si } T \in Q^0_\mathcal{B} \\ Q^0_\mathcal{B} & \text{autrement} \end{cases}$$
$$\mathcal{R}_{\mathcal{B}'} = (\mathcal{R}_\mathcal{B} \setminus \{\langle R, F, T \rangle\}) \cup \{\langle R, F, T' \rangle \mid T' \in Q^0_\mathcal{C}\} \cup \mathcal{R}_\mathcal{C}$$

est un $\wp$-UTGBA sur $\mathcal{A}$.

Démonstration. Les propriétés (5.1) et (5.2) de la définition 5.5 sont satisfaites par $\mathcal{B}'$ par définition. Il reste à prouver les propriétés (5.3) à (5.5).

(5.3) On doit montrer que $\bigcup_{S \in Q^0_{\mathcal{B}'}} S = Q^0_\mathcal{A}$. Puisque $\mathcal{B}$ est un $\wp$-UTGBA sur $\mathcal{A}$, on a $\bigcup_{S \in Q^0_\mathcal{B}} S = Q^0_\mathcal{A}$. Si $T \notin Q^0_\mathcal{B}$ alors $Q^0_{\mathcal{B}'} = Q^0_\mathcal{B}$ et la propriété (5.3) est satisfaite. Autrement, puisque $\mathcal{C}$ est un $\wp$-UTGBA sur $\mathcal{A}[T]$, on a $\bigcup_{S \in Q^0_\mathcal{C}} S = T$, donc (5.3) est satisfaite aussi.

(5.4) Soit $\langle s, F, s' \rangle \in \mathcal{R}_\mathcal{A}$ une transition de $\mathcal{A}$, et $S \in \text{Reach}(\mathcal{B}')$ tel que $s \in S$. Pour prouver (5.4) on doit trouver une transition $\langle S, F, S' \rangle \in \mathcal{R}_{\mathcal{B}'}$ telle que $s' \in S'$. On distingue trois cas qui couvrent tous les états de $\text{Reach}(\mathcal{B}')$:
- Si $S \in (\text{Reach}(\mathcal{B}) \setminus \{R\}) \vee (S = R \wedge s' \notin T)$ alors puisque $\mathcal{B}$ est un $\wp$-UTGBA sur $\mathcal{A}$, sa propriété (5.4) garantit l'existence d'une telle transition.
- Si $S = R \wedge s' \in T$, alors puisque $\mathcal{C}$ est un $\wp$-UTGBA sur $\mathcal{A}[T]$, sa propriété (5.3) assure que $\exists T' \in Q^0_\mathcal{C}$ tel que $s' \in T'$, par conséquent $\langle R, F, T' \rangle \in \mathcal{R}_{\mathcal{B}'}$ par définition de $\mathcal{B}'$.
- Finalement, si $S \in \text{Reach}(\mathcal{C})$ alors puisque $\mathcal{C}$ est un $\wp$-UTGBA sur $\mathcal{A}[T]$, sa propriété (5.4) garantit l'existence d'une telle transition.

(5.5) Pour prouver (5.5), on doit assurer que tous les tuples $\langle S, F, S' \rangle$ de $\mathcal{R}_{\mathcal{B}'}$ vérifient $\forall s' \in S', \exists s \in S, \langle s, F, s' \rangle \in \mathcal{R}_\mathcal{A}$. On sait que $\mathcal{B}$ est un $\wp$-UTGBA sur $\mathcal{A}$, donc la propriété est satisfaite pour toute transition $\langle S, F, S' \rangle$ de $\mathcal{R}_\mathcal{B}$. D'une manière similaire, puisque $\mathcal{C}$ est un $\wp$-UTGBA sur $\mathcal{A}[T]$, (5.5) est vraie pour tout tuple dans $\mathcal{R}_\mathcal{C}$. Il reste à vérifier l'ensemble des transitions suivant : $\{\langle R, F, T' \rangle \mid T' \in Q^0_\mathcal{C}\}$. Cependant, puisque (5.5) était vraie pour $\langle R, F, T \rangle$, elle reste valide pour toutes ces transitions.

□

5.3 Test de vacuité d'un $\wp$-UTGBA

Un automate de Büchi généralisé accepte un mot (*i.e.* est non vide) s'il contient un cycle acceptant dans lequel toutes les conditions d'acceptation apparaissent. Les *emptiness checks*, qui testent cette propriété, peuvent être divisés en deux classes :

ceux effectuant des parcours en profondeurs imbriqués (NDFS), et ceux basés sur les composantes fortement connexes (SCC).

Les NDFS ont longtemps été préférés aux algorithmes basés sur les SCC à cause de leur meilleure consommation mémoire (une paire de bits supplémentaires par état au lieu d'un entier complet). Cependant, Schwoon et Esparza [SE05] ont mis en avant le fait que ces données supplémentaires sont négligeables devant la taille d'un état (une centaine d'octets). De plus, les algorithmes basés sur les SCC ont de nombreux avantages : ils prouvent la non vacuité d'un automate en explorant moins d'états [GV05, CDLP05], et supportent des conditions d'acceptation généralisées sans sur-coût [CDLP05]. L'algorithme que nous présentons est basé sur les SCC et procède de celui présenté par Couvreur [Cou99], lui-même dérivé de [Tar71, Tar72].

5.3.1 Description de l'algorithme originel

L'idée de cet algorithme est d'énumérer toutes les *composantes fortement connexes maximales (MSCC)* d'un automate : tout graphe contient au moins une MSCC sans arc sortant (MSCC terminale). Pour lister toutes les MSCC, on doit trouver une MSCC terminale, la retirer de l'automate, puis trouver une MSCC terminale de l'automate résultant. On itère ce processus jusqu'à l'obtention de toutes les MSCC de l'automate. L'algorithme effectue pour cela *un parcours en profondeur (DFS)* sur la structure de l'automate. Durant ce PEP, il maintient une pile des SCC traversées. Quand une nouvelle transition est rencontrée, la pile des SCC peut être augmentée ou réduite. Quand une SCC est retirée de la pile (elle est terminale) on teste si elle est acceptante : dans ce cas, l'algorithme se termine. Autrement, chaque état de cette composante est marqué *retiré*, de façon à l'ignorer si le PEP le rencontre une nouvelle fois.

5.3.2 Adaptation au $\wp$-$\mathcal{UTGBA}$

L'algorithme que nous proposons diffère de l'originel [Cou99] sur deux points.

Point 1. Le test des états à retirer est généralisé : tout état D retiré par la DFS de la procédure précédente est aussi retiré par la DFS du nouvel algorithme, mais aussi peut être retiré tout état T vérifiant $T \subseteq D$. Ceci est possible grâce à la proposition 5.2.

Point 2. La Fig. 5.3 illustre la seconde différence. Considérons l'automate $\mathcal{B}_1$ où la DFS est en train d'examiner la transition $\langle R, F, T \rangle$, T étant un nouvel état (non encore visité). Notons qu'il existe un état D dans la pile de recherche (ou plus généralement, dans n'importe quelle SCC de la pile de recherche)

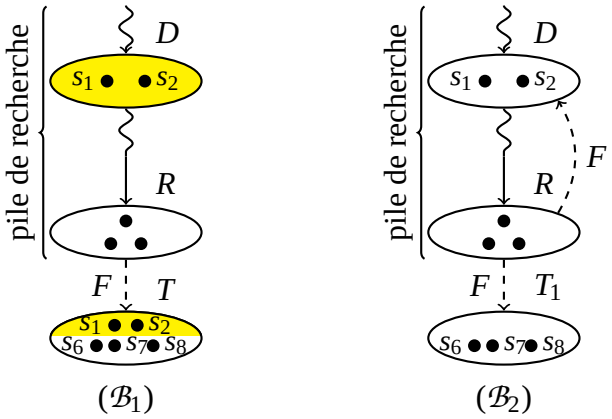


FIG. 5.3 – Tests d'inclusion dans la pile. On réécrit $\mathcal{B}_1$ en $\mathcal{B}_2$.

tel que $D \subseteq T$. Du point de vue de l'automate subalterne $\mathcal{A}$, ceci équivaut à l'existence d'un ensemble d'états dans R qui peuvent atteindre des états de D et vice-versa. Par conséquent, ils appartiennent à la même SCC. Pour le test de vacuité, il est bénéfique de diviser la transition $\langle R, F, T \rangle$ comme indiqué dans $\mathcal{B}_2$: cet automate rend explicite la boucle sur la SCC et réutilise les états déjà visités. Une telle décomposition est correcte du fait de la proposition 5.3, mais nécessite des contraintes supplémentaires, que nous formalisons dans ce qui suit.

Soit $\mathcal{B} = \langle Q_\mathcal{B}, Q^0_\mathcal{B}, \mathcal{R}_\mathcal{B}, \mathcal{F} \rangle$ un $\wp$-UTGBA sur $\mathcal{A}$. Decomp$(\mathcal{B}, \langle S, F, T \rangle, D)$ est une opération qui réalise la décomposition de la proposition 5.3. En plus de $\mathcal{B}$ et $\langle S, F, T \rangle$, qui ont la même signification que celle de la proposition, l'argument D est un état de $\mathcal{B}$ qui est, aussi, un sous-ensemble de T. Decomp construit l'automate $\mathcal{C}$ avec deux contraintes supplémentaires :

Contrainte 1. On doit assurer que $D \in Q^0_\mathcal{C}$ (les autres états de $Q^0_\mathcal{C}$ vont, par définition, compléter T).

Contrainte 2. $\mathcal{R}_\mathcal{C}$ ne doit pas ajouter de transitions pour les états de $\mathcal{B}$, autrement dit $\{\langle S, F, S' \rangle \in \mathcal{R}_\mathcal{C} \mid S \in \text{Reach}(\mathcal{B})\} \subseteq \mathcal{R}_\mathcal{B}$.

Decomp retourne une paire $\mathcal{B}', Q^0_\mathcal{C}$: le nouvel automate, et l'ensemble des états initiaux de $\mathcal{C}$.

Note 5.1. *Puisqu'en pratique on construit $\mathcal{B}$ à la volée et que* Decomp *ne rajoute pas de transitions à la partie de $\mathcal{B}$ qui a été déjà visitée par l'algorithme peut considérer tout automate issu de* Decomp *comme s'il s'agissait*

de l'automate de départ.

L'algorithme 1 montre le listing complet du processus de test de vacuité d'un $\wp$-$\mathcal{UTGBA}$. Trois opérations sont nécessaires pour manipuler les ensembles d'états : le test d'égalité, l'inclusion et la décomposition. On les note respectivement, $=$, $\subseteq$ et Decomp. Ces opérations sont présentées génériquement car leur réalisation dépend de la manière dont les ensembles d'états sont codés.

5.3.3 Correction du nouvel algorithme

On utilise les notations suivantes pour la preuve. A tout moment de l'exécution, on note le contenu de *todo* et *SCC* comme suit :

$$todo = \langle state_0, succ_0\rangle\langle state_1, succ_1\rangle \cdots \langle state_m, succ_m\rangle$$
$$SCC = \langle root_0, la_0, acc_0, rem_0\rangle\langle root_1, la_1, acc_1, rem_1\rangle \cdots \langle root_n, la_n, acc_n, rem_n\rangle$$

todo est la pile gérée par la DFS, chaque élément de cette pile est une paire $\langle state_i, succ_i\rangle$ où $succ_i$ est l'ensemble des transitions sortantes de l'état $state_i$ qui n'ont pas été encore visitées. On appelle la séquence $state_0 \ldots state_m$ *le chemin de recherche*.

SCC est la pile des composantes fortement connexes. Chaque quadruplet dans *SCC* représente donc une composante fortement connexe, S_i, traversée par le chemin de recherche. $root_i$ est le numéro du premier état de S_i. En association avec H (une application qui numérote chaque état visité), ils permettent de définir l'ensemble des états S_i, formant la $i^{ème}$ SCC :

$$S_i = \{s \in Q_{\mathcal{B}} \mid root_i \leqslant H[s] < root_{i+1}\} \qquad \text{pour } 0 \leqslant i < n$$
$$S_n = \{s \in Q_{\mathcal{B}} \mid root_n \leqslant H[s]\}$$

Tout terme acc_i représente l'ensemble des conditions d'acceptation traversées par les transitions qui relient les états de S_i. Le terme la_i représente l'ensemble des conditions d'acceptation de la transition entre la $(i-1)^{ème}$ et la $i^{ème}$ composantes. Une chaîne des SCC de la pile est illustrée par la Fig. 5.4. Finalement, rem_i est un ensemble d'états à retirer quand la composante est retirée de la pile (voir l'explication plus loin).

Données :
$\mathcal{B}$: $\langle Q_{\mathcal{B}}, Q^0_{\mathcal{B}}, \mathcal{R}_{\mathcal{B}}, \mathcal{F} \rangle$ un $\wp$-$\mathcal{UTGBA}$ à vérifier
$todo$: pile de $\langle state \in Q_{\mathcal{B}}, succ \subseteq \mathcal{R}_{\mathcal{B}} \rangle$
SCC : pile de $\langle root \in \mathbb{N}, la \subseteq \mathcal{F}, acc \subseteq \mathcal{F}, rem \subseteq Q_{\mathcal{B}} \rangle$
H : application de $Q_{\mathcal{B}} \mapsto \mathbb{N}$
$max := 0$

```
1  DFSpush(F ⊆ 𝓕, S ∈ Q) :
2  début
3      max := max + 1
4      H[S] := max
5      SCC.push(⟨max, F, ∅, ∅⟩)
6      todo.push(⟨S, {⟨R, F, T⟩ ∈ 𝓡_B | R = S}⟩)
7  fin

8  DFSpop() :
9  début
10     ⟨S, _⟩ := todo.pop()
11     SCC.top().rem.insert(S)
12     si H[S] = SCC.top().root alors
13         pour tous les R ∈ SCC.top().rem faire
14             H[R] := 0
15         SCC.pop()
16 fin

17 Merge(F ⊆ 𝓕, n ∈ ℕ) :
18 début
19     r := ∅
20     tant que n < SCC.top().root faire
21         F := (F ∪ SCC.top().acc ∪ SCC.top().la)
22         r := r ∪ SCC.top().rem
23         SCC.pop()
24     SCC.top().acc := SCC.top().acc ∪ F
25     SCC.top().rem := SCC.top().rem ∪ r
26     retourner SCC.top().acc
27 fin
```

```
28 Main() :
29 début
30     pour tous les S^0 ∈ Q^0_B faire
31         DFSpush(∅, S^0)
32         tant que ¬todo.empty() faire
33             si todo.top().succ = ∅ alors
34                 DFSpop()
35             sinon
36                 Prendre ⟨R,F,T⟩ dans todo.top().succ
37                 si ∃D ∈ H.keys() tel que (T ⊆ D) ∧ H[D] = 0 alors
38                     continue
39                 sinon si T ∉ H.keys() alors
40                     si ∃D ∈ H.keys() tel que D ⊆ T ∧ H[D] > 0 alors
41                         B, Q^0_C := Decomp(B, ⟨R,F,T⟩, D)
42                         todo.top().succ :=
                           todo.top().succ ∪ {⟨R,F,D⟩} ∪ {(R,F,T') | T' ∈ Q^0_C}
43                     sinon
44                         DFSpush(F, T)
45                 sinon si H[T] > 0 alors
46                     si merge(F, H[T]) = F alors
47                         retourner ⊥
48     retourner ⊤
49 fin
```

Algorithme 1 : Test de vacuité d'un $\wp$-$\mathcal{UTGBA}$.

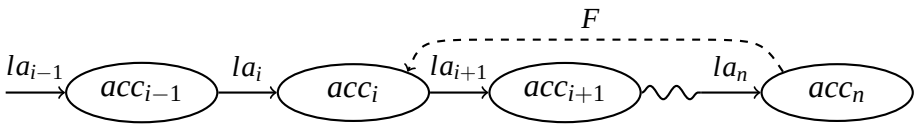


FIG. 5.4 – La signification de *la* et *acc* dans la pile *SCC*.

Les états de l'automate $\mathcal{B}$ sont partitionnés en trois ensembles :
– Les états *activés* sont ceux qui correspondent à des clés de H et ont une valeur (dans H) différente de 0 : *activés*= $\{S \in Q_{\mathcal{B}} \mid H[S] > 0\}$. Un état $S \in$ *activés* est dit *activé*.
– les états *retirés* sont ceux qui correspondent à des clés de H et ont une valeur égale à 0 : *retirés*= $\{S \in Q_{\mathcal{B}} \mid H[S] = 0\}$. Un état $S \in$ *retirés* est dit *retiré*.
– enfin, les états *non-explorés* sont ceux qui ne correspondent pas à des clés pour H.

Initialement, tous les états sont *non-explorés*. La procédure "DFSpush" est le seul endroit où un état peut être déplacé de l'ensemble *non-explorés* à l'ensemble *activés*, et la procédure "DFSpop" est le seul endroit où il peut être déplacé de l'ensemble *activés* à l'ensemble *retirés*.

Pour prouver la correction de l'algorithme, nous montrons que les invariants suivants sont préservés sur chaque ligne de la fonction "Main" :

Proposition 5.4. *$m \geqslant n$ (dans la notation de todo et SCC) et il existe une function strictement croissante f telle que $\forall i \leqslant n, root_i = H[state_{f(i)}]$. En d'autres termes, $root_0, root_1, \ldots, root_n$ est une sous-séquence de $H[state_0], H[state_1], \ldots, H[state_m]$. (Les racines des composantes fortement connexes sont dans le chemin de recherche et dans le même ordre.)*

Proposition 5.5. *Pour tout $i \leqslant n$, le sous-graphe dont les états sont dans S_i est une SCC. De plus, il existe un cycle dans cette SCC qui visite toutes les conditions d'acceptation de acc_i. Enfin, $S_0, S_1, \cdots, S_n$ est une partition de l'ensemble* activés.

Proposition 5.6. $\forall i < n, \exists s \in S_i, \langle s, la_{i+1}, state_{f(i+1)} \rangle \in \mathcal{R}$.

Proposition 5.7. *Pour tout $i < n$, rem_i regroupe tous les états de S_i qui ne sont pas dans le chemin de recherche.*

Proposition 5.8. *Pour tout état s dans l'ensemble* retirés, $\mathrm{Acc}(\mathcal{B}[\{s\}]) = \emptyset$.

Intuitivement, les deux premières propositions garantissent que si l'algorithme trouve une composante S_i tel que $acc_i = \mathcal{F}$, alors cette composante est accessible (proposition 5.4) et acceptante (proposition 5.5). La dernière proposition exprime

le fait que si l'algorithme retire tous les états alors il n'existe pas d'exécution acceptante.

Démonstration. Toutes les propositions 5.4-5.8 sont vérifiées quand l'algorithme atteint la ligne 32 pour la première fois. Il y a un seul élément dans *todo* et *SCC*, et la manière dont il a été mis par "DFSpush" garantit la proposition 5.4. S_0 contient un seul état, donc la proposition 5.5 est vérifiée. Puisque $n = m = 0$, les propositions 5.6 et 5.7 sont trivialement vérifiées. Aucun état n'a été retiré, donc la proposition 5.8 est aussi vérifiée.

Quand une transition $\langle R, F, T \rangle$ est retirée de $succ_m$, alors on est en présence de l'un des cas suivants :

– $H[T] = 0$, signifiant que T est un état retiré (lignes 37–38). A cause de la proposition 5.8, ni T, ni aucun de ses descendants ne doit faire partie d'un chemin acceptant : T peut être ignoré de la recherche. Le fait que les structures de données n'ont pas été altérées implique que les propositions 5.4 à 5.8 restent préservées.

– $\exists D, H[D] = 0 \wedge T \subseteq D$. Il existe un état retiré D qui contient T (lignes 37–38). En utilisant la proposition 5.2, on peut ignorer T et les propositions 5.4-5.8 sont préservées.

– $T \notin H.keys()$ et $\exists D, H[D] > 0 \wedge D \subseteq T$. Il existe un état activé D inclus dans la destination courante T, et T n'a pas encore été visité (lignes 39–42).

  Puisque on a déjà vu D on veut éviter de re-visiter cette partie dans T. On applique donc Decomp pour diviser T en D plus un ensemble d'états complétant T. Puisque cette opération ne rajoute pas de transitions pour les états qui appartiennent à $\mathcal{B}$ (cette contrainte est énoncée dans la définition de Decomp) et n'élimine aucun état ou transition déjà visités, on peut remplacer l'automate $\mathcal{B}$ par l'automate produit par Decomp sans altérer la DFS. Ceci revient au remplacement de la transition $\langle S, F, T \rangle$ par la liste de transitions dont les destinations sont D et l'ensemble des états de $Q^0_{\mathcal{C}}$.

  Ce cas n'a pas d'effet de bord sur la totalité de l'algorithme, qui peut continuer sur l'automate résultant comme s'il avait été l'automate initial. Les propositions 5.4-5.8 ne sont pas affectées.

– $T \notin H.keys()$ et nous n'avons pas trouvé $D \in$ *activés* tel que $D \subseteq T$ (lignes 43–44). T est une SCC triviale, il est rajouté dans H, empilé dans *SCC* comme nouvelle racine, et empilé dans *todo*. Ceci garantit les propositions 5.4-5.6 et

n'affecte pas les propositions 5.7-5.8.

– $H[T] > 0$, c'est à dire, T est un état activé (lignes 45–47). Soit $root_i$ la plus grande racine telle que $root_i < H[T]$, et notons $r_i = state_{f(i)}$ l'état associé. Grâce à la propositions 5.5, r_i et T appartiennent à la même SCC. De plus, r_i et R sont sur le même chemin de recherche alors r_i peut atteindre R. Puisque le traitement porte sur la transition $\langle R, F, T \rangle$, alors r_i, R, et T appartiennent à la même SCC.

  Par conséquent, on fusionne toutes les SCCs au-dessus de $root_i$. La nouvelle SCC est construite par l'union de $S_i, \ldots, S_n$, et à l'intérieur de cette SCC on trouve un cycle qui traverse les conditions d'acceptation $acc_i, \ldots, acc_n$ de chacune des SCCs précédentes, plus l'ensemble $\{la_{i+1}, \ldots, la_n\}$ des étiquettes des transitions intermédiaires. La Fig. 5.4 illustre la situations avant la fusion, montrant les conditions d'acceptation dans et entre les SCCs.

  La fonction "Merge" effectue la fusion des conditions d'acceptation tout en respectant la proposition 5.5. Aussi, la fusion des états retirés est réalisée en préservant la proposition 5.7. Les autres propositions ne sont pas altérées par cette opération. Le "Merge" retourne l'ensemble des conditions d'acceptation de la SCC obtenue après fusion. Si cet ensemble est $\mathcal{F}$ alors l'algorithme invalide la propriété à vérifier (ligne 48).

On considère à présent le cas où $succ_m = \emptyset$ (lignes 33–34). Les propriétés du DFS impliquent que l'algorithme a visité tous les successeurs de $state_m$ et leurs descendants.

$state_m$ est retiré du chemin de recherche (ligne 10), et pour préserver la proposition 5.7 il est ajouté à rem_n (ligne 11). L'ajout de $state_m$ n'affectera pas la proposition 5.4 tant que cet état n'est pas une racine d'une SCC. Si c'est le cas, la SCC doit aussi être retirée. Ceci n'affecte pas la proposition 5.6 mais pour préserver la proposition 5.5 on doit aussi retirer les états de S_i de l'ensemble *activés*. Arrivé à ce point, on sait que la SCC en tête de la pile est maximale :

– aucun état non exploré ne peut faire partie de cette SCC car nous avons exploré tous les descendants de cet état,
– aucun état *activé* d'une SCC plus basse (dans la pile) ne peut faire partie de celle de tête, car on les aurait déjà fusionnées si un descendant $state_m$ pouvait l'atteindre,
– aucun état *retiré* ne peut faire partie de cette SCC. Ceci est du à la proposition 5.8.

Aussi, on sait que cette SCC ne contient pas de cycle acceptant. Autrement, l'ajout d'un tel cycle aurait conduit l'algorithme à se terminer à la ligne 47. Par conséquent, $\forall q \in S_i, \mathrm{Acc}(\mathcal{B}[q]) = \emptyset$, et on peut marquer tous ces états comme retirés sans invalider la proposition 5.8. A cause de la proposition 5.7 (ligne 38), rem_n contient tous les états de S_i, donc cette boucle retire tous les états de la SCC comme c'est nécessaire pour la proposition. 5.5.

Si l'algorithme se termine à la ligne 48, la pile *todo* est vide. Par application de la proposition 5.4, ceci signifie que l'ensemble *activés* est vide. Puisque la DFS a exploré tous les états accessibles de l'automate, on peut conclure que tous les états accessibles ont été retirés, ce qui, grâce à la proposition 5.8, implique que $\mathrm{Acc}(\mathcal{B}) = \emptyset$.

Ainsi, si l'algorithme se termine sur la ligne 47, alors il existe une exécution acceptante. Par contre, s'il se termine sur la ligne 48 alors il n'y a pas d'exécution acceptée. La terminaison de l'algorithme est garantie parce qu'il effectue une DFS sur un automate fini. Du fait de l'opération Decomp, l'algorithme peut explorer plus de $|Q_{\mathcal{B}}|$ états, mais il ne peut explorer plus que $|2^{Q_{\mathcal{A}}}|$ états.

Finalement, on peut conclure que l'algorithme retourne $\perp$ ssi $\mathcal{B}$ contient un chemin acceptant.

□

5.4 Test de vacuité approximatif d'un $\wp$-$\mathcal{UTGBA}$

Considérons à nouveau la Fig. 5.3. Dans la section précédente, nous avons vu que les lignes 40–42 de l'algorithme de test de vacuité transforment la situation illustrée par l'automate $\mathcal{B}_1$, où l'algorithme atteint l'état $T \supseteq D$ tel que D appartient à la pile de recherche, en celle illustrée par l'automate $\mathcal{B}_2$. Cette transformation a pour but la réutilisation des états existants et par conséquent la construction des SCCs le plus rapidement possible. Nous avons prouvé que cette transformation préserve le résultat du test de vacuité ($\mathcal{B}_1 \overset{\wp}{\equiv} \mathcal{B}_2$).

A présent, nous voulons examiner la situation illustrée par l'automate $\mathcal{B}_3$ de la Fig. 5.5, où le test de vacuité traite une transition $\langle R, F, T\rangle$ telle que $T \subseteq D$ et D est dans la pile de recherche. Nous voulons remplacer cette transition par $\langle R, F, D\rangle$ comme illustré dans $\mathcal{B}_4$. Sur l'algorithme précédent, ceci se traduit par le remplacement des lignes 40–42 par :

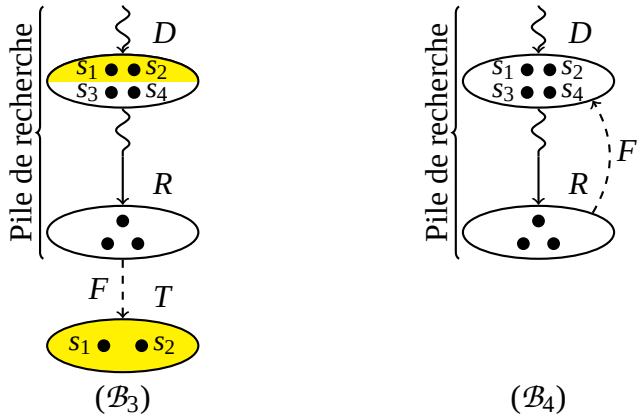


FIG. 5.5 – Tests d'inclusion dans la pile de recherche, où $\mathcal{B}_3$ est réécrit en $\mathcal{B}_4$.

40 **si** $\exists D \in H.keys()$ *such that* $T \subseteq D \wedge H[D] > 0$ **alors**
41 | // Notez l'ordre de T et D ci-dessus.
42 | $todo$.top().$succ$:= $todo$.top().$succ \cup \{\langle R, F, D\rangle\}$
43 **sinon**

Considérons que $\mathcal{B}_3$ est un $\wp$-UTGBA sur $\mathcal{A}$. On note que la transformation ci-dessus ne préserve pas la propriété (5.5) de la définition 5.5, car s_3 et s_4 n'ont pas de prédécesseur dans R ; donc $\mathcal{B}_4$ *n'est pas un* $\wp$-UTGBA sur $\mathcal{A}$. Cependant, en ajoutant des transitions à $\mathcal{A}$ pour respecter la propriété (5.5), il est possible de dériver un automate $\mathcal{A}'$ tel que $\mathcal{B}_4$ soit un $\wp$-UTGBA sur $\mathcal{A}'$.

Par conséquent, si l'algorithme de test de vacuité trouve une composante acceptante dans $\mathcal{B}_4$, alors il existe un chemin acceptant dans $\mathcal{A}$' *mais pas nécessairement* dans $\mathcal{A}$. Cependant, puisque les chemins de $\mathcal{A}$ sont aussi des chemins de $\mathcal{A}'$ alors si l'algorithme ne trouve pas de chemin acceptant dans $\mathcal{B}_4$ alors, *forcément*, il n'y aura pas de chemin acceptant dans $\mathcal{A}$, ni $\mathcal{A}'$.

Autrement dit, cet algorithme modifié retourne *"vide"* ou *"Je ne sais pas"*. Cette manière, *semi-décisionnelle*, de procéder s'avère très utile dans les cas où l'on est *presque sûr* de la véracité d'une propriété et que l'on veut une preuve rapide.

5.5 Les contre-exemples

Dans la vérification de modèles par la théorie des automates, la présence d'un chemin acceptant signifie l'existence d'une exécution dans le système étudié qui invalide la propriété testée, *i.e.*, *un contre-exemple* de la propriété. Par conséquent,

toutes les fois ou l'algorithme de test de vacuité se termine par $\perp$, signifiant que l'automate a un chemin acceptant, l'utilisateur a besoin de voir un tel chemin pour apporter les modifications qui s'imposent au système.

Dans un test de vacuité classique, basé sur des algorithmes de *recherche (imbriquée) en profondeur d'abord* sur des automates de Büchi non-généralisés [SE05], la pile de recherche fournit un chemin acceptant directement à la terminaison. Au contraire, le test de vacuité basé sur la recherche des SCC ne peut décrire le contre exemple qu'en termes de SCCs. Pour produire un contre-exemple réel à partir de la pile des SCCs, ces SCCs doivent être parcourues une autre fois [CDLP05]. Premièrement, on calcule un cycle acceptant à l'intérieur de la SCC la plus haute dans la pile de recherche, puis on essaye de l'atteindre à partir de l'état initial. Les recherches sont localisées, car les états à explorer appartiennent, nécessairement, aux SCC de la pile de recherche.

Notre algorithme produit une telle pile de SCCs, mais la recherche d'un chemin acceptant réel s'avère délicate à cause de l'utilisation des inclusions et des décompositions. L'algorithme décrit dans [CDLP05] ne peut être utilisé qu'avec la garantie que pendant sa nouvelle exploration des SCCs, il visite exactement les mêmes états que ceux que l'algorithme de test de vacuité a déjà visité. Dans notre cas, ceci est difficile à réaliser car le calcul des successeurs d'un état en utilisant l'inclusion et la décomposition dépend de la valeur de $H.keys()$, qui a évolué depuis la première visite par le test de vacuité. De plus, le contre-exemple généré est basé sur les états de $\mathcal{B}$, tandis que l'utilisateur est plutôt intéressé par un contre exemple basé sur les états de $\mathcal{A}$, plus simples à interpréter.

Notre proposition est qu'une fois le test de vacuité de $\mathcal{B}$ terminé (avec $\perp$), nous cherchons un contre-exemple dans $\mathcal{A}$, mais en utilisant les structures de données calculées par l'algorithme de test de vacuité de $\mathcal{B}$ pour contenir la recherche. Le calcul de l'ensemble d'états de chaque SCC est simplement réalisé en explorant les piles de recherche $todo$ et SCC : l'ensemble des états qui appartiennent à la SCC au sommet de la pile est $S_n = rem_n \cup \{state_{f(root_n)}, state_{f(root_n)+1}, \ldots, state_m\}$. L'ensemble des états qui appartiennent à la SCC suivante est $S_{n-1} = rem_{n-1} \cup \{state_{f(root_{n-1})}, state_{f(root_{n-1})+1}, \ldots, state_{f(root_n)-1}\}$, etc. Vérifier si un état $s \in Q_{\mathcal{A}}$ appartient à la $i^{\grave{e}}$ SCC revient à tester si $\exists S \in S_i$ tel que $s \in S$. C'est donc possible de contraindre la recherche dans $\mathcal{A}$ pour rester à l'intérieur de ces SCCs.

5.6 Correction de la construction du produit synchronisé quotient

En utilisant les notions introduites dans ce chapitre, nous pouvons à présent prouver que le produit synchronisé quotient que nous avons introduit dans la section 4.3 est équivalent au produit synchronisé ordinaire, du point de vue du test de vacuité. En fait, ceci revient à prouver que $\mathcal{B} = \overline{\mathcal{K}\overline{\otimes}\mathcal{A}_c \otimes \mathcal{A}_f}$ est un $\wp$-$\mathcal{UTGBA}$ sur $\mathcal{A} = \mathcal{K}\overline{\otimes}\mathcal{A}_c \otimes \mathcal{A}_f$. Par conséquent, nous avons l'équivalence de vacuité entre les deux automates ($\mathcal{A} \overset{\wp}{\equiv} \mathcal{B}$).

On rappelle que $\mathcal{A} = \mathcal{K}\overline{\otimes}\mathcal{A}_c \times \mathcal{A}_f = \langle Q_{\mathcal{A}}, Q^0_{\mathcal{A}}, \mathcal{R}_{\mathcal{A}}, \mathcal{F}\rangle$. Donc chaque état $\langle \mathcal{H}, O, l, q\rangle \in Q_{\mathcal{B}}$ représente l'ensemble $\{\langle x, l, q\rangle \in Q_{\mathcal{A}} \mid x \in O\}$ d'états de $\mathcal{A}$. Par conséquent, on peut écrire $\langle x, l, q\rangle \in \langle \mathcal{H}, O, l, q\rangle$, et nous prouvons que $\mathcal{B}$ est un $\wp$-UTGBA sur $\mathcal{A}$.

- Puisque chaque état $\langle \mathcal{H}, O, l, q\rangle \in Q_{\mathcal{B}}$ représente l'ensemble $\{\langle x, l, q'\rangle \in Q_{\mathcal{A}} \mid x \in O \wedge l \in L \wedge q' = q\}$, on a $Q_{\mathcal{B}} \subseteq 2^{Q_{\mathcal{A}}}$ et la propriété (5.1) est vérifiée trivialement.
- La propriété (5.2) est vérifiée par définition de $\mathcal{B}$.
- La propriété (5.3) est vérifiée car :

$$\bigcup_{S \in Q^0_{\mathcal{B}}} S = \{\langle x, l, q'\rangle \mid s \in \mathcal{S}_0, x \in \mathcal{G}.s, l \in \{l_0\}, q' \in Q^0_{\mathcal{A}_f}\} = (\mathcal{G}.\mathcal{S}_0) \times \{l_0\} \times Q^0_{\mathcal{A}_f} = \mathcal{S}_0 \times \{l_0\} \times Q^0_{\mathcal{A}_f} = Q^0_{\mathcal{A}}$$

- La propriété (5.4) est réécrite comme suit :

$$\forall \big\langle \langle s, l, q\rangle, F, \langle s', l', q'\rangle \big\rangle \in \mathcal{R}_{\mathcal{A}}, \forall \langle \mathcal{H}, O, l, q\rangle \in \text{Reach}(\mathcal{B}), \langle s, l, q\rangle \in \langle \mathcal{H}, O, l, q\rangle \Longrightarrow$$
$$\exists \langle \mathcal{H}', O', l', q'\rangle \in Q_{\mathcal{B}}, \langle s', l', q'\rangle \in \langle \mathcal{H}', O', l', q'\rangle, \big\langle \langle \mathcal{H}, O, l', q\rangle, F, \langle \mathcal{H}', O', l', q'\rangle \big\rangle \in \mathcal{R}_{\mathcal{B}}$$

  Soit $\big\langle \langle s, l, q\rangle, F, \langle s', l', q'\rangle \big\rangle \in \mathcal{R}_{\mathcal{A}}$. Par application de la définition de $\mathcal{A}$, il existe $\langle s, s'\rangle \in \mathcal{R}_{\mathcal{K}}$, $\langle l, l'\rangle \in \mathcal{R}_{\mathcal{A}_c}$ et $\langle q, q'\rangle \in \mathcal{R}_{\mathcal{A}_f}$ où $\Pi_{\mathcal{A}_f}(q, q') \subseteq \Pi_{\mathcal{K}}(s)$ et $\Gamma(s, s') \in \Delta(l, l')$. Soit $\langle \mathcal{H}, O, l, q\rangle \in \text{Reach}(\mathcal{B})$ tel que $\langle s, l, q\rangle \in \langle \mathcal{H}, O, l, q\rangle$. Par définition de $\mathcal{R}_{\mathcal{B}}$, il existe $\big\langle \langle \mathcal{H}, O, l, q\rangle, F, \langle \mathcal{H}', O', l, q'\rangle \big\rangle \in \mathcal{R}_{\mathcal{B}}$ où $O' = (\mathcal{H} \cap \mathcal{G}_{\Delta(l,l')} \cap \mathcal{G}_{\Pi_{\mathcal{A}_f}(q,q')}.s'$ et $\mathcal{H}' \subseteq \mathcal{G}_{O'}$. Alors $\langle s', l', q'\rangle \in \langle \mathcal{H}', O', l', q'\rangle$.

- Finalement, la propriété (5.5) est réécrite par :

$$\forall \big\langle \langle \mathcal{H}, O, l, q\rangle, F, \langle \mathcal{H}', O', l', q'\rangle \big\rangle \in \mathcal{R}_{\mathcal{B}}, \forall \langle s', l', q'\rangle \in \langle \mathcal{H}', O', l', q'\rangle, \text{alors}$$
$$\exists \langle s, l, q\rangle \in \langle \mathcal{H}, O, l, q\rangle, \text{tel que} \big\langle \langle s, l, q\rangle, F, \langle s', l', q'\rangle \big\rangle \in \mathcal{R}_{\mathcal{A}}$$

Soit $\big\langle\langle \mathcal{H}, O, l, q\rangle, F, \langle \mathcal{H}', O', l', q'\rangle\big\rangle \in \mathcal{R}_{\mathcal{B}}$, et $\langle s', l', q'\rangle \in \langle \mathcal{H}', O', l', q'\rangle$. Par définition de $\mathcal{R}_{\mathcal{B}}$, $\exists\big\langle\langle x, l, q\rangle, F, \langle x', l', q'\rangle\big\rangle \in \mathcal{R}_{\mathcal{K}}$ où $\langle x, l, q\rangle \in \langle \mathcal{H}, O, l, q\rangle$ et $\langle x', l', q'\rangle \in \langle \mathcal{H}', O', l', q'\rangle$.

Puisque $\langle x', l', q'\rangle$ et $\langle s', l', q'\rangle$ appartiennent à $\langle \mathcal{H}', O', l', q'\rangle$, il existe $g \in \mathcal{H}'$ tel que $g.x' = s'$. Puisque l'action de $\mathcal{H}' \subseteq \mathcal{G}$ est congruente par rapport à la relation de transition, on a $\big\langle\langle g.x, l, q\rangle, F, \langle g.x', l', q'\rangle\big\rangle = \big\langle\langle g.x, l, q\rangle, F, \langle s', l', q'\rangle\big\rangle \in \mathcal{R}_{\mathcal{A}}$.

5.7 Conclusion

Les automates de Büchi *ensemblistes* ($\wp$-UTGBA) sont des automates dont les états représentent des ensembles d'états (ordinaires). Ils peuvent être considérés comme des représentants compactes des automates ordinaires, dans lesquels l'intersection entre deux états n'est pas forcement vide.

Dans ce chapitre, et dans le cadre d'un processus de verification à- la-volée ("on the fly"), nous avons développé une approche générale qui exploite ces inclusions pour réduire la taille de ce type d'automates et par conséquent l'optimisation du processus de vérification.

Il est évident que l'approche n'aura un intérêt pratique que si la construction est automatique et efficace. Cette problématique est analysée en détaille dans les deux chapitres suivants.

Dans le chapitre 6, nous introduirons le modèle des réseaux de Petri Bien Formés. Dans ce cadre, nous étudierons la vérification des systèmes concurrents suivant une approche par symétries globales (aussi appelée, par symétries statiques) et nous montrerons notre contribution pour la mise en œuvre approches existantes.

Dans le chapitre 7, il sera question de l'application de nos approches générique dans le cade de ce formalisme, *i.e.*, la verification des systèmes concurrent suivant une approche par symétries locales (aussi appelée, par symétries dynamiques).

Chapitre 6

Les Réseaux de Petri Bien Formés : approches par symétries statiques

6.1 Introduction

Parmi les outils couramment utilisés pour modéliser les systèmes distribués et concurrents, les réseaux de Petri occupent une place privilégiée, d'une part parce qu'ils permettent de représenter facilement les synchronisations, d'autre part parce qu'ils offrent des outils d'analyse qualitative du système, en utilisant des approches *structurelles* et *comportementales*.

Si l'on étudie des systèmes conformes à la réalité, la taille du modèle rend sa représentation impossible. Par conséquent, on a recours à des modèles de haut niveau qui permettent une représentation plus concise du système. Cependant, diminuer la taille du modèle ne résout pas en général le problème de la complexité de l'analyse. En effet, les approches structurelles ne se transposent pas toujours au modèle de haut niveau. Quant aux approches comportementales, elles explorent un espace d'états dont la taille n'a pas été réduite par le changement de niveau de représentation.

Au cours des dernières décennies, différents modèles de réseaux de Petri de *haut-niveau* (HLPN, pour High Level Petri Nets) ont été introduits [GL81, Jen82]. Dans ces réseaux, les composants de même nature sont identifiés au moyen de couleurs ou de variables, et des fonctions sur les arcs permettent de spécifier le comportement des ces différents objets. Malheureusement, lorsque le réseau de haut-niveau est décrit de manière complètement générale, il est impossible de l'utiliser pour obtenir directement des résultats d'analyse. Par exemple, il n'existe pas d'algorithme pour calculer des réductions structurelles sur le modèle.

Pour obtenir une représentation compacte de l'espace d'états engendré, on a cherché à exploiter les propriétés de *symétrie* du modèle afin de définir des équivalences entre états. Cependant, quand le formalisme utilisé est celui des réseaux colorés généraux, alors la recherche de ces symétries s'avère difficile.

Cette difficulté a été surmontée par l'introduction des *réseaux réguliers* [Had87], dont les contraintes de description permettent de déterminer automatiquement les symétries admissibles (mais avec une perte en puissance d'expression par rapport aux HLPN). Ces symétries sont utilisées pour la construction automatique d'un graphe d'états symbolique qui est un graphe quotient du graphe d'accessibilité ordinaire, dont chaque nœud est une classe d'équivalence d'états.

Le modèle des réseaux de Petri bien Formés [Dut91] est une extension des réseaux réguliers, dont la principale avancée est son pouvoir d'expression équivalent à celui des HLPN.

Dans ce chapitre nous présenterons en détail les réseaux de Petri bien Formés[1], puis nous exposerons les limites de ce formalisme pour le traitement des *systèmes asymétriques*. Nous présenterons ensuite notre apport au formalisme pour le traitement de ces systèmes, en adaptant les méthodes génériques présentées dans les chapitres précédents.

6.2 Les Réseaux de Petri bien Formés

Les réseaux de Petri (RdP) sont des graphes orientés bipartis. Les deux types de nœuds les composants sont des places et des transitions. Ces éléments représentent respectivement l'état du système et les actions susceptibles de modifier cet état global. Les seuls arcs valides sont ceux reliant une place et une transition. Les arcs sont orientés et étiquetés par une valuation entière. Une place est dite en entrée d'une transition s'il existe un arc orienté de la place vers la transition. Réciproquement, une place dite est en sortie d'une transition s'il existe un arc allant de la transition à la place.

Les réseaux de Petri bien formés (WN, pour Well formed Petri Nets) sont une classe des HLPN dont le formalisme respecte une syntaxe simple et rigoureuse. Celle-ci permet une modélisation concise et paramètrée des systèmes, mais surtout une représentation implicite et une exploitation automatique de leurs symétries. En plus des composants classiques des RdP (les places et les transitions), la

[1] Par rapport à la version originelle, plusieurs notations et définitions ont été modifiées afin de faciliter la compréhension et simplifier l'introduction de nos méthodes.

définition des WN repose sur les trois concepts suivants : *les domaines de couleurs, les fonctions de couleurs et les prédicats standard*.

Les domaines de couleurs. Un domaine de couleurs exprime la notion mathématique de *domaine de définition*. Un domaine permet de typer les places et les transitions d'un WN. Il est défini par le produit cartésien d'un nombre quelconque d'ensembles, appelés *classes d'objets*.

`Les classes d'objets.` Une classe d'objets (classe pour simplifier) est un ensemble fini, regroupant les objets du système qui ont la même nature, par exemple, les processeurs, les processus, les sites, etc. Il est possible de munir une classe d'une fonction successeur, on parle alors de *classe ordonnée* (même si la fonction successeur ne définit pas une relation d'ordre sur la classe). Aussi, au sein d'une classe, il est souvent important de distinguer différents groupes d'objets suivant leurs caractéristiques comportementales, par exemple, les processeurs lents et les processeurs rapides. Ces groupes sont connus sous le nom de *sous-classes statiques*.

Formellement, soit $\mathcal{C}$ une famille de classes :

$$\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_h, \mathcal{C}_{h+1}, \ldots, \mathcal{C}_n\},$$

où $0 \leq h \leq n$, $i \neq i' \Rightarrow \mathcal{C}_i \cap \mathcal{C}_{i'} = \emptyset$ et
- $\mathcal{C}_i$, $0 \leq i \leq h$, sont des classes non ordonnées.
- $\mathcal{C}_i$, $h < i \leq n$, sont des classes ordonnées. À chaque classe ordonnée est associée une fonction successeur $\oplus$ vérifiant : $\forall c \in \mathcal{C}_i, \mathcal{C}_i = \bigcup_{k=1}^{|\mathcal{C}_i|} \{\oplus^k c\}$, où $\oplus^k$ dénote la k-composition de la fonction $\oplus$.

La partition d'une classe non-ordonnée $\mathcal{C}_i$ en n_i sous-classes statiques est définie par :

$$\mathcal{C}_i = \bigcup_{j=1}^{n_i} \mathcal{D}_{i,j}, \text{tel que} \begin{cases} \forall\, 0 < j \leq n_i, |\mathcal{D}_{i,j}| > 0, \\ \forall\, 0 < j \neq j' \leq n_i \Rightarrow \mathcal{D}_{i,j} \cap \mathcal{D}_{i,j'} = \emptyset. \end{cases} \tag{6.1}$$

L'ensemble $\{\mathcal{D}_{i,j}\}_{j=1\ldots n_i}$ est appelé *partition statique de* $\mathcal{C}_i$ et est noté $\mathfrak{Part}_i$. Plus globalement, nous noterons $\mathfrak{Part} = \{\mathfrak{Part}_i\}_{i=1,\ldots,n}$ le partitionnement statique de l'ensemble des classes de $\mathcal{C}$.

Dans le cas où $\mathcal{C}_i$ est une classe ordonnée, une contrainte supplémentaire est ajoutée : elle assure que les objets de chacune de ses sous-classes statiques soit conti-

gus (par rapport à la fonction successeur) et que l'ordre de numérotation de ces sous-classes statiques est compatible avec celui des objets :

$$\forall \mathcal{D}_{i,j} \in \mathfrak{Part}_i, \exists! c' \in \mathcal{D}_{i,j}, \oplus c' \in \mathcal{D}_{i,j\oplus 1}. \text{ Où } j \oplus 1 = (j \bmod n_i) + 1. \quad (6.2)$$

`Les domaines de couleurs.` Le produit cartésien définissant un domaine de couleurs peut être nul, ce qui est défini alors comme un ensemble réduit à une unique couleur, appelée *couleur neutre* (notée ε). Ce type de domaine est utilisé par exemple pour représenter des états booléens du système. Aussi, le produit cartésien peut se réduire à une unique classe, dans ce cas, une couleur est en fait un objet. Dans le cas général, la couleur est une association d'objets, éventuellement de même type.

Définition 6.1 (Domaine de couleur)**.** *Soit $I = \{1, \ldots, n\}$ l'ensemble des indices des classes. Soit $J = \sum_{i=1}^{n} e_i.i$, un multi-ensemble sur I. Un domaine de couleur $\mathcal{C}_J$ est défini par $\mathcal{C}_J = \prod_{i=1}^{n} \prod_{j=1}^{e_i} (\mathcal{C}_i)$, noté $\mathcal{C}_J = \prod_{i=1}^{n} \mathcal{C}_i^{e_i}$, un produit cartésien de classes. Un n-uplet d'objets $c_J \in \mathcal{C}_J$ est noté $c_J = \Pi_{i=1}^{n} \Pi_{j=1}^{e_i} c_i^j$ et appelé couleur.*

Définition 6.2 (Partition statique d'un domaine de couleur)**.** *Soit $J = \sum_{i=1}^{n} e_i.i$ un multi-ensemble sur I. Soit $\mathcal{C}_J$ un domaine de couleur. Alors $\mathfrak{Part}_J$, la partition statique de $\mathcal{C}_J$ est définie par :*

$$\mathfrak{Part}_J = \{\Pi_{i=1}^{n} \Pi_{j=1}^{e_i} \mathcal{D}_{i,\nu(i,j)} | 1 \leq \nu(i,j) \leq n_i\}$$

On associe à chaque couleur $c \in \mathcal{C}_J$, l'élément $\tilde{c} \in \mathfrak{Part}_J$ tel que :

$$c_i^k \in \mathcal{D}_{i,j} \Leftrightarrow \tilde{c}_i^k = \mathcal{D}_{i,j}.$$

Exemple 6.1. *La Fig. 6.1 présente un réseau de Petri Bien Formé. Dans ce WN, p_1, p_2 et p_3 sont des places. t est une transition. $\mathcal{C}$ est une famille de classes. $\mathcal{C}_1$ et $\mathcal{C}_2$ sont des classes d'objets non ordonnées. $\mathcal{C}_1$ est composé de l'unique sous classe statique $\mathcal{D}_{1,1}$ ($\mathfrak{Part}_1 = \{\mathcal{D}_{1,1}\}$), dont les éléments sont c_1 et c_2. $\mathcal{C}_2$ est partitionnée en deux sous classes statiques $\mathcal{D}_{2,1}$ et $\mathcal{D}_{2,2}$ ($\mathfrak{Part}_2 = \{\mathcal{D}_{2,1}, \mathcal{D}_{2,2}\}$). Le domaine de couleur de p_1 est $\mathcal{C}_1$, celui de p_2 est $\mathcal{C}_2$ et celui de p_3 est le produit cartésien $\mathcal{C}_1 \times \mathcal{C}_2$. Le domaine de couleur de la transition t est $\mathcal{C}_1 \times \mathcal{C}_2$. Les étiquettes $\langle x \rangle$ et $\langle y \rangle$ et $\langle x, y \rangle$ sur les arcs sont des fonctions de couleurs dont nous allons voir la définition dans le prochain paragraphe.*

$$\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2\}$$
$$\mathcal{C}_1 = \mathcal{D}_{1,1} = \{c_1, c_2\}$$
$$\mathcal{C}_2 = \mathcal{D}_{2,1} \cup \mathcal{D}_{2,2} \text{ tel que : } \mathcal{D}_{2,1} = \{c'_1, c'_2\},\ \mathcal{D}_{2,2} = \{c'_3, c'_4\}.$$

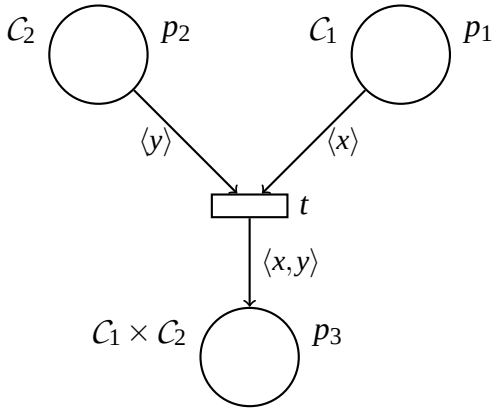


FIG. 6.1 – Exemple d'un WN

Les fonctions de couleurs. Une étude attentive des systèmes réels montre que les actions de base de leurs composants sont en nombre restreint. On cite :
– *l'identification* d'un objet du système, par exemple, un processus à mettre en attente.
– l'exécution *d'une diffusion (synchronisation)*, par exemple, un processus qui envoie un signal de terminaison à tous ses fils, ou une synchronisation de tous les processus avant de poursuivre l'exécution.
– l'exécution d'une opération basée sur *la notion d'ordre*, par exemple, envoyer un message au successeur sur un anneau.

La représentation WN de ces actions repose sur trois *fonctions de base* : la fonction *identité (projection)* ; la fonction *diffusion*, comme son nom l'indique, pour les diffusions ; la fonction successeur, pour prendre en compte la notion d'ordre.

Définition 6.3 (Les fonctions de base)**.** *Considérons le domaine de couleur $\mathcal{C}_J$ et la classe $\mathcal{C}_i$. Une fonction de base est l'une des fonctions suivantes :*

1. *la fonction identité : cette fonction sélectionne, dans un n-uplet de $\mathcal{C}_J$, la k^e occurrence d'un objet de la classe $\mathcal{C}_i$.*

$$X_i^k : \mathcal{C}_J \rightarrow Bag(\mathcal{C}_i)$$

$$X_i^k(\Pi_{j=1}^{n}\Pi_{q=1}^{e_j} c_j^q) = c_i^k$$

2. *la fonction diffusion : cette fonction sélectionne tous les objets de la sous-classe statique $\mathcal{D}_{i,q}$ de la classe $\mathcal{C}_i$.*

$$S_i^q : \mathcal{C}_J \rightarrow Bag(\mathcal{C}_i)$$

$$S_i^q(c_J) = \sum_{c' \in \mathcal{D}_{i,q}} c'$$

3. *la fonction successeur, uniquement pour $\mathcal{C}_i$ ordonnée : cette fonction sélectionne, dans un n-uplet $\in \mathcal{C}_J$, le successeur de la k^e occurrence d'un objet de la classe $\mathcal{C}_i$.*

$$\oplus X_i^k : \mathcal{C}_J \rightarrow Bag(\mathcal{C}_i)$$

$$\oplus X_i^k(\Pi_{j=1}^{n}\Pi_{q=1}^{e_j} c_j^q) = \oplus c_i^k$$

Si $\mathcal{C}_i$ n'est pas ordonnée, toute fonction dont le co-domaine est la classe $\mathcal{C}_i$ est une fonction $f_i : \mathcal{C}_J \rightarrow Bag(\mathcal{C}_i)$ définie par la combinaison linéaire des fonctions ci-dessus :

$$f_i = \sum_{q=1}^{n_i} \alpha_q \cdot S_i^q + \sum_{k=1}^{e_i} \beta_k \cdot X_i^k \tag{6.3}$$

Dans le cas où $\mathcal{C}_i$ est ordonnée, f_i est définie par :

$$f_i = \sum_{q=1}^{n_i} \alpha_q \cdot S_i^q + \sum_{k=1}^{e_i} (\beta_k \cdot X_i^k + \gamma_k \cdot \oplus X_i^k) \tag{6.4}$$

Où $\alpha_q, \beta_k, \gamma_k$ sont des entiers relatifs tels que $\forall c_J \in \mathcal{C}_J, \forall c_i \in \mathcal{C}_i, f_i(c_J)(c_i) \geq 0$.

Les fonctions de couleurs sont définies à partir de la généralisation des concepts précédents.

Définition 6.4 (Les fonctions de couleurs)**.** *Soit $\mathcal{C}_J$ et $\mathcal{C}_{J'}$ deux domaines. On définit la fonction de couleur $F : \mathcal{C}_J \rightarrow Bag(\mathcal{C}_{J'})$ par :*

$$F = \prod_{i=1}^{n}\prod_{j=1}^{e_i} f_i^j = \langle f_1^1, \ldots, f_1^{e_1}, \ldots, f_n^1, \ldots, f_n^{e_n} \rangle$$

$$\forall c \in \mathcal{C}_J, F(c) = \langle f_1^1(c), \ldots, f_1^{e_1}(c), \ldots, f_n^1(c), \ldots, f_n^{e_n}(c) \rangle$$

Où e_i représente le nombre d'occurrences de $\mathcal{C}_i$ dans $\mathcal{C}_{J'}$ et f_i^j est définie par 6.3 ou 6.4.

De manière générale une fonction de couleurs sur un arc entre une place p et une transition t a pour domaine le domaine de couleur de t et pour co-domaine l'ensemble des multi-ensembles sur le domaine de p.

Exemple 6.2. *Dans la Fig. 6.1, $\langle x \rangle$ est une simplification d'écriture pour la fonction de couleur $\langle X_1^1 \rangle$. De même, $\langle y \rangle$ est une simplification de $\langle X_2^1 \rangle$: $\langle x \rangle(c, c') = c$ et $\langle y \rangle(c, c') = c'$ pour $(c, c') \in \mathcal{C}_1 \times \mathcal{C}_2$. De manière générale, quand il n'y a pas de risque de confusion, les indices utilisés pour la déclaration des fonctions de couleur sont complètement omis.*

Les prédicats standard. Il est usuel que l'activation ou l'exécution d'un composant du système soit soumise à certaines contraintes. Par exemple, un serveur qui n'accepte de connexion que d'un ensemble particulier de clients et rejette toutes les autres demandes.

Les WN offrent un ensemble *de prédicats de base* qui, par leur composition linéaire, peuvent exprimer toute contrainte susceptible d'apparaître sur un composant.

Définition 6.5 (Les prédicats de base)**.** *Soit $\mathcal{C}_J$ un domaine de couleurs. L'ensemble des prédicats de base $\phi_J : \mathcal{C}_J \to \{vrai, faux\}$ est défini par :*

- $[X_i^{k'} = X_i^k](c_J) = (c_i^{k'} = c_i^k)$
- $[X_i^{k'} = \oplus X_i^k](c_J) = (c_i^{k'} = \oplus c_i^k)$
- $[d(X_i^k) = q](c_J) = (c_i^k \in \mathcal{D}_{i,q})$
- $[d(X_i^{k'}) = d(X_i^k)](c_J) = (\tilde{c}_i^{k'} = \tilde{c}_i^k)$
- $[Vrai](c_J) = (vrai)$.

Où $\mathcal{D}_{i,j} \in \mathfrak{Part}_i$.

Toute combinaison linéaire de ces prédicats par les opérateurs $\vee, \wedge$ et $\neg$ est autorisée :
- $(\phi_J \vee \phi'_J)(c_J) = \phi_J(c_J) \vee \phi'_J(c_J)$
- $(\phi_J \wedge \phi'_J)(c_J) = \phi_J(c_J) \wedge \phi'_J(c_J)$

– $(\neg\phi_J)(c_J) = \neg(\phi_J(c_J))$

L'association entre un prédicat ϕ_J et une fonction de couleurs F, définis tous deux sur $\mathcal{C}_J$, est possible et est définie par :

$$\forall c_J \in \mathcal{C}_J,\ \phi_J \cdot F(c_J) = Si\ \phi_J(c_J)\ alors\ F(c_J)\ sinon\ 0. \tag{6.5}$$

Cette association étend la notion de fonction de couleurs à celle de *fonction de couleurs gardée* et le nouvel ensemble de fonctions ainsi obtenu est connu sous le nom de *fonctions de couleurs standard*.

Exemple 6.3. *Reprenons le modèle de la Fig. 6.1. Pour restreindre le franchissement de la transition t aux éléments de la sous classe statique $\mathcal{D}_{2,1}$, nous associons à t le prédicat standard $[d(y) = 1]$ (y doit être instanciée dans la sous-classe statiques $\mathcal{D}_{2,1}$).*

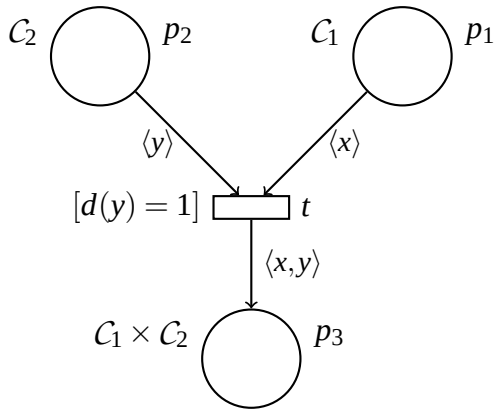


FIG. 6.2 – Exemple d'utilisation des prédicats standard

6.2.1 Définition formelle des WN

Nous donnons maintenant la définition formelle d'un WN.

Définition 6.6 (WN)**.** *Un réseau bien formé $WN =< P, T, \mathcal{C}, Cd, Pre, Post, Inh, \phi, \pi >$ est composé de :*

1. *P un ensemble fini des places,*
2. *T un ensemble fini des transitions, $P \cap T = \emptyset, P \cup T \neq \emptyset$,*
3. *$\mathcal{C}$ une famille des classe d'objets, $\mathcal{C} = \{C_1, \ldots, C_n\}$, telle que C_i est partitionnée en sous-classes statiques : $C_i = \bigcup_{q=1}^{n_i} \mathcal{D}_{i,q}$. On note $I = \{1, \ldots, n\}$*

l'ensemble ordonné des indices des éléments de $\mathcal{C}$ et la partition statique d'une classe $\mathcal{C}_i$ par $\mathfrak{Part}_i = \{\mathcal{D}_{i,q}\}_{q=1\ldots n_i}$. L'ensemble des partitions statiques de la famille de classes $\mathcal{C}$ est $\mathfrak{Part} = \{\mathfrak{Part}_i\}_{i=1,\ldots,n}$,

4. *$J : P \cup T \rightarrow Bag(I)$, où $Bag(I)$ est le multi-ensemble sur I. On note $Cd(r) = \mathcal{C}_{J(r)}$ le domaine de couleur de r.*[2]
5. *$Pre[p,t], Post[p,t] : Cd(t) \rightarrow Bag(Cd(p))$ sont les fonctions d'incidence avant et arrière, exprimées sous forme de fonctions de couleurs standard,*
6. *$Inh[p,t] : Cd(t) \rightarrow Bag(Cd(p))$, est la fonction d'inhibition, exprimée sous forme d'une fonction de couleurs standard,*
7. *$\phi_t : Cd(t) \rightarrow \{vrai, faux\}$, est le prédicat standard associé à la transition t,*
8. *$\pi : T \rightarrow \mathbb{N}$ est la fonction de priorité définie sur l'ensemble des transitions composant T.*

6.2.2 Marquage ordinaire d'un WN

Le marquage ordinaire (marquage, pour simplifier) m d'un WN représente un état global du système : c'est une fonction qui associe à chaque place p un multi-ensemble de $Cd(p) : m(p) \in Bag(Cd(p))$. Par conséquent, une place peut contenir plusieurs instances de la même couleur. Le marquage initial est noté m_0.

Exemple 6.4. *Sur le WN de la Fig. 6.2, nous pouvons définir le marquage initial $m_0 = p_1(c_1 + c_2) + p_2(c'_1 + c'_2 + c'_3 + c'_4)$, qui signifie que tous les objets de la classe $\mathcal{C}_1$ marquent la place p_1 et tous les objets de la classes $\mathcal{C}_2$ marquent la place p_2. L'écriture de m_0 est généralement simplifiée par $m_0 = p_1(S_1) + p_2(S_2)$. Le symbole S_i fait référence à la la fonction de diffusion sur la classe i : $S_i = \sum_{q=1}^{n_i} S_i^q$.*

6.2.3 Graphe de marquages d'un WN

La *règle de franchissement* d'un WN définit sa sémantique. Elle permet de construire, à partir d'un état (marquage) initial, un graphe dont chaque nœud représente un état du système. Les arcs de ce graphe représentent les événements dont l'occurrence permet de passer d'un état à un autre.

Ce graphe est appelé *graphe d'accessibilité (RG, pour Reachability Graph)* du WN et représente, du point de vue théorique, la $\mathcal{SKE}$ du système traité.

[2] Le domaine de couleur d'une transition t est un produit cartésien dans lequel le nombre d'occurrences d'une classe $\mathcal{C}_i$ est le nombre de variables X_i^k différentes, qui apparaissent dans l'ensemble des fonctions de couleurs étiquetant les arcs reliés à t.

Définition 6.7 (Instance d'une transition)**.** *On appelle instance de la transition $t \in T$, tout couple (t,c), tel que $c \in Cd(t)$.*

Définition 6.8 (Concession)**.** *Soit m un marquage et $t \in T$. On dit que l'instance (t,c) a une concession dans m ssi les trois points suivants sont vérifiés :*
- $\phi_t(c) = vrai$;
- $\forall p \in P, Pre[p,t](c) \leq m(p)$;
- $\forall p \in P, Inh[p,t](c) > m(p)$.

Définition 6.9 (Règle de franchissement)**.** *La règle de franchissement est définie par le test de franchissabilité et par le franchissement proprement dit :*
- *Une instance (t,c) est franchissable pour le marquage m, noté $m[(t,c)\rangle$, ssi (t,c) a une concession en m et il n'existe pas d'instance de transitions (t',c') telle que $\pi(t') > \pi(t)$ et (t',c') a une concession en m.*
- *Le franchissement de (t,c) à partir de m mène au marquage m', noté $m[(t,c)\rangle m'$, défini par : $\forall p \in P, m'(p) = m(p) + Post[p,t](c) - Pre[p,t](c)$.*

6.3 Le Graphe de Marquages Symboliques

Le graphe de marquage symbolique est une structure quotient du RG. il peut être vue comme une mise en œuvre du graphe quotient de la définition 3.3 ($\overline{\mathcal{K}} = (\overline{\mathcal{S}}, \overline{\mathcal{R}})$), appliquée au modèle des WN. Par conséquent, les symétries utilisées ici s'intègrent dans la catégorie des symétries globales.
Pour construire ce graphe quotient, il faut disposer d'une caractérisation, *au niveau du modèle*, du sous-groupe de permutations $\mathcal{G}$ de son graphe d'états RG. Pour des raisons d'efficacité, il faut en plus bénéficier d'une représentation compacte des classes d'équivalence (les éléments de $\overline{\mathcal{S}}$), que nous pourrons utiliser pour une construction automatique du graphe quotient.

Dans les WN, les symétries composant le groupe $\mathcal{G}$ sont appelées *symétries admissibles*, et la représentation compacte d'une classe de marquages est appelée *représentation symbolique* ou *marquage symbolique*.

6.3.1 Symétries admissibles et relation d'équivalence

Grâce à leurs domaines de couleurs et à la syntaxe de leurs fonctions de couleurs, les WN permettent une représentation implicite des symétries (le sous-groupe $\mathcal{G}$) du système étudié, par le biais des sous-classes statiques : les objets ayant des comportements similaires sont regroupés dans une même sous-classe statique. Par

conséquent, les symétries que nous pouvons définir sur les objets d'un modèle WN doivent préserver cette contrainte structurelle.Ainsi, nous avons une connaissance a priori de la relation d'équivalence qui lie les marquages.

Définition 6.10 (Les symétries admissibles)**.** *Les symétries admissibles d'un WN sont définies par* $\mathcal{G} = \{g = \otimes_{i=1}^{n} g_i\}$*, le sous-groupe de toutes les permutations sur* $\prod_{i=1}^{n} C_i$ *qui vérifie :*

- *Pour les classes non ordonnées,* $i \leq h$*,* g_i *est une* ***permutation*** *sur* C_i *telle que* $\forall \mathcal{D}_{i,q} \in \mathfrak{Part}_i$*,* $g_i.\mathcal{D}_{i,q} = \mathcal{D}_{i,q}$*.*
- *Pour les classes ordonnées,* $h < i \leq n$*,* g_i *est une* ***rotation*** *sur* C_i *telle que* $\forall\ \mathcal{D}_{i,q} \in \mathfrak{Part}_i$*,* $g_i.\mathcal{D}_{i,q} = \mathcal{D}_{i,q}$*. On note que cette condition implique que si* $n_i > 1$ *alors la seule rotation admise est l'identité.*

Note 6.1.

- *Les symétries admissibles d'un WN sont donc définies par un sous-groupe de permutations sur les objets des classes qui laissent les sous-classes statiques invariantes. Une conséquence importante de ceci est que* $\mathcal{G}$ *est totalement défini par la connaissance de la partition statique* $\mathfrak{Part}$ *de la famille de classes* $\mathcal{C}$ *et donc il est inutile de le représenter. Pour cette raison,* $\mathfrak{Part}$ *est qualifiée de* ***représentant implicite*** *de* $\mathcal{G}$*.*
- *On dit d'un WN qu'il est* ***totalement symétrique*** *ssi* $\forall C_i \in \mathcal{C}, |\mathfrak{Part}_i| = 1$*. Autrement dit, l'ensemble des objets de chaque classe forme une orbite par rapport à* $\mathcal{G}$*.*

Les symétries admissibles s'appliquent donc sur les classes, et naturellement sur les domaines de couleurs, mais aussi, par extension sur les marquages.

Soient C_J un domaine de couleur, $c_J \in C_J$ et $g \in \mathcal{G}$ alors :

$$g(\Pi_{i=1}^{n} \Pi_{j=1}^{e_i} c_i^j) = \Pi_{i=1}^{n} \Pi_{j=1}^{e_i} g_i.c_i^j$$

6.3.1.1 Équivalence de marquages

Soient m un marquage et $g \in \mathcal{G}$ une permutation, alors le marquage $g.m$ est défini par :

$$\forall p \in P, \forall c \in Cd(p), g.m(p)(\Pi_{i=1}^{n} \Pi_{j=1}^{e_i} (c_i^j)) = m(p)(\Pi_{i=1}^{n} \Pi_{j=1}^{e_i} (g_i^{-1}.c_i^j)) \quad (6.6)$$

Puisque $\mathcal{G}$ est un sous-groupe, on peut définir une relation d'équivalence entre marquages :

Définition 6.11 (Relation d'équivalence)**.** *Deux marquages, m et m', d'un WN sont équivalents ssi ils sont identiques à une symétrie près.*

$$m \sim m' \Leftrightarrow \exists g \in \mathcal{G},\ m' = g.m$$

$\sim$ *définit ainsi une partition de l'ensemble des marquages ordinaires du WN en classes d'équivalence.*

Un marquage qui reste invariant par l'application de n'importe quelle symétrie de $\mathcal{G}$ est dit *marquage symétrique*.

Définition 6.12 (Marquage symétrique)**.** *Un marquage m est dit symétrique ssi :*

$$\forall g \in \mathcal{G},\ m = g.m$$

6.3.1.2 Équivalence de franchissement

La relation d'équivalence définie entre les marquages d'une même classe s'étend au niveau des franchissements des transitions.

Proposition 6.1 (Equivalence de franchissement)**.** *La propriété de franchissement est préservée par l'application d'une permutation sur les marquages de départ et d'arrivée, ainsi que sur les couleurs instanciant la transition. Soient m, m' deux marquages, alors*

$$\forall t \in T, \forall c \in Cd(t),\ \forall g \in \mathcal{G},\ m[(t,c)\rangle m' \Longleftrightarrow g.m[(t,g.c)\rangle g.m'$$

Cette proposition énonce que tous les marquages appartenant à une même classe d'équivalence permettent les mêmes franchissements à une symétrie près. De plus, les marquages atteints à partir de ces franchissements sont aussi équivalents. Les preuves de cette proposition et de celle qui suit peuvent être trouvée dans le chapitre 4 de [Dut91].

Proposition 6.2 (Equivalence d'accessibilité)**.** *Si le marquage initial d'un WN est symétrique alors tous les marquages d'une classe d'équivalence sont accessibles.*

6.3.2 Représentation symbolique

Pour pouvoir construire automatiquement le graphe quotient de RG, il faut définir une représentation compacte pour les classes d'équivalence de marquages, ainsi qu'une règle de franchissement à partir de cette représentation.

La solution proposée par l'approche théorique de la section 3.2, et qui consiste à prendre un élément de la classe d'équivalence comme représentant, n'est pas satisfaisante. En effet, pour chaque nouveau marquage construit, il faudrait tester (à l'aide de la relation d'équivalence) si l'élément étudié est équivalent ou égal à un autre marquage, et ce test est très coûteux à réaliser.

La nouveauté apportée par les WN est une technique de *représentation symbolique* : pour chaque classe d'équivalence de marquages on calcule une *représentation générique unique*, appelée *Marquage Symbolique (SM, pour Symbolic Marking)*, qui référence l'ensemble des marquages de la classe. Grâce à cette représentation, il suffit d'opérer un simple test d'égalité entre deux SMs pour savoir s'ils représentent la même classe d'équivalence.

6.3.2.1 Exemple introductif

Considérons le WN de la Fig. 6.3. Les places p_1 et p_2 ont $\mathcal{C}_1$ comme domaine de couleurs. La définition de la classe d'objets $\mathcal{C}_1$ précise que les objets c_1, c_2 et c_3 ont le même comportement, alors que l'objet c_4 a un comportement différent (la partition statique de $\mathcal{C}_1$ est $\mathfrak{Part}_i = \{\mathcal{D}_{1,1}, \mathcal{D}_{1,2}\}$, comme défini dans la section 6.2). Si le marquage initial de ce WN est symétrique et que $m_1 = p_1(c_1 + c_2) + p_2(c_3 + c_4)$ est un marquage accessible, alors $m_2 = p_1(c_1 + c_3) + p_2(c_2 + c_4)$ et $m_3 = p_1(c_2 + c_3) + p_2(c_1 + c_4)$ sont aussi des marquages accessibles et les trois marquages forment la classe d'équivalence $M_{eq} = \{m_1, m_2, m_3\}$. Le défi est donc de trouver une représentation de M_{eq} sans avoir à en énumérer explicitement les éléments.

L'idée intuitive serait d'utiliser une notation ensembliste, qui consiste à remplacer les objets par des variables, et conserver la trace de leur appartenance aux sous-classes statiques, c.-à-d.,

$$M_{eq} = \{p_1(x_1 + x_2) + p_2(x_3 + y) |\, x_1, x_2, x_3 \in \mathcal{D}_{1,1} \wedge x_1 \neq x_2, x_1 \neq x_3, x_2 \neq x_3 \wedge$$

$$y \in \mathcal{D}_{1,2}\}$$

Cependant, nous constatons que distinguer x_1 et x_2 n'est pas nécessaire, car elles ont la même distribution dans les places du WN et appartiennent à la même sous-classe statique. Nous remplaçons donc nos variables d'objets par des *variables ensemblistes*, en précisant la cardinalité de chacune d'elles. Nous obtenons,

$$M_{eq} = \{p_1(E_1) + p_2(E_2 + E_3) |\, E_1, E_2 \subseteq \mathcal{D}_{1,1}, E_3 \subseteq \mathcal{D}_{1,2},\ |E_1| = 2, |E_2| = 1,$$

$$E_1 \cap E_2 = \emptyset, |E_3| = 1\}.$$

$\mathcal{C}_1 = \mathcal{D}_{1,1} \cup \mathcal{D}_{1,2}$ tel que $\begin{cases} \mathcal{D}_{1,1} = \{c_1, c_2, c_3\} \\ \mathcal{D}_{1,2} = \{c_4\} \end{cases}$ $\begin{array}{r} Z_1^1, Z_1^2 \in \mathcal{D}_{1,1} \text{ et } |Z_1^1| = 2, |Z_1^2| = 1 \\ Z_1^3 \in \mathcal{D}_{1,2} \text{ et } |Z_1^3| = 1 \end{array}$

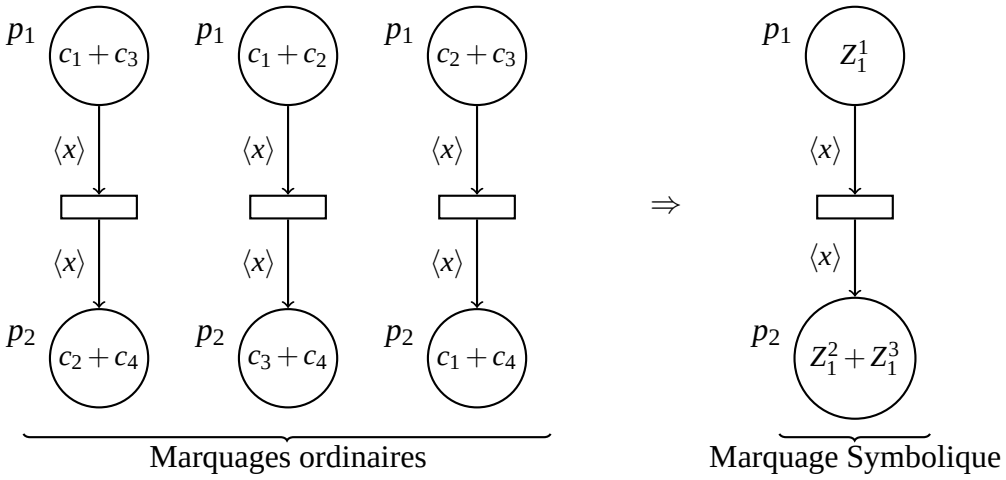


FIG. 6.3 – Exemple d'un marquage symbolique

Cette représentation est l'essence même de la représentation symbolique adoptée par les WN, elle porte le nom de *marquage symbolique* et est notée $\widehat{m}$. Les variables ensemblistes utilisées sont nommées *sous-classes dynamiques*, et notées Z_i^q : l'indice i identifie la classe $\mathcal{C}_i$ à laquelle appartiennent les objets représentés par la sous-classe dynamique, et l'indice q identifie la sous-classe dynamique parmi l'ensemble des sous-classes dynamiques de la classe $\mathcal{C}_i$. L'ensemble des sous-classes dynamiques d'une classe définit *une partition symbolique* temporaire des objets de la classe.

La classe M_{eq} peut être représentée par le marquage symbolique $\widehat{m}$:

$$\widehat{m} = p_1(Z_1^1) + p_2(Z_1^2 + Z_1^3) \text{ tel que } |Z_1^1| = 2, |Z_1^2| = |Z_1^3| = 1$$

6.3.2.2 Formalisation

Dans ce qui suit nous formalisons les concepts précédemment introduits.

Définition 6.13 (Partition symbolique d'une famille de classes)**.** *Soit* $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ *une famille de classes et* $\mathfrak{Part}$ *la partition statique de* $\mathcal{C}$*. On définit une partition symbolique de* $\mathcal{C}$ *par* $\mathfrak{Symb} = \{\mathfrak{Symb}_i\}_{i=1,\ldots,n}$ *telle que* $\mathfrak{Symb}_i = \langle \mathfrak{Dyn}_i, card_i, d_i \rangle$ *est la partition symbolique de la classe* $\mathcal{C}_i$*, avec :*

- $\mathfrak{Dyn}_i = \{Z_i^k\}_{k=1,\ldots,k_i}$ *est une représentation d'un ensemble de partitions de* $\mathcal{C}_i$. *Chaque élément* Z_i^k *est appelé sous-classe dynamique. Ainsi,* $\mathfrak{Dyn}_i$ *est appelé l'ensemble des sous classes dynamiques de la classe* $\mathcal{C}_i$.
- $card_i : \{1,\ldots,k_i\} \rightarrow \mathbb{N}^*$, *telle que* $card_i$ *est la fonction qui associe à une sous classe dynamique* Z_i^k *(de la classe i) sa cardinalité, i.e., le nombre d'objets de la classe qu'elle représente.*
- $d_i : \{1,\ldots,k_i\} \rightarrow \{1,\ldots,n_i\}$, *telle que :*
  1. $\mathcal{D}_{i,d_i(k)}$ *est la sous classe statique qui contient* Z_i^k, *i.e., la sous classe statique dans laquelle sont instanciées les objets représentées par la sous classe dynamique.*
  2. $\sum_{k|d_i(k)=j} card_i(k) = |\mathcal{D}_{i,j}|$. *Le nombre d'objets représentés par un ensemble de sous classes dynamiques, qui réfèrent la même sous classe statiques, doit être égal à la cardinalité de la sous classe statique.*
  3. $\forall k,k' \in \{1,\ldots,k_i\} : d_i(k) > d_i(k') \Rightarrow k > k'$. *L'indexation des sous classes dynamiques doit être compatible avec celle des sous classes statiques.*

Exemple 6.5. *Reprenons l'exemple précédent. La famille de classe* $\mathcal{C}$ *est constitué de l'unique classe* $\mathcal{C}_1 = \{c_1,c_2,c_3,c_4\}$. *La partition statique de cette classe est* $\mathfrak{Part}_1 = \{\mathcal{D}_{1,1},\mathcal{D}_{1,2}\}$ *avec* $\mathcal{D}_{1,1} = \{c_1,c_2,c_3\}$ *et* $\mathcal{D}_{1,2} = \{c_4\}$. *Ainsi,* $\mathfrak{Part} = \{\mathfrak{Part}_1\}$. *La partition symbolique de la classe* $\mathcal{C}_1$ *dans laquelle il y a trois parties : la première representant deux objets instanciés dans* $\mathcal{D}_{1,1}$ *; la deuxième représentant un objet instancé aussi dans* $\mathcal{D}_{1,1}$ *; et la troisième représentant un objet instancié dans* $\mathcal{D}_{1,2}$ *est définie par* $\mathfrak{Symb}_1 = \langle \mathfrak{Dyn}_1, card_1, d_1 \rangle$ *où :* $\mathfrak{Dyn}_1 = \{Z_1^1, Z_1^2, Z_1^3\}$ *;* $card_1(1) = 2$, $card_1(2) = 1$ *et* $card_1(3) = 1$ *;* $d_1(1) = 1$, $d_1(2) = 1$ *et* $d_1(3) = 2$. *Dans ce cas,* $\mathfrak{Symb} = \{\mathfrak{Symb}_1\}$ *est la partition symbolique de* $\mathcal{C}$.

Pour formaliser la notion de marquage symbolique, nous devons d'abord étendre la notion de partition symbolique d'une famille de classe aux domaines de couleurs.

Définition 6.14 (Partition symbolique d'un domaine de couleurs)**.** *Soit* $J = \sum_{i=1}^{n} e_i.i$, *un multi-ensemble sur* I. *Soit* $\mathfrak{Symb} = \{\mathfrak{Symb}_i\}_{i=1,\ldots,n}$, *tel que* $\mathfrak{Symb}_i = \langle \mathfrak{Dyn}_i, card_i, d_i \rangle$, *une partition symbolique de* $\mathcal{C}$. *On définit la partition symbolique du domaine de couleurs* $\mathcal{C}_J$ *par :* $\mathfrak{Symb}_J = \prod_{i=1}^{n} \prod_{j=1}^{e_i} (\mathfrak{Dyn}_i)$, *notée* $\mathfrak{Symb}_J = \prod_{i=1}^{n} (\mathfrak{Dyn}_i)^{e_i}$.

Définition 6.15 (Marquage symbolique)**.** *Soit un réseau bien formé* $WN = \langle P, T, \mathcal{C}, Cd, Pre, Post, Inh, \phi, \pi \rangle$. *Un marquage symbolique est un couple* $\langle \widehat{m}, \mathfrak{Symb} \rangle$ *tel que* $\mathfrak{Symb}$ *est une partition symbolique de* $\mathcal{C}$ *et* $\widehat{m}$ *est une fonction qui associe à chaque place* $p \in P$ *un multi-ensemble de* $\mathfrak{Symb}_{J(p)}$, $\widehat{m}(p) \in Bag(\mathfrak{Symb}_{J(p)})$.[3]

Exemple 6.6. *Le marquage symbolique* $\langle \widehat{m}, \mathfrak{Symb} \rangle$ *représentant l'ensemble des marquages ordinaires où : deux objets de la sous classe* $\mathcal{D}_{1,1}$ *marquent la place* p_1 *; le troisième objet de* $\mathcal{D}_{1,1}$ *marque la place* p_2 *; l'unique objet de* $\mathcal{D}_{1,2}$ *marque la place* p_2*, est défini en utilisant la partition symbolique* $\mathfrak{Symb}$ *de l'exemple précédent tel que :*

- $\widehat{m}(p_1) = Z_1^1$ *: p_1 est marquée par deux objets* $(card_1(1) = 2)$ *de la sous classe statique* $\mathcal{D}_{1,1}$ $(d_1(1) = 1)$.
- $\widehat{m}(p_2) = Z_1^2 + Z_1^3$ *: p_2 est marquée par un objet* $(card_1(2) = 1)$ *de la sous classe statique* $\mathcal{D}_{1,1}$ $(d_1(2) = 1)$ *et un objet* $(card_1(3) = 1)$ *de la sous classe* $\mathcal{D}_{1,2}$ $(d_1(3) = 2)$.

La définition suivante met en relation un marquage symbolique et la classe d'équivalence de marquages ordinaire qu'il représente.

Définition 6.16 (Marquage symbolique et représentation de classe de marquages)**.** *Un marquage symbolique* $\langle \widehat{m}, \mathfrak{Symb} \rangle$ *est une représentation d'une classe d'équivalence de marquages* M_{eq} *par rapport à la relation d'équivalence* $\sim$ *ssi :*

$\forall m \in M_{eq}, \exists \psi_i : \mathcal{C}_i \to \{1, \ldots, k_i\}$ *tel que :*

1. $|\psi_i^{-1}(k)| = card_i(k)$,
2. $\psi_i^{-1}(k) \subseteq \mathcal{D}_{i,d_i(k)}$,
3. $\forall p \in P, \forall c \in Cd(p), m(p)(\Pi_{i=1}^{n} \Pi_{j=1}^{e_i} c_i^j) = \widehat{m}(p)(\Pi_{i=1}^{n} \Pi_{j=1}^{e_i} Z_i^{\psi_i(c_i^j)})$,
4. $\forall\, h < i \leqslant n, \forall k \in \{1, \ldots, k_i\}, \exists! c \in \psi_i^{-1}(k), \exists k' \in \{1, \ldots, k_i\}$ *tel que* $d_i(k') = d_i(k) \oplus 1$*, vérifiant :* $\oplus c \in \psi_i^{-1}(k')$*. Où* $d_i(k) \oplus 1 = (d_i(k) \bmod n_i) + 1$.

On note $[\widehat{m}] = M_{eq}$.

Exemple 6.7. *Le marquage symbolique définit dans l'exemple précédent est une représentation symbolique de la classe d'équivalence de marquages* $M_{eq} =$

[3] On rappel que $J(p)$ est le domaine de couleur de p, suivant la définition 6.6.

$\{m_1, m_2, m_3\}$ *telle que* $m_1 = p_1(c_1+c_2)+p_2(c_3+c_4)$, $m_2 = p_1(c_1+c_3)+p_2(c_2+c_4)$ *et* $m_3 = p_1(c_2+c_3)+p_2(c_1+c_4)$, *car pour chacun de ces marquages, nous pouvons définir une fonction* ψ_1, *qui vérifient les conditions de la définition précédente.*
Soit $\psi_1 : \mathcal{C}_1 \rightarrow \{1,\ldots,3\}$, *telle que :* $\psi_1(c_1)=1; \psi_1(c_2)=1; \psi_1(c_3)=2; \psi_1(c_4)=3$. *Alors nous avons :*
– $|\psi_1^{-1}(1)| = card_1(1)$; $|\psi_1^{-1}(2)| = card_1(2)$; $|\psi_1^{-1}(3)| = card_1(3)$.
– $\psi_1^{-1}(1) = \{c_1, c_2\} \subseteq \mathcal{D}_{1,1}$; $\psi_1^{-1}(2) = \{c_3\} \subseteq \mathcal{D}_{1,1}$; $\psi_1^{-1}(3) = \{c_4\} \subseteq \mathcal{D}_{1,2}$;
– $m_1(p_1)(c_1) = \widehat{m}(p_1)(Z_1^1) = 1$; $m_1(p_1)(c_2) = \widehat{m}(p_1)(Z_1^1) = 1$; $m_1(p_2)(c_3) = \widehat{m}(p_2)(Z_1^2) = 1$; $m_1(p_2)(c_4) = \widehat{m}(p_2)(Z_1^3) = 1$;

Le même raisonnement est applicable pour m_2 *(avec,* $\psi_1(c_1)=1$; $\psi_1(c_3)=1$; $\psi_1(c_2)=2$; $\psi_1(c_4)=3$*), et* m_3 *(avec,* $\psi_1(c_2)=1$; $\psi_1(c_3)=1$; $\psi_1(c_1)=2$; $\psi_1(c_4)=3$*).*

Notations importantes. Vue l'importance et l'utilisation intensive des notions présentées dans cette section, nous alors définir quelques abréviations et notations utiles :
– La cardinalité, $card_i(k)$, d'une sous classe dynamique Z_i^k est notée $| Z_i^k |$. Une sous classe dynamique Z_i^k telle que $d_i(k) = j$ est notée $Z_{i,j}^k$. Par conséquent, une partition symbolique $\mathfrak{Smb}_i = \langle \mathfrak{Dyn}_i, card_i, d_i \rangle$ est totalement définie en utilisant ces abréviations. Par exemple, $\mathfrak{Smb}_1 = \langle \mathfrak{Dyn}_1, card_1, d_1 \rangle$ telle que $\mathfrak{Dyn}_1 = \{Z_1^1, Z_1^2\}$, $card_1(1) = 1$ et $card_1(2) = 2$, $d_1(1) = 1$, $d_1(2) = 2$, est totalement décrite par : $|Z_{1,1}^1| = 1$ et $|Z_{1,2}^2| = 2$.
– Nous noterons $\mathfrak{Smb}_i.\mathfrak{Dyn}_i$, $\mathfrak{Smb}_i.card_i$ et $\mathfrak{Smb}_i.d_i$ les différents éléments de la partition symbolique $\mathfrak{Smb}_i = \langle \mathfrak{Dyn}_i, card_i, d_i \rangle$. Cette notation est trivialement étendue sur $\mathfrak{Smb} = \{\mathfrak{Smb}_i\}_{i=1,\ldots,n}$.
– Un n-uplet de sous classes dynamiques $Z \in \mathfrak{Smb}_J$ est noté $Z = \Pi_{i=1}^{n} \Pi_{q=1}^{e_i} Z_i^{w(i,q)}$ tel que $1 \leqslant w(i,q) \leqslant |\mathfrak{Dyn}_i|$.
– Un marquage symbolique $\langle \widehat{m}, \mathfrak{Smb} \rangle$ est noté $\widehat{m}_{\mathfrak{Smb}}$ ou plus simplement $\widehat{m}$ (s'il n'y a pas de confusion sur la partition symbolique utilisée).

6.3.2.3 Représentation symbolique canonique

Sans contraintes supplémentaires, les définition précédentes laissent la possibilité d'avoir plusieurs représentations symboliques pour une même classe d'équivalence. Pour éviter cette situation, une forme unique, appelée *représentation canonique*, a été proposée dans [Dut91]. Cette forme est obtenue sur la base de propriétés *de minimalité et d'ordonnancement*.

La minimalité, obtenue par une opération de regroupement sur les sous classes dynamique, consiste à utiliser un minimum de sous classes dynamiques (en respectant les contraintes de la définition d'une partition symbolique).

L'ordonnancement consiste à établir un ordre lexicographique unique pour l'écriture du marquage. Ceci prend en compte, entre autres, l'ordre des places, et les indices des sous classes dynamiques.

Nous renvoyons le lecteur intéressé à la référence précédente pour une étude plus détaillée.

6.3.3 Règle de franchissement symbolique

Pour obtenir une construction automatique et directe du graphe quotient, une règle de tir symbolique est définie à partir d'un marquage symbolique. Le graphe construit est appelé *graphe des marquages symboliques* (SRG, pour Symbolic Reachability Graph).

Dans une représentation symbolique, les couleurs ne figurent plus et sont remplacées par des sous-classes dynamiques, qui peuvent chacune représenter plusieurs couleurs. Par conséquent, il n'est plus possible d'instancier un franchissement par des couleurs. Cette instanciation doit être effectuée par des n-uplets de sous-classes dynamiques. Il faut donc, pour tester la franchissabilité puis le franchissement, évaluer les fonctions de couleurs et les prédicats directement sur les sous-classes dynamiques.

L'*instanciation symbolique* nécessite l'introduction de deux fonctions supplémentaires, λ et μ. Étant donnée une transition t à instancier, la fonction λ sert à identifier la sous-classe dynamique dans laquelle est instanciée la $e^{ème}$ occurrence de C_i dans $Cd(t)$. Dans le cas où deux occurrences, e et e', de la même classe non ordonnée C_i sont instanciées dans la même sous-classe dynamique, on utilise la fonction μ pour savoir si les objets concernés par l'instanciation sont les mêmes ou non. Dans le cas où C_i est une classe ordonnée alors μ est utilisée pour désigner la position d'un objet dans la sous-classe dynamique choisie pour le franchissement.

Définition 6.17 (Instance symbolique)**.** *Soient t une transition telle que $Cd(t) = \prod_{i=1}^{n} C_i^{e_i}$. Soit $\mathfrak{Symb}$ une partition symbolique et $\widehat{m}_{\mathfrak{Symb}}$ un marquage symbolique. Soient $\lambda = \{\lambda_i : \{1,\ldots,e_i\} \to \{1,\ldots,|\,\mathfrak{Symb}_i\,.\,\mathfrak{Dyn}_i\,|\}$, $\mu = \{\mu_i : \{1,\ldots,e_i\} \to \mathbb{N}^*\}$. $(t,[\lambda,\mu])$ est une instance symbolique de t dans $\widehat{m}_{\mathfrak{Symb}}$, ssi $\forall i \in I, \forall 0 < k \leq e_i$:*
– $\mu_i(k) \leq \mathfrak{Symb}_i.card_i(\lambda_i(k))$,

– *Si* $i < h$ *alors* $\forall 0 < l < \mu_i(k), \exists k' < k$ *tel que* $\lambda_i(k') = \lambda_i(k) \wedge \mu_i(k') = l$.

Si $e_i = 0$ *alors* μ_i *et* λ_i *ne sont pas définis.*

Notation. L'instance symbolique $(t, [\lambda, \mu])$ est notée $(t, \hat{c})$ telle que : $\hat{c} = \prod_{i=1}^{n} \prod_{j=1}^{e_i} Z_i^{\lambda_i(j), \mu_i(j)}$. Où $(\prod_{i=1}^{n} \prod_{j=1}^{e_i} Z_i^{\lambda_i(j)})$ est un n-uplet de sous classes dynamiques de $\mathrm{S}\mathfrak{y}\mathfrak{m}\mathfrak{b}_{J(t)}$.

Le tir symbolique de l'instance symbolique $(t, [\lambda, \mu])$ à partir d'un marquage symbolique $\hat{m}$, noté $\hat{m}[[(t, [\lambda, \mu])\rangle\rangle$, est effectué en trois étapes : la division, le tir, puis la canonisation du marquage symbolique obtenu. Nous décrivons succinctement ces étapes et renvoyons le lecteur au chapitre 4 de la référence [Dut91] pour une étude détaillée.

Division d'un marquage symbolique. Pour analyser la franchissabilité d'une instance symbolique à partir d'un marquage symbolique, il faut réécrire ce dernier de manière à isoler symboliquement les objets sélectionnés par les fonctions λ et μ : dans chaque sous classe dynamique on crée autant de sous classes dynamiques unitaires qu'il y a d'objets différents utilisés pour le franchissement. La représentation obtenue est appelée *marquage symbolique divisé*.

Le tir symbolique. Une fois la division effectuée, un tir ordinaire est effectué, à la seule différence que les sous-classes dynamiques unitaires qui marquent les places en entrée d'une transition vont se substituer aux couleurs ordinaires.

Canonisation du marquage symbolique obtenu. Après un tir symbolique, il faut procéder à la canonisation du marquage symbolique obtenu, afin de garantir son unicité et effectuer le test d'existence d'une façon efficace.

La figure 6.4 illustre les différentes étapes du franchissement symbolique et le calcul d'un successeur symbolique canonique sur un exemple très simple. On note que l'étape de canonisation est représentée de façon décomposée : le regroupement puis l'ordonnancement. Dans cette figure, les notations $Z_1^{2,0}$ et $Z_1^{2,1}$ signifient que la sous classe dynamique Z_i^2 a été décomposée en deux sous classes dynamiques dans le marquage symbolique divisé, suivant les définition des fonctions λ et μ.

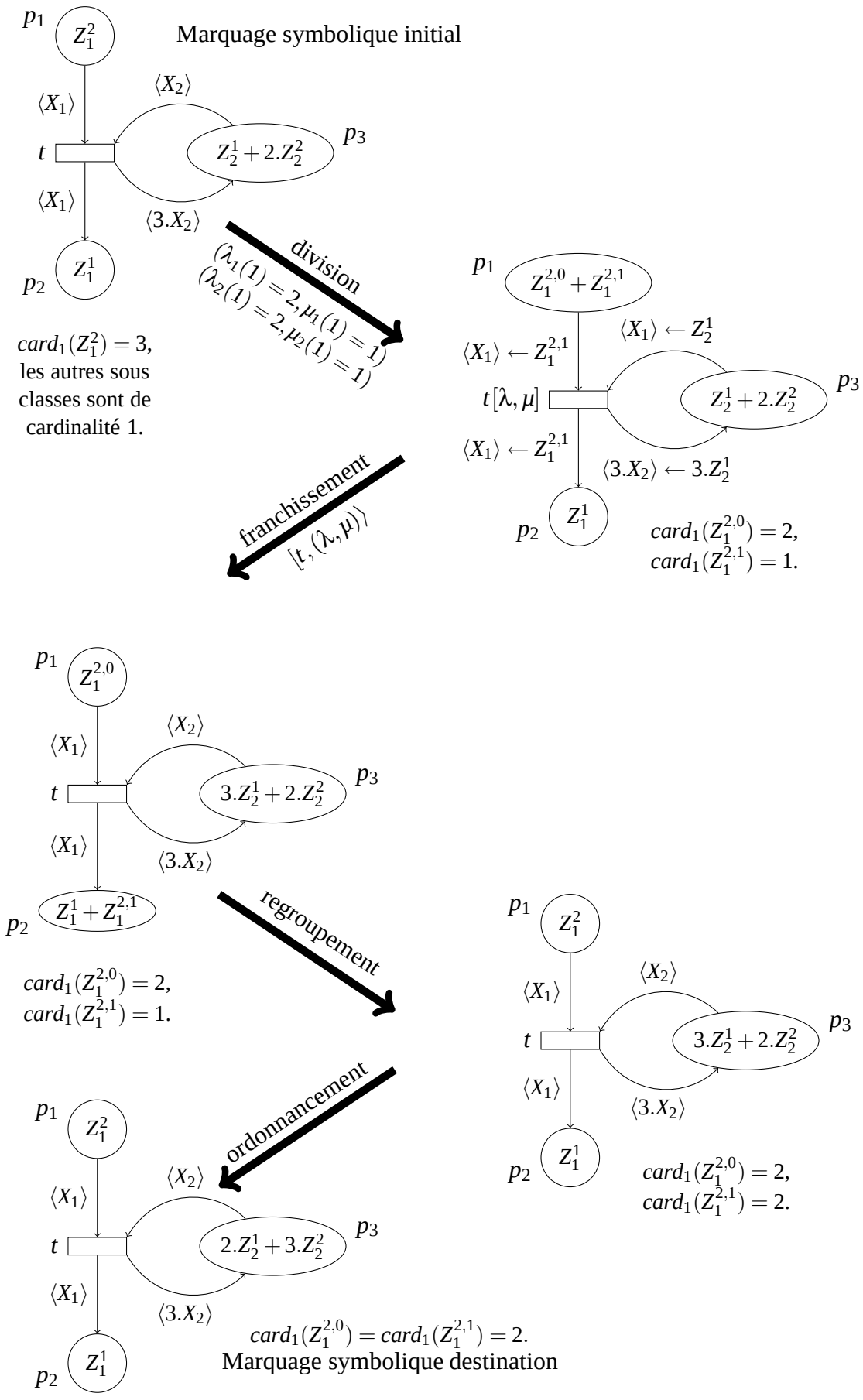


FIG. 6.4 – Exemple des étapes du franchissement symbolique.

6.4 Exemples de WN et limites de l'approche SRG

Dans cette section, nous étudierons les apports et les limites des WN au travers de l'étude d'un algorithme d'accès à une section critique dans un environnement distribué. Cet algorithme est basé sur un mécanisme de notifications. Quand un processus veut accéder à la section critique, il envoie une notification à tous les autres processus. Cet envoi est conditionné par : (1) l'absence de processus en section critique ; (2) l'absence de processus en attente de la section critique. Une fois la notification envoyée et reçue par tous les autres processus, le processus émetteur passe en mode attente. Il ne pourra accéder à la section critique que lorsque toutes les notifications ont été reçues, et que la section critique est libre.

La Fig. 6.5 représente une modélisation WN de cet algorithme. Le domaine de couleur de toutes les places est constitué de l'unique classe $C_1 = \{pr_1, \ldots, pr_n\}$, représentant l'ensemble des processus.

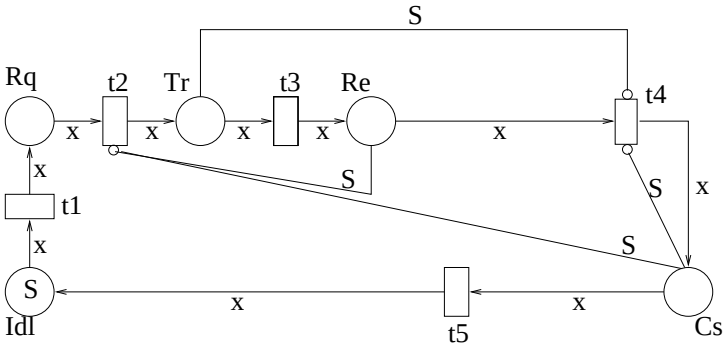
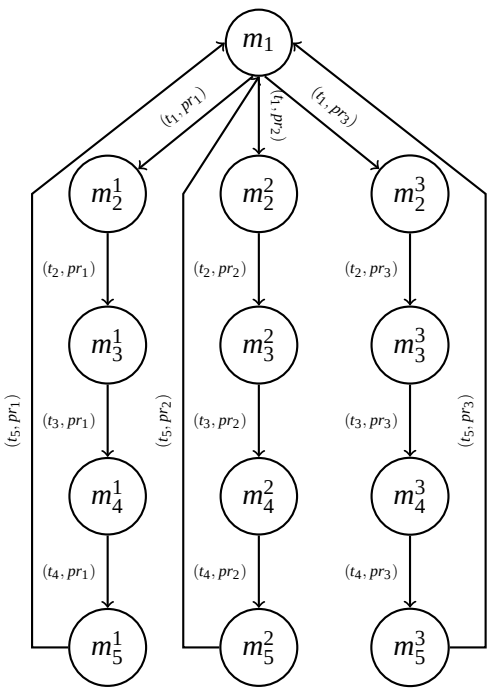


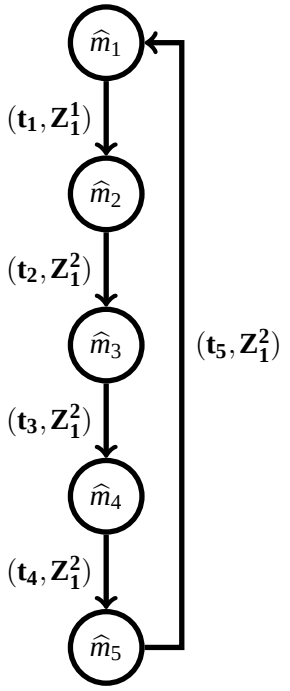
FIG. 6.5 – WN symétrique de l'algorithme d'accès à la section critique.

Prenons un processus *pr* dans C_1. Le marquage de la place Idl par *pr* représente les états où *p* effectue des actions en dehors de la section critique. Si *pr* marque *Rq* alors il exprime le besoin d'accéder à la section critique. Le marquage de *Tr* par *pr* correspond à l'émission de la notification, et son passage à *Re* signifie que la notification a été reçue, et par conséquent *pr* passe en mode attente. Le système étant distribué, plusieurs processus peuvent émettre des requêtes de manière concurrente et donc se retrouver simultanément en attente. Lorsque la section critique, *Cs*, est libre, le système opère un choix non déterministe parmi les processus en attente. Les arcs inhibiteurs issus de *Re* et de *Cs* représentent la contrainte, citée plus haut, sur l'émission des notifications.[4] L'arc inhibiteur issu de *Tr* permet de traiter toutes les notifications en transit avant d'autoriser un accès.

[4]un arc inhibiteur dont la fonction de couleur est *S* impose (par définition) que la place doit être vide pour que la transition soit franchissable.



(a) Une partie du RG



(b) Une partie du SRG

Marquage(s) ordinaire(s)	Marquage symbolique représentant
$m_1 = Idl(pr_1 + pr_2 + pr_3)$	$\widehat{m}_1 = Idl(Z_1^1)$ $\lvert Z_{1,1}^1 \rvert = 3$
$m_2^1 = Idl(pr_2 + pr_3) + Rq(pr_1)$ $m_2^2 = Idl(pr_1 + pr_3) + Rq(pr_2)$ $m_2^3 = Idl(pr_1 + pr_2) + Rq(pr_3)$	$\widehat{m}_2 = Idl(Z_1^1) + Rq(Z_1^2)$ $\lvert Z_{1,1}^1 \rvert = 2, \lvert Z_{1,1}^2 \rvert = 1$
$m_3^1 = Idl(pr_2 + pr_3) + Tr(pr_1)$ $m_3^2 = Idl(pr_1 + pr_3) + Tr(pr_2)$ $m_3^3 = Idl(pr_1 + pr_2) + Tr(pr_3)$	$\widehat{m}_3 = Idl(Z_1^1) + Tr(Z_1^2)$ $\lvert Z_{1,1}^1 \rvert = 2, \lvert Z_{1,1}^2 \rvert = 1$
$m_4^1 = Idl(pr_2 + pr_3) + Re(pr_1)$ $m_4^2 = Idl(pr_1 + pr_3) + Re(pr_2)$ $m_4^3 = Idl(pr_1 + pr_2) + Re(pr_3)$	$\widehat{m}_4 = Idl(Z_1^1) + Re(Z_1^2)$ $\lvert Z_{1,1}^1 \rvert = 2, \lvert Z_{1,1}^2 \rvert = 1$
$m_5^1 = Idl(pr_2 + pr_3) + Cs(pr_1)$ $m_5^2 = Idl(pr_1 + pr_3) + Cs(pr_2)$ $m_5^3 = Idl(pr_1 + pr_2) + Cs(pr_3)$	$\widehat{m}_5 = Idl(Z_1^1) + Cs(Z_1^2)$ $\lvert Z_{1,1}^1 \rvert = 2, \lvert Z_{1,1}^2 \rvert = 1$

FIG. 6.6 – Marquages du RG et du SRG du WN de la Fig. 6.5

On remarque que sur ce WN on ne fait aucune distinction entre les comportements des processus : aucune référence aux identités des processus n'est faite sur les composants du WN. Par conséquent, ce WN est *totalement symétrique* : $\mathfrak{Part}_1 = \{\mathcal{D}_{1,1}\}$ telle que $\mathcal{D}_{1,1} = \{pr_1, \ldots, pr_n\}$ (tous les processus sont indiscernables et sont initialement dans la même place). Ainsi, l'application de l'approche SRG apporte une réduction maximale par rapport au RG correspondant. Par exemple, pour $|\mathcal{C}_1| = 3$, le SRG contient 26 nœuds alors que le RG en a 91.

Le fonctionnement qui consiste à faire évoluer un seul processus jusqu'à l'obtention de la section critique (puis sa libération) est illustré dans la Fig. 6.6 sous la forme d'un graphe de marquages ordinaires (*RG*) et sous la forme d'un graphe de marquages symboliques (*SRG*). Les $\widehat{m}_i$ sont des marquages symboliques représentant des classes d'équivalence de marquages ordinaires. Par exemple, $\widehat{m}_2$ représente l'ensemble $\{m_2^1, m_2^2, m_2^3\}$ de marquages ordinaires. Par ailleurs, chaque franchissement sélectionne un unique objet, donc l'instance symbolique est définie par le couple $(\lambda_1(1), \mu_1(1))$. Par définition $\mu_1(1) = 1$, donc on représente l'instance symbolique par $(t_i, Z_1^{\lambda_1(1)})$. Aussi, les couples (t_i, Z_1^j) sont des instances de franchissement symbolique, représentant des classes d'équivalence d'instances ordinaires. Par exemple, (t_1, Z_1^1) représente l'ensemble $\{(t_1, pr_1), (t_1, pr_2), (t_1, pr_3)\}$ d'instances ordinaires. Ces représentations sont calculées suivant les méthodes introduites dans les sections 6.3.2 et 6.3.3.

A présent, nous voulons augmenter l'algorithme précédent par une gestion déterministe des conflits entre les processus en attente de la section critique. Ici, nous considérons une gestion basée sur les identités des processus : $i > j \Rightarrow pr_i$ a priorité sur pr_j. Par exemple, si pr_1, pr_2 et pr_3 sont en conflit, alors pr_3 obtiendra en premier la section critique, puis pr_2 et enfin pr_1.

La Fig. 6.7 représente une modélisation possible du nouvel algorithme, partant de celle présentée dans la Fig. 6.5.

Dans le cas d'un conflit, tous les processus impliqués dans le conflit finissent par marquer *Re*. La sélection s'opère avec l'arrivée d'un candidat dans *Sl*, et si ce candidat n'est pas le plus prioritaire, il est remplacé. Les franchissements successifs de t_6 conduisent à repousser les demandes en attente des processus les moins prioritaires. t_4 moins prioritaire[5] ne devient franchissable que lorsque t_6 ne l'est plus, ce qui se produit quand le prédicat, $[x < y]$, est évalué à faux. La syntaxe des WN exige que le prédicat $[x < y]$ soit modélisé par : $\bigvee_{i<j}([d(x) = i] \bigwedge [d(y) = j])$.

[5]La priorité de t_6 par rapport au autres transitions est exhibée dans la figure par sa couleur différente (noire).

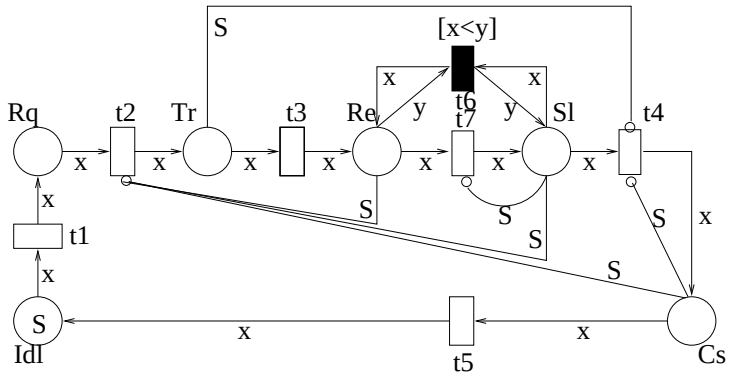


FIG. 6.7 – Modélisation d'un algorithme distribué d'accès à une section critique, avec priorité.

$\mathcal{D}_{1,1}, \ldots, \mathcal{D}_{1,n}$ sont des sous-classes statiques élémentaires (un objet par sous-classe) de la classe $\mathcal{C}_1$.

A cause de cette séparation élémentaire des objets, la seule symétrie autorisée sur le WN est l'identité ($\mathcal{G} = \{id\}$), et ceci implique que la taille du SRG de ce nouveau modèle est la même que celle du RG : *aucune réduction n'est possible par cette méthode*.

Ainsi, la modification de protocole considérée produit un système *totalement asymétrique*. L'application de l'approche *SRG* à ce système n'est donc pas propice car elle nécessite d'exprimer des symétries globales (section 3.2), ici, réduites à l'identité.

Bilan. Nous pouvons conclure que les WN sont un formalisme de modélisation très expressif, dont le graphe quotient est construit automatiquement, mais cette construction n'est efficace que pour les systèmes exhibant une forte symétrie globale.

Sachant qu'un des moyens prometteurs pour une analyse efficace de ce type de systèmes est de faire appel aux approches par symétries locales (chapitres 4 et 5), notre défi est donc de développer des approches qui tirent profit de la puissance des WN, tout en exploitant les symétries locales des systèmes étudiés.

6.5 Graphe des Marquages Symboliques Étendus

La première approche, [HITZ95], qui traite les systèmes asymétriques, modélisés par un WN, est une mise en œuvre de la méthode générique présentée dans la section 3.5. Le point crucial dans cette dernière est de déterminer *la sous-structure symétrique* $\mathcal{K}_s$ d'une structure de Kripke $\mathcal{K}$ et ce, *en opérant au niveau du modèle*. En d'autres termes, sur un WN (modélisant un système asymétrique), il faut identifier formellement les parties symétrique et asymétrique de manière à produire un graphe quotient dont la représentation du fonctionnement de la partie symétrique est "totalement" réduite. Le graphe construit par cette approche est appelé *graphe des marquages symboliques étendus (ESRG, pour Extended Symbolic Reachability Graph)*

Nous apportons deux critiques à la méthode originelle. La première est un problème de forme, car la présentation de la méthode a nécessité l'introduction de notions et notations, qui ne sont (à notre avis) que purement techniques, puisqu'elles ne concernent que l'implémentation de la méthode. Le deuxième problème est plus fondamental, car il concerne l'algorithme de construction. Ce dernier est composé de deux phases : la première construit des nœuds du graphe quotient, dont certains ne sont pas forcement accessibles et la deuxième élimine les nœuds non accessibles.

Ici, nous donnons une nouvelle présentation de la méthode et nous proposons un algorithme de construction optimisé, qui n'a besoin que d'une seule phase [6].

6.5.1 Caractérisation formelle des asymétries d'un WN

Dans les WN, les comportements asymétriques sont dus à l'existence de *transitions asymétriques*, c'est-à-dire, les transitions dont la franchissabilité dépend directement des identités des objets qui la sensibilisent.

Définition 6.18 (Transition asymétrique)**.** *Soit t une transition d'un WN, F_t l'ensemble des fonctions de couleurs des arcs adjacents à t, P_t l'ensemble des prédicats associés aux fonctions de F_t, ainsi que la garde de t. La transition t est asymétrique ssi l'une des deux conditions suivantes est vérifiée :*

1. *$\exists f \in F_t$, telle que f contient une fonction de base de la forme S_i^q (voir déf. 6.3).*

[6] Ces améliorations ont permit la réalisation de l'approche présentée dans [BFBI06]

2. *$\exists pr \in P_t$, tel que pr contient un prédicat de base de la forme $[d(X_i^k) = q]$ ou $[d(X_i^{k'}) = d(X_i^k)]$ (voir déf. 6.5).*

L'ensemble des transitions T d'un WN peut être décomposé en deux sous-ensembles T_{asym} et T_{sym} tels que : T_{asym} est l'ensemble des transitions asymétriques et $T = T_{asym} \uplus T_{sym}$. Si l'ensemble T_{asym} est vide, alors chacune des classes d'objets du réseau ne contient qu'une unique sous-classe statique et le réseau est dit *complètement symétrique*. Dans le cas contraire, il est *asymétrique*.

Exemple 6.8. *Dans le WN de la Fig.6.7, la transition t_6 est asymétrique car elle est gardée par le prédicat $[x < y]$, dont la syntaxe en WN est $\bigvee_{i<j}([d(x) = i] \bigwedge [d(y) = j])$. Toutes les autres transitions sont symétriques.*

Du point de vue de la méthode générique (section 6.5 du chapitre 3), la $\mathcal{SK}$ (RG) du sous-réseau engendré en ignorant les transitions asymétriques correspond à la structure $\mathcal{K}_s$. Ici, le sous-groupe de la définition 6.10 implémente celui de la méthode générique et nous définissons dans ce qui suit l'implémentation de $\mathcal{G}_s$.

Définition 6.19 (Les symétries relâchées)**.** *Soit $\mathcal{G}_s = \{g = \otimes_{i=1}^n g_i\}$ le sous-groupe de permutations sur $\prod_{i=1}^n \mathcal{C}_i$ tel que :*
– *Pour les classes non ordonnées, $i \leq h$, g_i est une **permutation** sur $\mathcal{C}_i$.*
– *Pour les classes ordonnées, $h < i \leq n$, g_i est une **rotation** sur $\mathcal{C}_i$.*

$\mathcal{G}_s$ est appelé sous-groupe relâché sur $\prod_{i=1}^n \mathcal{C}_i$.

La conséquence directe de cette nouvelle définition est la considération de deux relations d'équivalence entre les marquages du WN :
– la première est la relation $\sim$ de la définition 6.11. C'est la relation classique des WN, induite par le sous-groupe $\mathcal{G}$: si $\mathcal{C}$ est la famille des classes du modèle, alors $\mathcal{G}$ est implicitement représenté par $\mathfrak{Part}$, la partition statique de $\mathcal{C}$;
– la seconde est la relation $\sim_s$, formalisée de manière identique à celle de la définition 6.11, à la différence qu'elle est induite par $\mathcal{G}_s$: dans ce cas, la partition statique de $\mathfrak{Part}$ est remplacée par la partition $\mathfrak{Part}^{\mathfrak{s}} = \{\mathfrak{Part}^{\mathfrak{s}}{}_i\}_{i=1,\ldots,n}$ (représentant implicite de $\mathcal{G}_s$). Dans cette partition et pour chaque classe i, tous les objets sont regroupés dans une seule partie : $\forall 1 \leq i \leq n, |\mathfrak{Part}^{\mathfrak{s}}{}_i| = 1$. $\mathfrak{Part}^{\mathfrak{s}}$ est appelée *partition relâchée* de la famille de classes $\mathcal{C}$.

6.5.2 Représentation symbolique étendue

Pour la représentation symbolique des classes d'équivalence de marquages, nous sommes tentés de réutiliser celle présentée dans la section 6.3.2. Cependant, la co-existence de deux relations d'équivalence, $\sim$ et $\sim_s$, et par conséquent deux types de classes d'équivalence, nécessite une adaptation de cette représentation de manière à associer *explicitement* à chaque marquage symbolique la relation à laquelle il fait référence.

Dans la définition 6.15, $\hat{m}_{\mathfrak{Symb}}$ est un marquage symbolique qui fait *implicitement* référence à une relation d'équivalence représentée par la partition statique (unique) $\mathfrak{Part}$ (associée au WN). À présent, ceci n'est plus suffisant pour décrire un marquage symbolique, car il y a deux partitions de $\mathcal{C}$, la partition statique $\mathfrak{Part}$ et la partition relâchée $\mathfrak{Part}^{\mathfrak{s}}$, qu'il faut rajouter explicitement à la définition d'un marquage symbolique pour identifier exactement les symétries sur lesquelles est bâtie la classe d'équivalence qu'il représente. Nous aurons donc deux représentations symboliques :

– $\langle \mathfrak{Part}, \hat{m}_{\mathfrak{Symb}} \rangle$: c'est la représentation symbolique classique des WN, appelée marquage symbolique (*SM*). Elle fait référence à la partition statique classique des WN.
– $\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^{S}_{\mathfrak{Symb}^{\mathfrak{s}}} \rangle$: c'est une représentation qui fait référence à la partition relâchée, définie par rapport à $\mathcal{G}_S$. Elle sera appelée *marquage symbolique relâché (RSM, pour Relaxed Symbolic Marking)*.

6.5.3 Franchissement symbolique étendu

La co-existence des deux types de représentations symboliques, SM et RSM, impose deux types de franchissement symbolique :

– le *franchissement instancié*, à partir d'un SM et resultant en un autre SM. Il correspond au franchissement symbolique classique des WN, défini dans la section 6.3.3.
– Le *franchissement générique*, à partir d'un RSM et résultant en un autre RSM : cette règle est semblable à celle définie dans la section 6.3.3. Cependant, les seules transitions concernées par ce franchissement sont les transitions symétriques et les instances de franchissement symbolique sont définies en considérant la partition relâchée, $\mathfrak{Part}^{\mathfrak{s}}$.

Comme l'instanciation symbolique est construite en utilisant la partition statique du marquage symbolique source, la partition se propage au marquage symbolique

destination. C'est pourquoi un (R)SM mène vers un autre (R)SM. Les deux types de franchissements sont regroupés sous l'appellation *franchissement symbolique étendu*.

6.5.4 Algorithme de construction de l'ESRG

L'ESRG est un graphe quotient du RG dont les nœuds sont des RSM et des SM, et les arcs correspondent à des franchissements symboliques étendus, reliant ces représentations. Pour la construction de ce graphe, on est tenté d'utiliser un algorithme semblable à celui utilisé pour la construction du SRG. C'est-à-dire, à partir d'un (R)SM initial, on construit tous les (R)SM successeurs, par l'application de la règle de franchissement symbolique étendu. Pour chacun de ces successeurs, on calcule ses successeurs et on répète cette opération jusqu'à ce qu'il n'y ait plus de nouveau nœud à ajouter au graphe. Cependant, un tel algorithme est susceptible d'introduire des redondances dans la construction. En effet, sans contrainte de construction supplémentaire, il est possible qu'un ensemble de marquages ordinaires soit représenté à la fois par un RSM et par un ensemble de SM. Pour éviter ce cas de figure, on privilégie les franchissements à partir des transitions symétriques (franchissements génériques). Ceci permet une construction au plus tôt des RSM, évitant ainsi la construction des SM quand elle n'est pas nécessaire. Cette nécessité apparaît dans les deux cas suivants : (1) le RSM représentant les marquages ordinaires du SM à rajouter n'existe pas dans le graphe ; (2) il y a des franchissements asymétriques à partir du SM à rajouter.

Aussi, pour traiter les transitions asymétriques et optimiser la construction, il est nécessaire de disposer d'opérations symboliques qui permettent de passer d'un RSM à l'ensemble des SM qu'il représente et inversement : nous appelons *Raffinement (symbolique)* l'opération qui permet de passer d'un RSM $\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^{S}_{\mathfrak{Symb}^{\mathfrak{s}}} \rangle$ à l'ensemble $E = \{\langle \mathfrak{Part}, \hat{m}^{i}_{\mathfrak{Symb}^{i}} \rangle\}_i$ des SM qu'il représente. Son dual est appelée *Regroupement (symbolique)*. Dans cette partie, nous considérons simplement l'existence de ces deux opérations, laissant leur généralisation et étude détaillée à la section 7.3.2.

Pour simplifier la présentation de l'algorithme de construction, nous avons omis les partitions dans la description des représentations symboliques, et nous avons utilisé la fonction *Type* pour identifier la nature d'un marquage symbolique : RSM ou SM.

La procédure *TestSaturation* teste si le SM ajouté permet de compléter une classe d'équivalence d'états. Si de plus aucun des SMs de la classe n'autorise de fran-

chissement asymétrique, alors cet ensemble de SMs peut être substitué par un RSM représentant.

La procédure *TraitTransSym* traite la franchissabilité et le franchissement des transitions symétriques. Dans cette procédure, on insère les RSM en début de liste (ligne 13) afin de privilégier les franchissements symétriques.

La procédure *DévelopperESMs* constitue le cœur de l'algorithme de construction. Elle assure que tous les RSM ont été traités en priorité, et n'autorise l'ajout d'un nouveau SM dans le graphe que si : (1) il produit des franchissements asymétriques (lignes 23-26) ; (2) l'ensemble des marquages qu'il représente n'est pas déjà représenté par un RSM (lignes 35-36) ; (3) son ajout ne permet pas la création d'un RSM représentant (lignes 37-45), si c'est le cas, les lignes 41-43 permettent de réduire la taille du graphe : l'ensemble E des SM est remplacé par un seul RSM (les marquages ordinaires représentés par E forment une classe d'équivalence par rapport à $\mathcal{G}_s$). Enfin, on note que si les marquages représentés par un SM sont déjà représentés par un RSM alors il faut traiter uniquement les franchissements asymétriques à partir de ce SM, car les transitions symétriques auront été traitées au niveau du RSM. Sinon, toutes les transitions (symétriques et asymétriques) sont à considérer pour ce SM (lignes 28-31).

Données :
$ESRG$: le graphe de marquages symboliques étendus
$Atraiter$: une liste de marquages symboliques étendus

```
1  TestSaturation(m̂) :
2  début
3  |  si ∃E = {m̂'|m̂' ∈ ESRG.Nœuds() ∧ Type(m̂') = SM} et m̂ ∈ ATraiter
   |  tel que ∪_{m̂'∈E∪{m̂}} [m̂'] est une classe d'équivalence p.r.a. G_s et
4  |  ∄t ∈ T_asym franchissable à partir de e ∈ E ∪ {m̂} alors
5  |  |  retourner E
6  |  retourner ∅
7  fin

8  TraitTransSym(m̂) :
9  début
10 |  pour tous les t ∈ T_sym tel que m̂[(t,ĉ)⟩m̂' faire
11 |  |  si (m̂' ∉ ESRG.Nœuds()) alors
12 |  |  |  ESRG.AjouterNœud(m̂')
13 |  |  |  ATraiter.InsérerEnTête(m̂')
14 |  |  ESRG.AjouterArc(m̂ --(t,ĉ)--> m̂')
15 fin
```

```
16 DévelopperESMs() :
17 début
18     tant que ATraiter ≠ ∅ faire
19         m̂ :=ATraiter.RetirerTête()
20         si Type(m̂)=RSM alors
21             TraitTransSym(m̂)
22             L :=Raffinement(m̂)
23             si {m̂′ ∈ L|∃t ∈ T_asym, m̂′[(t,ĉ)⟩} ≠ ∅ alors
24                 pour tous les m̂′ ∈ L faire
25                     ESRG.AjouterNœud(m̂′)
26                     ATraiter.InsérerEnQueue(m̂′)
27         sinon
28             si ∃m̂′ ∈ ESRG.Nœuds() tel que Type(m̂′)=RSM et [m̂] ⊆ [m̂′]
               alors
29                 Trans :=T_asym
30             sinon
31                 Trans :=T_asym ⊎ T_sym
32             pour tous les t ∈ Trans tel que m̂[(t,ĉ)⟩m̂′ faire
33                 si m̂′ ∈ ESRG.Nœuds() alors
34                     ESRG.AjouterArc(m̂ --(t,ĉ)--> m̂′)
35                 sinon si ∃ m̂″ ∈ ESRG.Nœuds() tel que Type(m̂″)=RSM et
                   [m̂′] ⊆ [m̂″] alors
36                     ESRG.AjouterArc(m̂ --(t,ĉ)--> m̂″)
37                 sinon si (E :=TestSaturationSym(m̂′))≠ ∅ alors
38                     m̂″ :=Regroupement(E ∪ {m̂′})
39                     ESRG.AjouterNœud( m̂″)
40                     ATraiter.InsérerEnTête(m̂″)
41                     pour tous les m̂1 --(t′,ĉ′)--> m̂2 ∈ ESRG.Arcs() tel que m̂2 ∈ E
                       faire
42                         ESRG.RetirerArc(m̂1 --(t′,ĉ′)--> m̂2)
43                         ESRG.AjouterArc(m̂1 --(t,ĉ)--> m̂″)
44                     pour tous les m̂2 ∈ E faire
45                         ESRG.RetirerNœud(m̂2)
46                 sinon
47                     ESRG.AjouterNœud( m̂′)
48                     ESRG.AjouterArc(m̂ --(t,ĉ)--> m̂′)
49                     ATraiter.InsérerEnQueue(m̂′)
50 fin
```

Algorithme 2 : Algorithme de construction du ESRG

L'algorithme 2 présente le listing du processus de construction de l'ESRG. Il utilise deux structures globales : *ESRG* et *ATraiter*. La structure *ESRG* représente le graphe à construire. La structure *ATraiter* est une liste dont chaque élément est un RSM ou un SM.

6.5.5 Exemple et Bilan de l'approche ESRG

Dans le cas d'un WN *faiblement asymétrique*[7], la réduction apportée par l'ESRG par rapport au SRG est très importante, car on a rarement besoin de représenter des nœuds de type SM.

Les Fig. 6.8 et 6.9 montrent l'ESRG [8] associé au WN de la Fig. 6.7, pour $\mathcal{C}_1 = \{pr_1, pr_2, pr_3\}$. Ce graphe contient 71 nœuds alors que le SRG de ce réseau contient 139 nœuds. Les nœuds représentés en gras sont des RSMs et ceux représentés en gris sont des SMs. Nous rappelons que la partition relâchée de la classe $\mathcal{C}_1$, associée aux RSMs, est $\mathfrak{Part}^s = \{\mathfrak{Part}^s{}_1\}$ telle que $\mathfrak{Part}^s{}_1 = \{\mathcal{D}^s_{1,1}\} = \{\{pr_1, pr_2, pr_3\}\}$, alors que celle associée aux SMs (la partition statique) est $\mathfrak{Part} = \{\mathfrak{Part}_1\}$ telle que $\mathfrak{Part}_1 = \{\mathcal{D}_{1,1}, \mathcal{D}_{1,2}, \mathcal{D}_{1,3}\}$, avec $\mathcal{D}_{1,1} = \{pr_1\}, \mathcal{D}_{1,2} = \{pr_2\}, \mathcal{D}_{1,3} = \{pr_3\}$. Par conséquent, tout SM représente un unique marquage ordinaire.

La Fig. 6.8 représente le comportement symétrique du WN : les arcs de ce graphe sont obtenus par des franchissements génériques (les transitions autres que t_6).

Les RSMs $\widehat{m}_{27}$, $\widehat{m}_{30}$, $\widehat{m}_{33}$ et $\widehat{m}_{35}$ représentent des marquages symboliques relâchés qui permettent des franchissements asymétriques (franchissements de la transition t_6). Par conséquent, les SMs de ces RSMs doivent être développés pour pouvoir construire le comportent asymétrique du WN (Fig. 6.9). Dans le tableau 6.1, nous avons regroupé tous les RSMs qui provoquent des franchissements asymétriques.

[7] L'effet des transitions asymétriques reste localisé par rapport à l'ensemble du comportement du système

[8] Les instances de franchissement ont été délibérément omises pour ne pas alourdir les figures.

RSM	Ensemble des SMs représentés
$\widehat{m}_{27} = Idl(Z_1^1) + Re(Z_1^2) + Sl(Z_1^3)$ $\lvert Z_{1,1}^1\rvert = 1, \lvert Z_{1,1}^2\rvert = 1, \lvert Z_{1,1}^3\rvert = 1$	$\widehat{m}_{27}^1 = Idl(Z_1^1) + Re(Z_1^3) + Sl(Z_1^2)$ $\widehat{m}_{27}^2 = Idl(Z_1^1) + Re(Z_1^2) + Sl(Z_1^3)$ $\widehat{m}_{27}^3 = Idl(Z_1^2) + Re(Z_1^3) + Sl(Z_1^1)$ $\widehat{m}_{27}^4 = Idl(Z_1^2) + Re(Z_1^1) + Sl(Z_1^3)$ $\widehat{m}_{27}^5 = Idl(Z_1^3) + Re(Z_1^2) + Sl(Z_1^1)$ $\widehat{m}_{27}^6 = Idl(Z_1^3) + Re(Z_1^1) + Sl(Z_1^2)$ $\lvert Z_{1,1}^1\rvert = 1, \lvert Z_{1,2}^2\rvert = 1, \lvert Z_{1,3}^3\rvert = 1$
$\widehat{m}_{30} = Rq(Z_1^1) + Re(Z_1^2) + Sl(Z_1^3)$ $\lvert Z_{1,1}^1\rvert = 1, \lvert Z_{1,1}^2\rvert = 1, \lvert Z_{1,1}^3\rvert = 1$	$\widehat{m}_{30}^1 = Rq(Z_1^1) + Re(Z_1^2) + Sl(Z_1^3)$ $\widehat{m}_{30}^2 = Rq(Z_1^2) + Re(Z_1^1) + Sl(Z_1^3)$ $\widehat{m}_{30}^3 = Rq(Z_1^3) + Re(Z_1^1) + Sl(Z_1^2)$ $\widehat{m}_{30}^4 = Rq(Z_1^1) + Re(Z_1^3) + Sl(Z_1^2)$ $\widehat{m}_{30}^5 = Rq(Z_1^2) + Re(Z_1^3) + Sl(Z_1^1)$ $\widehat{m}_{30}^6 = Rq(Z_1^3) + Re(Z_1^2) + Sl(Z_1^1)$ $\lvert Z_{1,1}^1\rvert = 1, \lvert Z_{1,2}^2\rvert = 1, \lvert Z_{1,3}^3\rvert = 1$
$\widehat{m}_{33} = Tr(Z_1^1) + Re(Z_1^2) + Sl(Z_1^3)$ $\lvert Z_{1,1}^1\rvert = 1, \lvert Z_{1,1}^2\rvert = 1, \lvert Z_{1,1}^3\rvert = 1$	$\widehat{m}_{33}^1 = Tr(Z_1^1) + Re(Z_1^3) + Sl(Z_1^2)$ $\widehat{m}_{33}^2 = Tr(Z_1^1) + Re(Z_1^2) + Sl(Z_1^3)$ $\widehat{m}_{33}^3 = Tr(Z_1^2) + Re(Z_1^3) + Sl(Z_1^1)$ $\widehat{m}_{33}^4 = Tr(Z_1^2) + Re(Z_1^1) + Sl(Z_1^3)$ $\widehat{m}_{33}^5 = Tr(Z_1^3) + Re(Z_1^2) + Sl(Z_1^1)$ $\widehat{m}_{33}^6 = Tr(Z_1^3) + Re(Z_1^1) + Sl(Z_1^2)$ $\lvert Z_{1,1}^1\rvert = 1, \lvert Z_{1,2}^2\rvert = 1, \lvert Z_{1,3}^3\rvert = 1$
$\widehat{m}_{35} = Re(Z_1^1) + Sl(Z_1^2)$ $\lvert Z_{1,1}^1\rvert = 2, \lvert Z_{1,1}^2\rvert = 1$	$\widehat{m}_{35}^1 = Re(Z_1^2 + Z_1^3) + Sl(Z_1^1)$ $\widehat{m}_{35}^2 = Re(Z_1^1 + Z_1^2) + Sl(Z_1^3)$ $\widehat{m}_{35}^3 = Re(Z_1^1 + Z_1^3) + Sl(Z_1^2)$ $\lvert Z_{1,1}^1\rvert = 1, \lvert Z_{1,2}^2\rvert = 1, \lvert Z_{1,3}^3\rvert = 1$

TAB. 6.1 – Les RSMs asymétriques et leurs développement en SMs.

$\widehat{m}^1_{31} = Idl(Z^1_1) + Re(Z^2_1) + Cs(Z^3_1)$ $\widehat{m}^2_{31} = Idl(Z^2_1) + Re(Z^1_1) + Cs(Z^3_1)$ $\widehat{m}^3_{31} = Idl(Z^3_1) + Re(Z^1_1) + Cs(Z^2_1)$ $\|Z^1_{1,1}\| = 1, \|Z^2_{1,2}\| = 1, \|Z^3_{1,3}\| = 1$	$\widehat{m}^1_{34} = Rq(Z^1_1) + Re(Z^2_1) + Cs(Z^3_1)$ $\widehat{m}^2_{34} = Rq(Z^2_1) + Re(Z^1_1) + Cs(Z^3_1)$ $\widehat{m}^3_{34} = Rq(Z^3_1) + Re(Z^1_1) + Cs(Z^2_1)$ $\|Z^1_{1,1}\| = 1, \|Z^2_{1,2}\| = 1, \|Z^3_{1,3}\| = 1$
$\widehat{m}^1_{36} = Re(Z^1_1 + Z^2_1) + Cs(Z^3_1)$ $\|Z^1_{1,1}\| = 1, \|Z^2_{1,2}\| = 1, \|Z^3_{1,3}\| = 1$	$\widehat{m}^1_{37} = Idl(Z^1_1) + Sl(Z^2_1) + Cs(Z^3_1)$ $\widehat{m}^2_{37} = Idl(Z^2_1) + Sl(Z^1_1) + Cs(Z^3_1)$ $\widehat{m}^3_{37} = Idl(Z^3_1) + Sl(Z^1_1) + Cs(Z^2_1)$ $\|Z^1_{1,1}\| = 1, \|Z^2_{1,2}\| = 1, \|Z^3_{1,3}\| = 1$
$\widehat{m}^1_{38} = Rq(Z^1_1) + Sl(Z^2_1) + Cs(Z^3_1)$ $\widehat{m}^2_{38} = Rq(Z^2_1) + Sl(Z^1_1) + Cs(Z^3_1)$ $\widehat{m}^3_{38} = Rq(Z^3_1) + Sl(Z^1_1) + Cs(Z^2_1)$ $\|Z^1_{1,1}\| = 1, \|Z^2_{1,2}\| = 1, \|Z^3_{1,3}\| = 1$	$\widehat{m}^1_{39} = Re(Z^1_1) + Sl(Z^2_1) + Cs(Z^3_1)$ $\widehat{m}^2_{39} = Re(Z^2_1) + Sl(Z^1_1) + Cs(Z^3_1)$ $\|Z^1_{1,1}\| = 1, \|Z^2_{1,2}\| = 1, \|Z^3_{1,3}\| = 1$

TAB. 6.2 – SMs résultant des franchissements asymétriques.

Le tableau 6.2 fournit les SMs résultant des franchissements asymétriques. Notons que le test de saturation (la fonction *TestSaturation*) revoie vide sur les SM de chaque case de ce tableau. En effet, aucun de ces ensembles de marquages symboliques ne correspond à une classe d'équivalence par rapport à $\mathcal{G}_s$. Par conséquent, il n'y a pas de regroupement possibles.

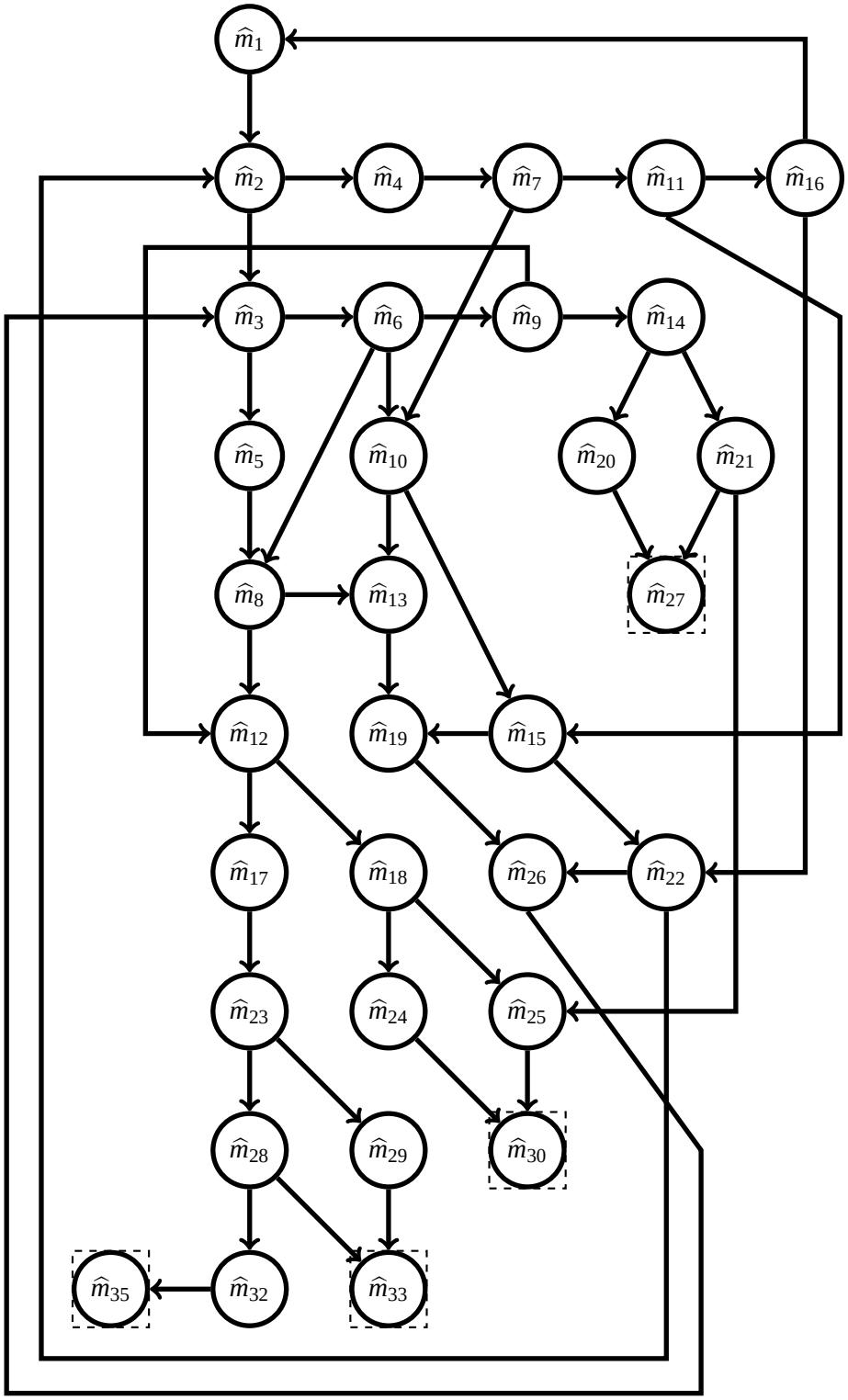


FIG. 6.8 – Partie symétrique de l'ESRG du WN de la Fig 6.7

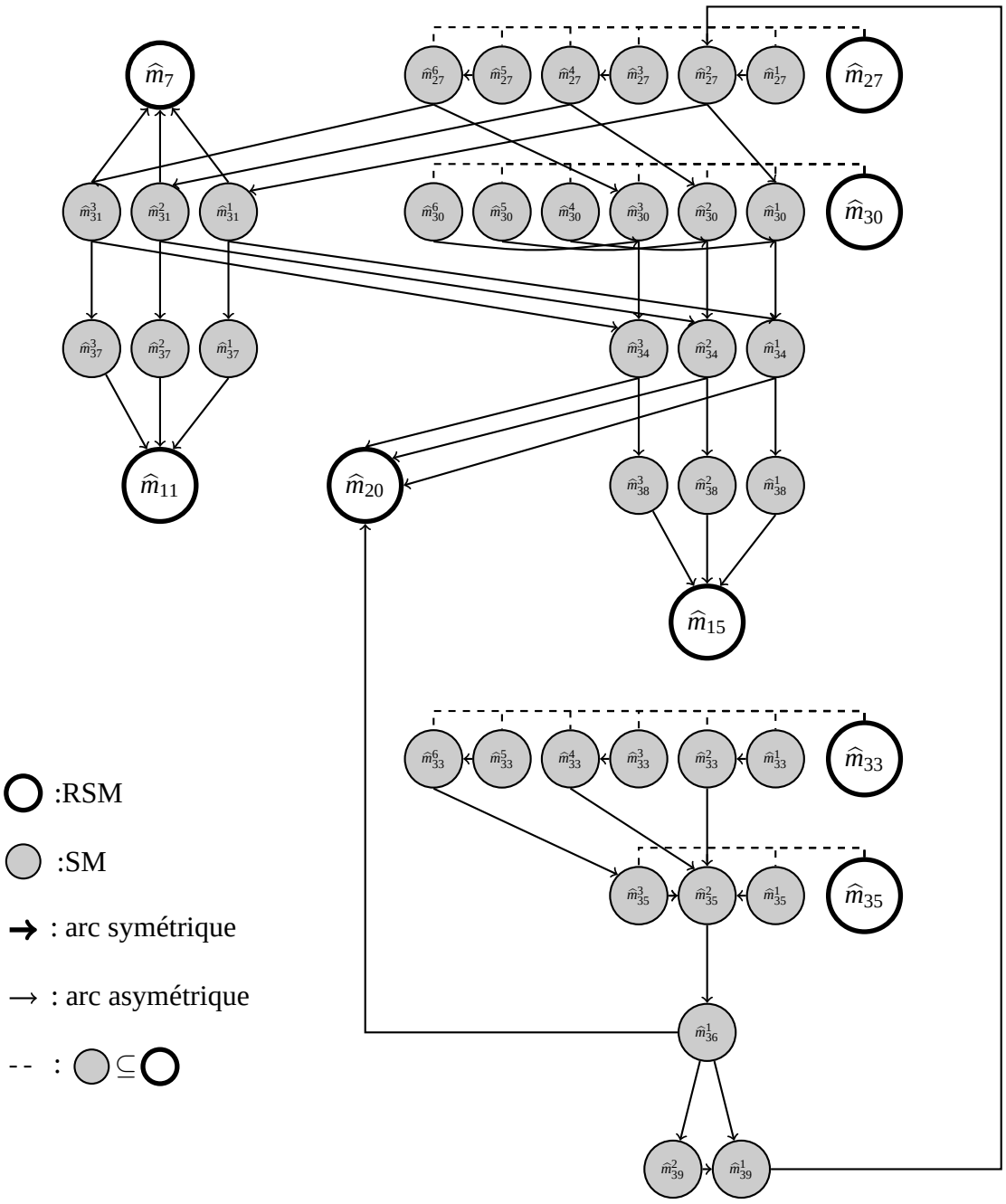


FIG. 6.9 – Partie asymétrique de l'ESRG du WN de la Fig 6.7

Dans le cas des WN *fortement asymétriques*, le gain de l'ESRG par rapport au SRG est pratiquement nul. En effet, l'existence d'un nombre important de transitions asymétriques oblige à développer un nombre important de SMs et par conséquent on perd la réduction apportée par les RSMs, et ceci explique le gain négli-

geable.

Aussi, la définition d'un unique sous-groupe $\mathcal{G}$ (représenté par la partition statique de $\mathcal{C} : \mathfrak{Part} = \{\mathfrak{Part}_1\} = \{\{\{p_1\},\{p_2\},\{p_3\}\}\}$) pour toutes les transitions asymétriques du système ne permet pas d'exploiter les symétries qui apparaissent localement sur chaque transition, durant l'exécution du système. Par exemple, si l'on considérait la transition t_6 et les deux marquages symboliques $\widehat{m}^3_{27}$ et $\widehat{m}^5_{27}$ alors on peut remarquer qu'ils sont symétriques par rapport au sous-groupe $\mathcal{H} = \{id, g\}$ tel que $g.p_3 = p_2$, $g.p_2 = p_3$ et $g.p_1 = p_1$ (implicitement représenté par la partition : $\mathfrak{Part}'_1 = \{\{p_1\},\{p_2,p_3\}\}$). En reprenant ce principe de localité des symétries sur les successeurs de ces marquages symboliques nous pouvons ajouter une réduction supplémentaire à une partie du graphe de la Fig. 6.9 (voir Fig. 6.10).

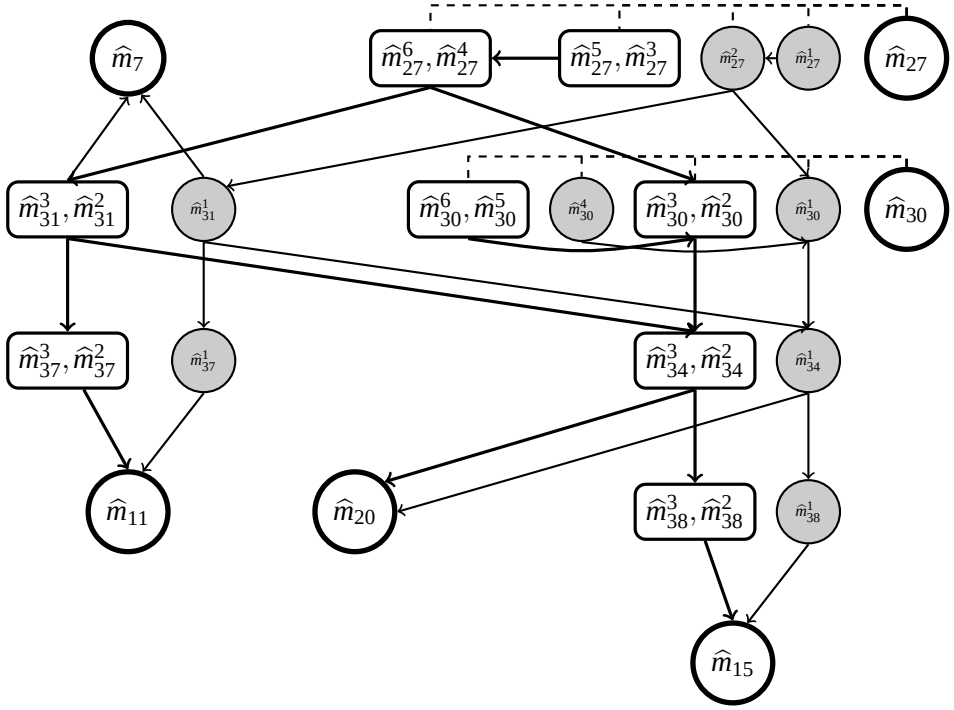


FIG. 6.10 – Exemple d'exploitation des symétries locales.

6.6 Conclusion

Nous avons étudié les WN comme formalisme de modélisation des systèmes concurrents (finis, dans notre cas) dont la sémantique est une $\mathcal{SKE}$, appelée graphe d'accessibilité (*RG*). Grâce à leur structuration, les WN permettent la construction automatique de deux graphes quotient : le SRG et l'ESRG.

Dans l'approche SRG, les symétries sont définies d'une manière statique et globale (au niveau du modèle). Ce sont les seules prises en compte pour la construction du graphe quotient. L'approche SRG ne bénéficie donc pas de la localité des conditions qui spécifient les asymétries, ce qui engendre une réduction faible, sinon inexistante, par rapport au RG dans le cas des modèles asymétriques. Sur l'exemple de la Fig. 6.7, le SRG est identique au RG, car l'asymétrie induite par le processus de sélection est prise en compte au niveau global, bien que son impact ne soit que local.

L'approche ESRG, quant à elle, apporte une première solution à ce problème par une définition statique des asymétries et une utilisation au besoin. Cependant, cette définition statique des asymétries est commune à toutes les transitions asymétriques du WN : considérons deux transitions asymétriques t et t', telles que pour le franchissement de t on distingue uniquement les objets pr_1 et pr_2, et pour t' on distingue les objets pr_3 et pr_4 (uniquement). L'approche ESRG considère une partition statique dans laquelle chacun de ces quatre objets est distingué. Cette partition va être prise en compte à chaque fois que l'une des deux transitions est franchissable. Ceci engendre une *asymétrie inutile* entre les objets pr_1 et pr_2 lors du franchissement de t' mais aussi une *asymétrie inutile* entre pr_3 et pr_4 lors du franchissement de t.

La solution originale que nous proposons est basée sur *une gestion totalement dynamique des (a)symétries du système*. Cette solution consiste à mettre en œuvre les techniques génériques présentées dans les deux chapitres précédents, tout en exploitant les apports du formalisme des WN. L'intérêt de cette approche est de permettre une vérification efficace des systèmes globalement asymétriques.

Chapitre 7

Les Réseaux de Petri Bien Formés : approche par symétries dynamiques[1]

Lors du développement de notre méthode générique pour la vérification des systèmes globalement asymétriques (chapitres 3 et 4), nous avons introduit la notion de $\mathcal{SKE}$ *contrôlée* (section 4.3). Dans cette approche, nous avons transféré toutes les asymétries du système sur l'automate de contrôle, ce qui nous a permis de les gérer d'une manière totalement dynamique. Ceci a abouti à la définition d'une nouvelle structure quotient sur laquelle la vérification d'un système globalement asymétrique peut être optimisée.

Dans ce chapitre, nous allons voir comment appliquer cette approche dans le cadre des WN.

7.1 Le modèle des WN Contrôlés

Les systèmes asymétriques que nous considérons sont modélisés par une paire $\langle N, \mathcal{A}_c \rangle$: $N = \langle P, T, C, Cd, Pre, Post, Inh, \phi, \pi \rangle$ représente un WN totalement symétrique[2] dont le fonctionnement est contrôlé par un automate de contrôle, $\mathcal{A}_c = \langle L, l_0, \mathcal{E}, \mathcal{R} \rangle$. Le contrôle porte sur le franchissement des transitions de N, par conséquent, $\mathcal{E} = \{(t,c) | t \in T \ \wedge \ c \in Cd(t)\}$.

[1] Les idées développées dans ce chapitre ont fait l'objet de plusieurs publications [IBB+04, TMBDLK04, BIDL04, HTMK+04, TMIB06] .

[2] Nous rappelons qu'un WN est symétrique ssi la partition statique de chaque classe du réseau n'est constituée que d'une unique sous-classe statique regroupant l'ensemble des objets de la classe.

Exemple 7.1. *Reprenons l'exemple de la section critique. Nous voulons maintenant représenter la phase de sélection sans préciser dans le modèle sur quel critère cette sélection est effectuée. Dans ce WN, tous les processus ont un comportement identique : si* $\mathcal{C}_1 = \{pr_1, \ldots, pr_n\}$*, nous avons alors* $\mathfrak{Part}_1 = \{pr_1, \ldots, pr_n\}$*. Ce WN est donc totalement symétrique.*

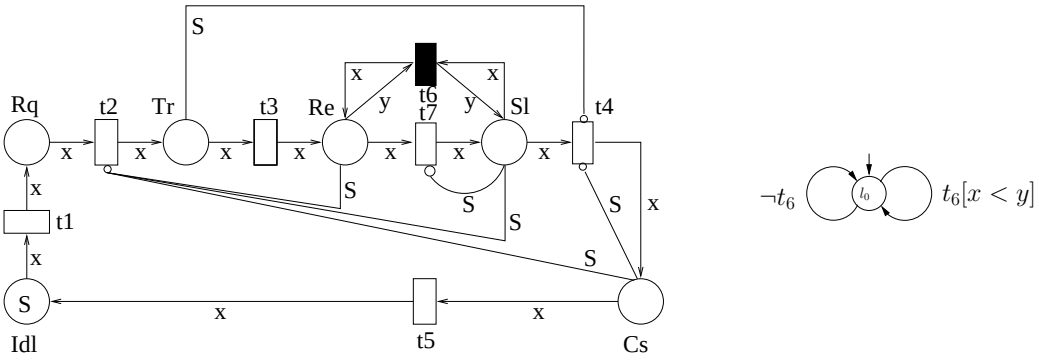


FIG. 7.1 – CWN de l'algorithme d'accès à la section critique [à gauche un WN symétrique et à droite son automate de contrôle].

Le critère de sélection est introduit par l'automate de contrôle $\mathcal{A}_c = \langle L, l_0, \mathcal{E}, \mathcal{R} \rangle$ *tel que :*

- $L = \{l_0\}$,
- $\mathcal{E} = \{(t,c) | t \in T \ \wedge \ c \in Cd(t)\}$,
- $\mathcal{R} = \{\langle l_0, t_6[x<y], l_0\rangle, \langle l_0, \neg t_6, l_0\rangle\}$ *où*
  - $t_6[x<y] = \{(t_6, (p_i, p_j)) \mid (p_i, p_j) \in Cd(t_6) \wedge j > i\}$;
  - $\neg t_6 = \{(t,c) \mid t \neq t_6 \wedge c \in Cd(t)\}$;

Ainsi, les asymétries représentées initialement sur le WN (Fig.6.7) sont totalement transférées sur l'automate de contrôle.

Ce nouveau modèle est appelé *le modèle des WN contrôlés (CWN, pour Controlled WN)* et la paire $\langle N, \mathcal{A}_c \rangle$ est notée $N_{\mathcal{A}_c}$. Un état de $N_{\mathcal{A}_c}$ est un couple (m,l) tel que m est un marquage de N et l est un état de $\mathcal{A}_c$. Un état initial de $N_{\mathcal{A}_c}$ est formé d'un marquage initial m_0 et d'un état initial l_0.

7.2 Sémantique associée aux CWN

La sémantique associée à un CWN est définie par une adaptation de la règle de franchissement classique des WN, qui prend en compte la synchronisation avec

l'automate de contrôle. Elle permet de construire à partir d'un état initial (m_0, l_0) un graphe dont chaque nœud est un état du système. Les arcs de ce graphe représentent les événements dont l'occurrence, jugée acceptable par l'automate de contrôle, permet de passer d'un état à l'autre.

Définition 7.1 (Règle de franchissement par synchronisation)**.** *Soit $N_{\mathcal{A}_c}$ un CWN, t une transition de N et $\langle l, \gamma, l' \rangle$ une transition de $\mathcal{A}_c$. L'instance de transition (t,c) est franchissable à partir de l'état (m,l), notée $(m,l)[(t,c)\rangle\rangle$, ssi :*

$$m[(t,c)\rangle \wedge (t,c) \in \gamma.$$

L'état obtenu par ce franchissement est (m',l') tel que, $m' = m[(t,c)\rangle$.

Exemple 7.2. *Considérons l'exemple précédent avec l'état (m,l_0) tel que $m = Re(pr_1 + pr_3) + Sl(pr_2)$. Dans ce cas, seule la transition t_6 est franchissable vis-à-vis du WN symétrique et l'ensemble des instances générées à partir de m est $\{(t_6,(pr_2,pr_1)),(t_6,(pr_2,pr_3))\}$. Cependant, la synchronisation de cet ensemble avec l'arc $\langle l_0, t_6[x < y], l_0\rangle$) de $\mathcal{A}_c$, donne $(t_6,(pr_2,pr_3))$ comme seule instance réellement franchissable à partir du marquage m.*

Si nous considérons que G_N est le RG de N alors, du point de vue de la méthode générique de la section 4.3, la règle de franchissement par synchronisation produit le graphe $G_N \overline{\otimes} \mathcal{A}_c$.

7.3 Produit synchronisé symbolique

En se basant sur cette nouvelle approche de modélisation, nous voulons maintenant construire la version symbolique du produit synchronisé quotient (défini dans la section 4.3) : ayant un système modélisé par un CWN $N_{\mathcal{A}_c}$, et une propriété f, exprimée par un $\mathcal{TGBA}$ $\mathcal{A}_f$, à vérifier sur ce système, notre objectif est de construire symboliquement un représentant de la structure $(G_N \overline{\otimes} \mathcal{A}_c) \otimes \mathcal{A}_f$. Cette structure symbolique est appelée *Produit synchronisé symbolique (SSP, pour Symbolic Synchronised Product).*

La construction de cette structure symbolique nécessite la représentation symbolique d'un nœud $\langle \mathcal{H}, O, l, q \rangle$ et la définition d'une règle de franchissement symbolique adaptée à ce nœud.

7.3.1 Représentation symbolique dans les CWN

Trouver une représentation symbolique pour un nœud $\langle \mathcal{H}, O, l, q \rangle$ se réduit à la recherche d'une représentation symbolique pour le couple $(\mathcal{H}, O)$ (dans ce cadre, les éléments de O sont des marquages).

7.3.1.1 Définition d'un état symbolique

Nous savons qu'un marquage symbolique (classique) d'un WN représente une classe d'équivalence de marquages, par rapport à la relation d'équivalence induite par le sous-groupe des symétries admissibles $\mathcal{G}$ (voir section 6.3.1). Aussi, suivant la construction proposée dans la section 4.3, dans chaque nœud $\langle \mathcal{H}, O, l, q \rangle$ du graphe quotient, l'ensemble O est une classe d'équivalence d'états par rapport à la relation d'équivalence induite par le sous-groupe $\mathcal{H}$. Ainsi, il est possible de représenter le couple $(\mathcal{H}, O)$ par une représentation symbolique similaire à celle utilisée dans l'approche ESRG.

En effet, chaque couple $(\mathcal{H}, O)$ peut être représenté par $\langle \mathfrak{Part}^l, \hat{m}_{\mathfrak{Symb}} \rangle$ tel que :
- $\mathfrak{Part}^l$ est une partition de $\mathcal{C}$ (la famille de classe du réseau), représentant implicite du sous-groupe de permutations $\mathcal{H}$. Par analogie à la partition statique, $\mathfrak{Part}^l = \{\mathfrak{Part}^l_i\}_{i=1,\ldots,n}$ est appelée *partition locale*. Nous utiliserons la notation $\mathcal{L}_{i,j}$ pour la partie locale j de la classe i : $\mathfrak{Part}^l_i = \{\mathcal{L}_{i,j}\}_j$.
- $\mathfrak{Symb}$ est une partition symbolique (locale) de $\mathcal{C}$, formalisée de manière analogue à celle de la définition 6.13. La seule différence est que la partition statique est remplacée par la partition locale $\mathfrak{Part}^l$.
- $\hat{m}$ est un représentant symbolique de l'ensemble des éléments de O. Sa formalisation est analogue à celle de la définition 6.15, en remplaçant la partition statique de $\mathcal{C}$ par la partition locale $\mathfrak{Part}^l$ et en prenant $\mathfrak{Symb}$ comme partition symbolique (locale).

Exemple 7.3. *Reprenons l'exemple de la Fig.7.1 avec* $\mathcal{C} = \{\mathcal{C}_1\}$ *et* $\mathcal{C}_1 = \{pr_1, pr_2, pr_3\}$.

Soit le couple $(\mathcal{H}, O)$ *tel que :*
- $O = \{m_1, m_2\}$ *est l'ensemble de marquages représentant les états du système où l'un des deux processus* pr_1 *ou* pr_2 *est au repos, alors que l'autre, parallèlement à* pr_3*, a envoyé une requête d'accès à la section critique :* $m_1 = Idl(pr_1) + Rq(pr_2 + pr_3)$ *et* $m_2 = Idl(pr_2) + Rq(pr_1 + pr_3)$.
- $\mathcal{H}$ *est un sous-groupe de permutations tel que :* $\mathcal{H} = \{id, g\}$ *avec* $g.m_1 = m_2$, $g.m_2 = m_1$. *En réalité,* $\mathcal{H}$ *est défini sur* $\mathcal{C}_1$ ($g.pr_1 = pr_2$, $g.pr_2 = pr_1$ *et*

$g.pr_3 = pr_3$), mais il s'étend naturellement sur les marquages.

L'état $(\mathcal{H}, O)$ peut être représenté symboliquement par le couple $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$ tel par :
- $\mathfrak{Part}^{\mathfrak{l}} = \{\mathfrak{Part}^{\mathfrak{l}}_1\} = \{\{\mathcal{L}_{1,1}, \mathcal{L}_{1,2}\}\}$, *avec* $\mathcal{L}_{1,1} = \{p_1, p_2\}$ *et* $\mathcal{L}_{1,2} = \{p_3\}$,
- $\mathfrak{Symb} = \{\langle \mathfrak{Dyn}_1, card_1, d_1 \rangle\}$ *où :*
  - $\mathfrak{Dyn}_1 = \{Z_1^1, Z_1^2, Z_1^3\}$,
  - $card_1(1) = 1, card(2) = 1$ *et* $card(3) = 1$,
  - $d_1(1) = 1, d_1(2) = 1$ *et* $d_1(3) = 2$.
- $\hat{m} = Idl(Z_1^1) + Rq(Z_1^2 + Z_1^3)$.

Ainsi, le quadruplet $\langle \mathcal{H}, O, l, q \rangle$ sera représenté par $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l, q \rangle$ et appelé *état symbolique*.

Notations.
- Les notations définis pour les marquages symboliques (classiques) des WN (section 6.3.2) restent valables dans le contexte actuel. Nous rappelons ici les plus importantes :
  - La cardinalité, $card_i(k)$, d'une sous classe dynamique Z_i^k est notée $\mid Z_i^k \mid$. La Z_i^k telle que $d_i(k) = j$ sera notée $Z_{i,j}^k$. Par conséquent, une partition symbolique $\mathfrak{Symb}_i = \langle \mathfrak{Dyn}_i, card_i, d_i \rangle$ peut être totalement définie en utilisant ces abréviations. Par exemple, $\mathfrak{Symb}_1 = \langle \mathfrak{Dyn}_1, card_1, d_1 \rangle$ telle que $\mathfrak{Dyn}_1 = \{Z_1^1, Z_1^2\}$, $card_1(1) = 1$ et $card_1(2) = 2$, $d_1(1) = 1$, $d_1(2) = 2$, est totalement décrite par : $|Z_{1,1}^1| = 1$ et $|Z_{1,2}^2| = 2$.
  - Nous noterons $\mathfrak{Symb}_i.\mathfrak{Dyn}_i$, $\mathfrak{Symb}_i.card_i$ et $\mathfrak{Symb}_i.d_i$ les différents éléments de la partition symbolique $\mathfrak{Symb}_i = \langle \mathfrak{Dyn}_i, card_i, d_i \rangle$. Cette notation est trivialement étendue sur $\mathfrak{Symb} = \{\mathfrak{Symb}_i\}_{i=1,\ldots,n}$.
- Quand il n'y a pas de confusion sur la partition symbolique utilisée, nous noterons $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l, q \rangle$ par $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}, l, q \rangle$.
- L'ensemble des marquages ordinaires représentés par $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$ est noté $[\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle]$.
- Nous notons aussi par $[\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l, q \rangle]$, l'ensemble des états ordinaires représentés par $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l, q \rangle$.

7.3.2 Opérations sur les états symboliques

Dans la méthode générique, nous avons utilisé des opérations spécifiques sur les nœuds du graphe quotient : l'inclusion ($\subseteq$) et la décomposition (Decomp). Nous avons énoncé que l'implémentation de ces opérations dépend du codage des nœuds. Ici, nous définissons les versions symboliques de ces opérations, adaptées

aux états symboliques définis précédemment.

Par ailleurs, nous définissons deux autres opérations : *le raffinement symbolique* et *le regroupement symbolique*. La première est nécessaire pour la réalisation des opérations d'inclusion et de décomposition. La deuxième est primordiale pour l'optimisation de la taille de la structure à construire.

Les opérations que nous allons réaliser se répartissent en deux catégories : (1) les opérations impliquant plusieurs états symboliques, qui ne sont autorisées que si elles opèrent sur des états qui ont le même état de l'automate de contrôle, et le même état du $\mathcal{TGBA}$. C'est le cas pour l'inclusion, la décomposition et le regroupement ; (2) les opérations n'impliquant qu'un seul état, qui sont de ce fait sans relation avec les états du $\mathcal{CA}$ et du $\mathcal{TGBA}$. C'est le cas du raffinement.

Par conséquent, pour la suite de cette section, les états seront décrits en utilisant uniquement leur partie symbolique, *i.e.*, le marquage symbolique et la partition locale : $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l, q \rangle$ sera désigné par $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$ ou plus simplement par $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m} \rangle$.

7.3.2.1 Raffinement symbolique

Le raffinement d'un état symbolique est l'opération de base sur laquelle repose la définition de toutes les autres opérations. Elle consiste en la substitution d'un état symbolique $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$ par un ensemble d'états symboliques, $E = \{\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}^i_{\mathfrak{Symb}^i} \rangle\}_i$, sachant que $\mathfrak{Part}^{\mathfrak{l}'}$ *est une partition plus fine que* [3] $\mathfrak{Part}^{\mathfrak{l}}$: si $\mathcal{H}$ est le sous-groupe de permutations associé à $\mathfrak{Part}^{\mathfrak{l}}$ alors $\mathfrak{Part}^{\mathfrak{l}'}$ est le représentant implicite d'un sous-groupe $\mathcal{H}' \subseteq \mathcal{H}$.

[3] Par définition, une partition est plus fine qu'une autre si elle fractionne les parties de cet autre en de plus petites parties.

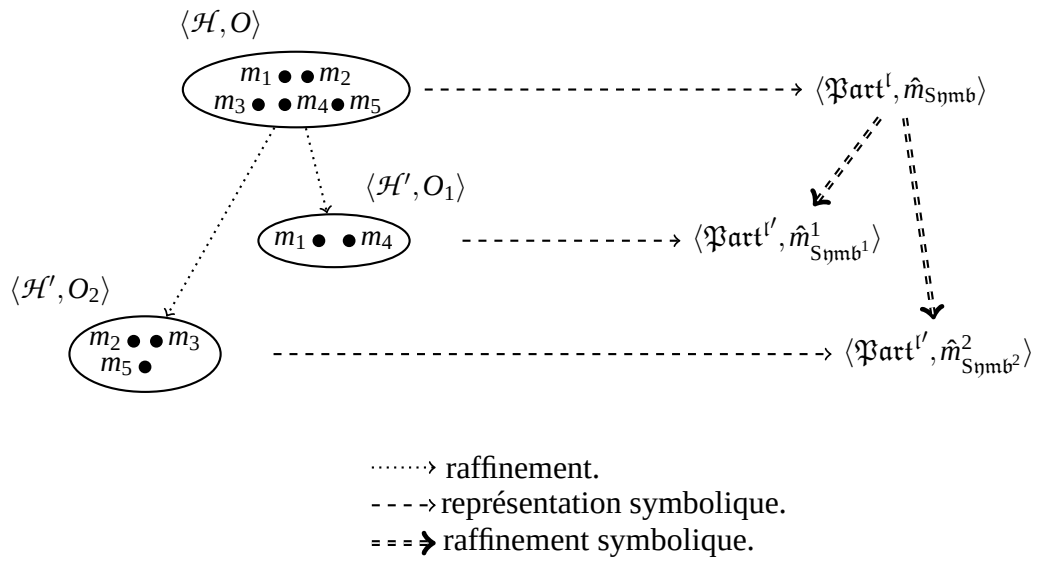


FIG. 7.2 – Raffinement *vs.* raffinement symbolique.

Nous allons montrer qu'il est possible de construire E directement à partir de $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}} \rangle$, moyennant des transformations sur la partition symbolique $\mathfrak{Symb}$. Ces transformations permettent la déduction de l'ensemble $\{\hat{m}^i\}_i$ à partir de $\hat{m}$.

L'exemple suivant introduit l'idée de la transformation à réaliser.

Exemple 7.4. *Soit $\mathcal{C}_1 = \{pr_1, pr_2, pr_3\}$ une classe d'objet, $\mathfrak{Part}^{\iota}_1 = \{\mathcal{L}_{1,1}\}$, une partition de cette classe, telle que $\mathcal{L}_{1,1} = \{pr_1, pr_2, pr_3\}$. Soit $\mathfrak{Symb}_1 = \langle \mathfrak{Dyn}_1, card_1, d_1 \rangle$ une partition symbolique de $\mathcal{C}_1$ telle que $\mathfrak{Dyn}_1 = \{Z^1_1, Z^2_1\}$ avec $|Z^1_{1,1}| = 1$ et $|Z^2_{1,1}| = 2$. Soit $\mathfrak{Part}^{\iota'}_1 = \{\mathcal{L}'_{1,1}, \mathcal{L}'_{1,2}\}$ un raffinement de $\mathfrak{Part}^{\iota}_1$ telle que $\mathcal{L}'_{1,1} = \{pr_1\}$, $\mathcal{L}'_{1,2} = \{pr_2, pr_3\}$.*

L'idée est de trouver l'ensemble des partitions symboliques représentants $\mathfrak{Symb}_1$, mais en respectant la nouvelle partition locale $\mathfrak{Part}^{\iota'}_1$. Intuitivement, il faut distribuer le nombre d'éléments représentés par chaque sous-classe dynamique de $\mathfrak{Dyn}_1$ sur des sous-classes dynamiques qui respectent les partitions de $\mathfrak{Part}^{\iota'}_1$. Puisque Z^2_1 représente deux éléments, nous pouvons les choisir : tous les deux dans $\mathcal{L}'_{1,2}$ ou un dans $\mathcal{L}'_{1,2}$ et l'autre dans $\mathcal{L}'_{1,1}$. Nous obtenons ainsi deux partitions symboliques.

Nous formalisons cette transformation par la notion de *raffinement d'une partition symbolique*.

Définition 7.2 (Raffinement d'une partition symbolique)**.** *Soient* $\mathfrak{Part}^{\iota}$ *et* $\mathfrak{Part}^{\iota'}$ *deux partitions de* $\mathcal{C}$ *telles que* $\mathfrak{Part}^{\iota'}$ *est un raffinement de* $\mathfrak{Part}^{\iota}$*. Soit* $\mathfrak{Symb} = \{\mathfrak{Symb}_i\}_{i=1,\ldots,n}$ *une partition symbolique de* $\mathcal{C}$ *suivant* $\mathfrak{Part}^{\iota}$*. Le raffinement de* $\mathfrak{Symb}$ *suivant* $\mathfrak{Part}^{\iota'}$ *est l'ensemble de toutes les partitions symboliques, noté* $\mathrm{Raf}(\mathfrak{Part}^{\iota}, \mathfrak{Part}^{\iota'}, \mathfrak{Symb})$*, qui vérifient :*

$\forall \mathfrak{Symb}' \in \mathrm{Raf}(\mathfrak{Part}^{\iota}, \mathfrak{Part}^{\iota'}, \mathfrak{Symb}), \forall \mathfrak{Symb}'_i \in \mathfrak{Symb}'$, $\exists$ *surjection* $\Upsilon_{\mathfrak{Symb}'} = \prod_{i=1}^{n} \Upsilon_{\mathfrak{Symb}'_i}$ *où* $\Upsilon_{\mathfrak{Symb}'_i} : \{1, \ldots, |\mathfrak{Symb}'_i.\mathfrak{Dyn}_i|\} \mapsto \{1, \ldots, |\mathfrak{Symb}_i.\mathfrak{Dyn}_i|\}$ *telle que :* $\forall k \in \{1, \ldots, |\mathfrak{Symb}_i.\mathfrak{Dyn}_i|\}$,

- *chaque objet d'une sous-classe dynamique de la partition initiale est représenté par une sous-classe dynamique de la partition raffinée :* $\sum_{k' \in \Upsilon^{-1}_{\mathfrak{Symb}'_i}(k)} \mathfrak{Symb}'_i.card_i(k') = \mathfrak{Symb}_i.card_i(k)$,
- *les sous classes dynamiques issues du raffinement d'une sous classe dynamique de la partition initiale ne peuvent être instanciées que par des couleurs appartenant à la partie locale de la sous classe dynamique raffinée :* $\bigcup_{k' \in \Upsilon^{-1}_{\mathfrak{Symb}'_i}(k)} \mathcal{L}'_{i,\mathfrak{Symb}'_i.d_i(k')} \subseteq \mathcal{L}_{i,\mathfrak{Symb}_i.d_i(k)}$.

  Où $\mathcal{L}'_{i,\mathfrak{Symb}'_i.d_i(k')} \in \mathfrak{Part}^{\iota'}_i$ *et* $\mathcal{L}_{i,\mathfrak{Symb}_i.d_i(k)} \in \mathfrak{Part}^{\iota}{}_i$,
- *le raffinement d'une sous-classe dynamique de la partition initiale ne doit pas contenir deux sous-classes dynamiques instanciées dans le même ensemble de la partition locale du raffinement :* $\forall k' \neq k'' \in \Upsilon^{-1}_{\mathfrak{Symb}'_i}(k), \mathfrak{Symb}'_i.d_i(k') \neq \mathfrak{Symb}'_i.d_i(k'')$.

Exemple 7.5. *Reprenons les partitions locales,* $\mathfrak{Part}^{\iota}$ *et* $\mathfrak{Part}^{\iota'}$*, et la partition symbolique* $\mathfrak{Symb} = \{\mathfrak{Symb}_1\}$ *de l'exemple précédent.*

Le raffinement de de la partition symbolique $\mathfrak{Symb}$ *par rapport à* $\mathfrak{Part}^{\iota'}$*, est donnée par* $\mathrm{Raf}(\mathfrak{Part}^{\iota}, \mathfrak{Part}^{\iota'}, \mathfrak{Symb}) = \{\mathfrak{Symb}^1, \mathfrak{Symb}^2\}$ *tel que :*

- $\mathfrak{Symb}^1 = \{\mathfrak{Symb}^1_1\}$ *où*
  - $\mathfrak{Symb}^1_1.\mathfrak{Dyn}_1 = \{Z^1_1, Z^2_1\}$;
  - $\mathfrak{Symb}^1_1.card_1(1) = 1, \mathfrak{Symb}^1_1.card_1(2) = 1$;
  - $\mathfrak{Symb}^1_1.d_1(1) = 1, \mathfrak{Symb}^1_1.d_1(2) = 2$.
- $\mathfrak{Symb}^2 = \{\mathfrak{Symb}^2_1\}$ *où*
  - $\mathfrak{Symb}^2_1.\mathfrak{Dyn}_1 = \{Z^1_1, Z^2_1, Z^3_1\}$;
  - $\mathfrak{Symb}^2_1.card_1(1) = 1, \mathfrak{Symb}^2_1.card_1(2) = 1, \mathfrak{Symb}^2_1.card_1(3) = 1$;
  - $\mathfrak{Symb}^2_1.d_1(1) = 1, \mathfrak{Symb}^2_1.d_1(2) = 2, \mathfrak{Symb}^2_1.d_1(3) = 2$.

Nous pouvons ainsi définir deux surjections $\Upsilon_{\mathfrak{Symb}_1^1}(\Upsilon_{\mathfrak{Symb}_1^1}(1)=1,\Upsilon_{\mathfrak{Symb}_1^1}(2)=2)$ *et* $\Upsilon_{\mathfrak{Symb}_1^1}(\Upsilon_{\mathfrak{Symb}_1^2}(1)=1,\Upsilon_{\mathfrak{Symb}_1^2}(2)=2,\Upsilon_{\mathfrak{Symb}_1^2}(3)=2)$ *qui vérifient la définition précédente.*

Á présent, il devient facile de dériver les états symboliques issus d'un raffinement symbolique à partir de l'état symbolique original.

Définition 7.3 (Raffinement d'un état symbolique)**.** *Soient* $\mathfrak{Part}^{\iota}$ *et* $\mathfrak{Part}^{\iota'}$ *deux partitions locales de la famille de classes* $\mathcal{C}$ *tel que* $\mathfrak{Part}^{\iota'}$ *est plus fine que* $\mathfrak{Part}^{\iota}$*. Le raffinement de l'état symbolique* $\langle\mathfrak{Part}^{\iota},\hat{m}_{\mathfrak{Symb}}\rangle$ *suivant* $\mathfrak{Part}^{\iota'}$ *associe à chaque partition symbolique* $\mathfrak{Symb}'\in \mathrm{Raf}(\mathfrak{Part}^{\iota},\mathfrak{Part}^{\iota'},\mathfrak{Symb})$ *et à chaque* $\Upsilon_{\mathfrak{Symb}'}$*, un état symbolique* $\langle\mathfrak{Part}^{\iota'},\hat{m}'_{\mathfrak{Symb}'}\rangle$*, tel que :*

$$\forall p\in P,\forall Z\in \mathfrak{Symb}'_{J(p)},\hat{m}'(p)(\Pi_{i=1}^{n}\Pi_{j=1}^{e_i}Z_i^{w(i,j)})=\hat{m}(p)(\Pi_{i=1}^{n}\Pi_{j=1}^{e_i}Z_i^{\Upsilon_{\mathfrak{Symb}'_i}(w(i,j))})$$

L'ensemble des états symboliques construits est noté $\mathrm{RafSymb}(\langle\mathfrak{Part}^{\iota},\hat{m}_{\mathfrak{Symb}}\rangle,\mathfrak{Part}^{\iota'})$.

Exemple 7.6. *Soit* $\mathcal{C}=\{\mathcal{C}_1\}$ *avec* $\mathcal{C}_1=\{pr_1,pr_2,pr_3\}$ *la famille de classes du CWN. Soit* $\mathfrak{Part}=\{\mathfrak{Part}^{\iota}{}_1\}=\{\{\mathcal{L}_{1,1}\}\}$*, une partition, telle que* $\mathcal{L}_{1,1}=\{pr_1,pr_2,pr_3\}$*.* $\mathfrak{Part}^{\iota'}=\{\mathfrak{Part}^{\iota'}{}_1\}=\{\{\mathcal{L}'_{1,1},\mathcal{L}'_{1,2},\mathcal{L}'_{1,3}\}\}$ *un raffinement de* $\mathfrak{Part}^{\iota}$ *tel que* $\mathcal{L}'_{1,1}=\{pr_1\}$*,* $\mathcal{L}'_{1,2}=\{pr_2\}$ *et* $\mathcal{L}'_{1,3}=\{pr_3\}$*.*

Soit $\langle\mathfrak{Part}^{\iota},\hat{m}_{\mathfrak{Symb}}\rangle$ *un état symbolique tel que :*
- $\mathfrak{Symb}=\{\mathfrak{Symb}_1\}$*, où* $\mathfrak{Symb}_1$ *est décrite par :* $|Z^1_{1,1}|=1,|Z^2_{1,1}|=2$*,*
- $\hat{m}=Idl(Z_1^1)+Rq(Z_1^2)$ *: un processus est au repos alors que les eux autres ont envoyé une requête d'accès à la section critique.*

Le raffinement symbolique de $\langle\mathfrak{Part}^{\iota},\hat{m}_{\mathfrak{Symb}}\rangle$ *p.r.a* $\mathfrak{Part}^{\iota'}$ *est défini par l'ensemble* $\mathrm{RafSymb}(\langle\mathfrak{Part}^{\iota},\hat{m}_{\mathfrak{Symb}}\rangle,\mathfrak{Part}^{\iota'})=\{\langle\mathfrak{Part}^{\iota'},\hat{m}^1_{\mathfrak{Symb}^1}\rangle,\langle\mathfrak{Part}^{\iota'},\hat{m}^2_{\mathfrak{Symb}^1}\rangle,\langle\mathfrak{Part}^{\iota'},\hat{m}^3_{\mathfrak{Symb}^1}\rangle\}$ *tel que* $\mathrm{Raf}(\mathfrak{Part}^{\iota},\mathfrak{Part}^{\iota'},\mathfrak{Symb})=\{\mathfrak{Symb}^1\}$ *où* $\mathfrak{Symb}^1=\{\mathfrak{Symb}^1_1\}$ *avec* $\mathfrak{Symb}^1_1$ *est décrite par* $|Z^1_{1,1}|=1$*,* $|Z^2_{1,2}|=1$*,* $|Z^3_{1,3}|=1$*. Nous avons donc,*
- $\hat{m}^1=Idl(Z^1_{1,1})+Rq(Z^2_{1,2}+Z^3_{1,3})$: $\Upsilon^1_{\mathfrak{Symb}^1_1}(1)=1,\Upsilon^1_{\mathfrak{Symb}^1_1}(2)=\Upsilon^1_{\mathfrak{Symb}^1_1}(3)=2$;
- $\hat{m}^2=Idl(Z^2_{1,2})+Rq(Z^1_{1,1}+Z^3_{1,3})$: $\Upsilon^2_{\mathfrak{Symb}^1_1}(2)=1,\Upsilon^2_{\mathfrak{Symb}^1_1}(1)=\Upsilon^2_{\mathfrak{Symb}^1_1}(3)=2$;
- $\hat{m}^3=Idl(Z^3_{1,3})+Rq(Z^1_{1,1}+Z^2_{1,2})$: $\Upsilon^3_{\mathfrak{Symb}^1_1}(3)=1,\Upsilon^3_{\mathfrak{Symb}^1_1}(1)=\Upsilon^3_{\mathfrak{Symb}^1_1}(2)=2$;

7.3.2.2 Regroupement symbolique

L'opération de regroupement vise à minimiser la taille de la structure à construire. En effet, ayant un ensemble E d'états symboliques de taille $|E|$, l'objectif est de substituer E par un autre ensemble d'états symboliques E' tel que E et E' représentent le même ensemble d'états ordinaires et $|E'| \leq |E|$. Pour réaliser cette substitution, nous commençons par caractériser un ensemble d'états symboliques regroupables.

Définition 7.4 (Ensemble d'états symboliques regroupables)**.** *Soit $\mathcal{C}$ une famille de classes, $E = \{\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}^1_{\mathfrak{Symb}^1}\rangle, \langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}^2_{\mathfrak{Symb}^2}\rangle \ldots\}$ un ensemble fini d'états symboliques. E est regroupable ssi il existe un état symbolique $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}\rangle$, tel que :*

$$\text{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}\rangle, \mathfrak{Part}^{\mathfrak{l}'}) = E.$$

Autrement dit, un ensemble d'états symboliques ne peut être regroupable que si l'ensemble des états ordinaires qu'il représente forme une classe d'équivalence par rapport à une relation d'équivalence induite par un sous-groupe de permutations $\mathcal{H}$. Ici $\mathcal{H}$ est implicitement représenté par la partition locale $\mathfrak{Part}^{\mathfrak{l}}$.

Il est, bien sûr, inconcevable de tester tous les états possibles jusqu'à l'obtention de $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}\rangle$ dont le raffinement donne l'ensemble E. Aussi, expliciter tous les états ordinaires représentés par E puis trouver le sous-groupe de permutations qui permet de relier les états entre eux, n'est pas une solution envisageable, car elle est très coûteuse et élimine tout l'intérêt de l'approche symbolique.

L'idée que nous proposons ici repose sur l'exploitation des représentations symboliques des états de E. C'est-à-dire que nous allons utiliser ces représentations pour chercher et trouver (s'il existe) un état $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}\rangle$ qui les représente.

L'approche que nous allons développer est incrémentale, basée sur la définition d'une condition nécessaire, qui va ensuite faciliter la recherche effective de l'état représentant.

Ensemble d'états potentiellement regroupable. Le premier point consiste à définir une condition nécessaire (mais pas suffisante) qui permet de diriger la recherche de l'état $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}\rangle$.

Cette condition est triviale, car elle dicte que les éléments de E ne peuvent être candidats à un regroupement que si pour toute paire d'états $m, m' \in$

$\bigcup_{\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle \in E} [\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle]$, on peut trouver une permutation qui ramène m à m'. En effet, sans cette contrainte on risque de vouloir regrouper deux états qui ne sont pas "comparables". Par exemple, $m_1 = p_1(c_1) + p_2(c_2)$ ne peut être associé à l'état $m_2 = p_1(c_1 + c_2)$, alors qu'il peut l'être avec l'état $m_3 = p_1(c_2) + p_2(c_1)$.

Cette condition est facilement testable en utilisant le sous-groupe relâché $\mathcal{G}_s$ (définition 6.19). En fait, $\mathcal{G}_s$ regroupe toutes les permutations possibles sur le WN d'un CWN. Ainsi, pour vérifier cette condition, il suffit de vérifier que :

$$\bigcup_{\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle \in E} [\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle] \subseteq \mathcal{G}_s.m, \text{ où } m \in \bigcup_{\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle \in E} [\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle].$$

La formulation symbolique de cette condition est donnée par la définition suivante.

Définition 7.5 (Ensemble d'états potentiellement regroupable)**.** *Soit $\mathcal{C}$ une famille de classes, $E = \{\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}^1_{\mathfrak{Symb}^1} \rangle, \langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}^2_{\mathfrak{Symb}^2} \rangle \ldots\}$ un ensemble fini d'états symboliques. Soit $m \in \bigcup_{\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \in E \rangle} [\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle]$ et $\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^s_{\mathfrak{Symb}^{\mathfrak{s}}} \rangle$, une représentation symbolique de la classe $\mathcal{G}_s.m$. On a alors,*

$$E \text{ est potentiellement regroupable } \Leftrightarrow E \subseteq \text{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^s_{\mathfrak{Symb}^{\mathfrak{s}}} \rangle, \mathfrak{Part}^{\mathfrak{l}'})$$

L'état symbolique $\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^s_{\mathfrak{Symb}^{\mathfrak{s}}} \rangle$, appelé *état symbolique relâché*, est calculé par la fonction *EtatRel*, dont la description est la suivante : on choisit un état symbolique $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ dans E, on remplace sa partition locale $\mathfrak{Part}^{\mathfrak{l}'}$ par la partition relâchée $\mathfrak{Part}^{\mathfrak{s}}$, puis on ré-indexe ses sous classes dynamiques suivant cette nouvelle partition. Formellement,

$$\mathfrak{Symb}^{\mathfrak{s}} = \{\langle \mathfrak{Symb}'_i.\mathfrak{Dyn}_i, \mathfrak{Symb}'_i.card_i, d^s_i \rangle \mid i \in \{1, \ldots, n\}\}$$

$$\text{avec } \forall k \in \{1, \ldots, |\mathfrak{Symb}'_i.\mathfrak{Dyn}_i|\}, d^s_i(k) = 1.$$

Ceci est justifié par le fait que, par définition, $\forall i \in \{1, \ldots, n\}$ $|\mathfrak{Part}^{\mathfrak{s}}{}_i| = 1$. L'état symbolique obtenu est ensuite canonisé [4], pour garantir son unicité.

[4] voir section 6.3.2

$EtatRel(\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle)$:
début
  $\mathfrak{Symb}^{\mathfrak{s}} = \{\langle \mathfrak{Symb}'_i.\mathfrak{Dyn}_i, \mathfrak{Symb}'_i.card_i, d_i^{\mathfrak{s}} \rangle \mid i \in \{1,\ldots,n\}\}$
  pour tous les $p \in P$ **faire**
    pour tous les $Z \in \mathfrak{Symb}^{\mathfrak{s}}_{J(p)}$ **faire**
      $\hat{m}(p)(\Pi_{i=1}^{n}\Pi_{q=1}^{e_i}(Z_i^{w(i,q)}) = \hat{m}'(p)(\Pi_{i=1}^{n}\Pi_{q=1}^{e_i}Z_i^{w(i,q)})$
  $\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^{S}_{\mathfrak{Symb}^{\mathfrak{s}}} \rangle = Canonisation(\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}_{\mathfrak{Symb}^{\mathfrak{s}}} \rangle)$
  retourner $\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^{S}_{\mathfrak{Symb}^{\mathfrak{s}}} \rangle$
fin

Algorithme 3 : Calcul de l'état symbolique relâché de $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$.

Ainsi, si tous les éléments de E ont $\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^{S}_{\mathfrak{Symb}^{\mathfrak{s}}} \rangle$ comme état relâché, alors nous garantissons que E est potentiellement regroupable. Dans ce cas, chaque élément $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ de E est identifié par un élément unique dans $\mathrm{Raf}(\mathfrak{Part}^{\mathfrak{s}}, \mathfrak{Part}^{\mathfrak{l}'}, \mathfrak{Symb}^{\mathfrak{s}})$, qui est la partition symbolique associée à $\hat{m}'$. L'ensemble des partitions symboliques identifiant les éléments de E est noté Raf_E.

Il est à noter que si $Raf_E = \mathrm{Raf}(\mathfrak{Part}^{\mathfrak{s}}, \mathfrak{Part}^{\mathfrak{l}'}, \mathfrak{Symb}^{\mathfrak{s}})$ alors, $E = \mathrm{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^{S}_{\mathfrak{Symb}^{\mathfrak{s}}} \rangle, \mathfrak{Part}^{\mathfrak{l}})$ et E peut être regrouper en $\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^{S}_{\mathfrak{Symb}^{\mathfrak{s}}} \rangle$, ce qui termine la recherche. Cependant, dans le cas général $\mathrm{Raf}_E \subsetneq \mathrm{Raf}(\mathfrak{Part}^{\mathfrak{s}}, \mathfrak{Part}^{\mathfrak{l}'}, \mathfrak{Symb}^{\mathfrak{s}})$.

Ensemble d'états regroupable. En plus de la condition nécessaire, il faut maintenant définir les conditions qui assurent l'existence de l'état symbolique $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$, dont le raffinement est E. En d'autres termes, il faut trouver une partition symbolique $\mathfrak{Symb}$ telle que tout élément du raffinement symbolique $\mathrm{Raf}(\mathfrak{Part}^{\mathfrak{l}}, \mathfrak{Part}^{\mathfrak{l}'}, \mathfrak{Symb})$ identifie un élément unique de E.

Le point clé est que $\mathfrak{Part}^{\mathfrak{l}'}$ est nécessairement une partition plus fine que $\mathfrak{Part}^{\mathfrak{l}}$. Par conséquent, $\mathfrak{Part}^{\mathfrak{l}}$ est obtenue par le regroupement de certaines parties de $\mathfrak{Part}^{\mathfrak{l}'}$ (on ne regroupe que des parties de la même classe). Le cas le plus simple est que $\mathfrak{Part}^{\mathfrak{l}}$ est obtenue à partir de $\mathfrak{Part}^{\mathfrak{l}'}$, par le regroupement de deux parties $\mathcal{L}'_{i,j}$ et $\mathcal{L}'_{i,j'}$ (et uniquement celles-ci) : soit une partition symbolique $\mathfrak{Symb}$ associée à $\mathfrak{Part}^{\mathfrak{l}}$. Alors, suivant la définition 7.2, le raffinement de la partition symbolique $\mathfrak{Symb}$ par rapport à $\mathfrak{Part}^{\mathfrak{l}'}$ n'affecte que les sous classes dynamiques $Z_i^q \in \mathfrak{Symb}_i.\mathfrak{Dyn}_i$ telles que $\mathcal{L}_{i,d_i(q)} = \mathcal{L}'_{i,j} \cup \mathcal{L}'_{i,j'}$, sans modifier les autres.

Soit $Z_i^q \in \mathfrak{Symb}^{\mathfrak{s}}{}_i.\mathfrak{Dyn}_i$, soit $\mathfrak{Symb}' \in Raf_E$. Soit X (resp. Y) le nombre d'éléments de $\mathcal{L}'_{i,j}$ (resp. $\mathcal{L}'_{i,j'}$) représentés par Z_i^q pour $\mathfrak{Symb}'$. Puisqu'on veut confondre les éléments de $\mathcal{L}'_{i,j}$ et $\mathcal{L}'_{i,j'}$, Raf_E doit contenir toutes les partitions symboliques dans lesquelles $X+Y$ éléments sont répartis entre $\mathcal{L}'_{i,j}$ et $\mathcal{L}'_{i,j'}$.

Pour effectuer un regroupement, on va donc de manière itérative essayer de regrouper deux à deux les parties locales. La définition suivante donne les conditions pour procéder au regroupement de deux parties locales.

Définition 7.6 (Regroupement symbolique)**.** *Soit $\mathcal{C} = \bigcup_{i=1\ldots n} C_i$ une famille de classes, $E = \{\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}^1_{\mathfrak{Symb}^1}\rangle, \langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}^2_{\mathfrak{Symb}^2}\rangle, \ldots\}$ un ensemble fini d'états symboliques. E est regroupable par rapport à la partition locale $\mathfrak{Part}^{\mathfrak{l}} = (\mathfrak{Part}^{\mathfrak{l}'} \setminus \{\mathcal{L}'_{i,j'}, \mathcal{L}'_{i,j''}\}) \cup \{\mathcal{L}_{i,r}\}$, telle que $\mathcal{L}_{i,r} = \mathcal{L}'_{i,j'} \cup \mathcal{L}'_{i,j''}$, si :*

1. *E est potentiellement regroupable : $E \subseteq \mathrm{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^{\mathfrak{s}}_{\mathfrak{Symb}^{\mathfrak{s}}}\rangle, \mathfrak{Part}^{\mathfrak{l}'})$,*
2. *Soit $\mathrm{Raf}_E \subseteq \mathrm{Raf}(\mathfrak{Part}^{\mathfrak{s}}, \mathfrak{Part}^{\mathfrak{l}'}, \mathfrak{Symb}^{\mathfrak{s}})$ l'ensemble des partitions symboliques identifiant les éléments de E. Alors, $\forall \mathfrak{Symb}^1 \neq \mathfrak{Symb}^2 \in \mathrm{Raf}_E$,*
   (a) $\mathfrak{Symb}^1$ et $\mathfrak{Symb}^2$ doivent avoir des partitions symboliques identiques pour toutes les parties de $\mathfrak{Part}^{\mathfrak{l}'}$ différentes de $\mathcal{L}'_{i,j'}$ et $\mathcal{L}'_{i,j''}$:
   $\forall i, \forall k' \in \{1,\ldots,|\mathfrak{Symb}^1_i.\mathfrak{Dyn}_i|\}, \forall k'' \in \{1,\ldots,|\mathfrak{Symb}^2_i.\mathfrak{Dyn}_i\}$ tels que,
   – $\mathfrak{Symb}^1_i.d_i(k') \neq j' \wedge \mathfrak{Symb}^1_i.d_i(k') \neq j''$,
   – $\mathfrak{Symb}^1_i.d_i(k') = \mathfrak{Symb}^2_i.d_i(k'')$,
   – $\Upsilon_{\mathfrak{Symb}^1_i}(k') = \Upsilon_{\mathfrak{Symb}^2_i}(k'')$,
   on a, $\mathfrak{Symb}^1_i.card_i k' = \mathfrak{Symb}^2_i.card_i(k'')$.
   (b) Soit $\mathfrak{Symb}^1$ une partition symbolique quelconque de Raf_E. Le cardinal de l'ensemble E doit être égal au nombre de solutions du système d'équations suivant,

$$\begin{cases} \forall k \in \{1,\ldots,|\mathfrak{Symb}^{\mathfrak{s}}{}_i.\mathfrak{Dyn}_i|\} : \\ X^k + Y^k = \mathfrak{Symb}^{\mathfrak{s}}{}_i.card_i(k) - \sum_{k' \in D} \mathfrak{Symb}^1_i.card_i(k') \\ \quad avec\ D = \{k' \in \Upsilon^{-1}_{\mathfrak{Symb}^1_i}(k) | \mathfrak{Symb}^1.d_i(k') \neq j' \wedge \mathfrak{Symb}^1.d_i(k') \neq j''\} \\ \sum_{k \in \{1,\ldots,|\mathfrak{Symb}^{\mathfrak{s}}{}_i.\mathfrak{Dyn}_i|\}} X^k = |\mathcal{L}_{i,j'}| \\ \sum_{k \in \{1,\ldots,|\mathfrak{Symb}^{\mathfrak{s}}{}_i.\mathfrak{Dyn}_i|\}} Y^k = |\mathcal{L}_{i,j''}| \end{cases}$$

où $\forall k, X^k \in \mathbb{N}$ *et* $Y^k \in \mathbb{N}$.

Exemple 7.7. *Soit la famille de classes* $\mathcal{C} = \{C_1\}$*, où* $C_1 = \{pr_1, pr_2, pr_3\}$*. Soit* $\mathfrak{Part}^{\mathfrak{l}} = \{\mathfrak{Part}^{\mathfrak{l}}{}_1\} = \{\{\mathcal{L}_{1,1}, \mathcal{L}_{1,2}, \mathcal{L}_{1,3}\}\}$ *et* $\mathcal{L}_{1,1} = \{pr_1\}, \mathcal{L}_{1,2} = \{pr_2\}, \mathcal{L}_{1,3} = \{pr_3\}$*, la partition locale de* $\mathcal{C}$*.*

Soit un ensemble E d'états symboliques tel que : $E = \{\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}^1_{\mathfrak{Symb}}\rangle, \langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}^2_{\mathfrak{Symb}}\rangle\}$ *où* $\mathfrak{Symb}$ *est définie par :* $|Z^1_{1,1}| = |Z^2_{1,2}| = |Z^3_{1,3}| = 1$*. Et,* $\hat{m}_1 = Idl(Z^1_1) + Rq(Z^2_1 + Z^3_1)$ *;* $\hat{m}_2 = Idl(Z^2_1) + Rq(Z^1_1 + Z^3_1)$*. On veut décider du regroupement de ces deux états symboliques.*

Intuitivement, chacun des deux marquages symboliques représente exactement un état ordinaire. A savoir, $\hat{m}_1$ *représente* $m_1 = Idl(pr_1) + Rq(pr_2 + pr_3)$ *et* $\hat{m}_2$ *représente* $m_2 = Idl(pr_2) + Rq(pr_1 + pr_3)$*. On constate que nous pouvons permuter les processes* pr_1 *et* pr_2 *sur l'un des marquages pour obtenir l'autre. Ceci nous donne une idée sur l'existence d'un sous-groupe de permutations* $\mathcal{H}$ *induisant une relation d'équivalence entre les deux marquages, et donc la possibilité d'une représentation symbolique unique pour les deux.*

Appliquons maintenant notre définition pour appuyer cette intuition. Les éléments de E sont potentiellement regroupables car l'état relâchéassocié aux deux états symboliques est $\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^s_{\mathfrak{Symb}^{\mathfrak{s}}}\rangle$ *tel que :*
– $\mathfrak{Part}^{\mathfrak{s}} = \{\mathfrak{Part}^{\mathfrak{s}}{}_1\} = \{\{\mathcal{L}^s_{1,1}\}\}$ *avec* $\mathcal{L}^s_{1,1} = \{pr_1, pr_2, pr_3\}$,
– $\mathfrak{Symb}^{\mathfrak{s}}$ *est décrite par :* $|Z^1_{1,1}| = 1$ *et* $|Z^2_{1,1}| = 2$,
– $\hat{m^s} = Idl(Z^1_1) + Rq(Z^2_1)$.

Il faut maintenant tester les deux états symboliques sont effectivement regroupables. Pour cela, on essaye de rassembler les parties $\mathcal{L}_{1,1}$ *et* $\mathcal{L}_{1,2}$*. La condition* (a) *ne s'applique qu'à la sous-classe dynamique* Z^3_1*, qui n'est instanciée ni dans* $\mathcal{L}_{1,1}$ *ni* $\mathcal{L}_{1,2}$*. Puisque elle a la la même cardinalité dans les deux états symboliques, la condition* (a) *est vérifiée.*

Pour vérifier la condition (b)*, prenons comme référence l'état symbolique* $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m^1_{\mathfrak{Symb}}}\rangle$*. La surjection entre les sous classes dynamiques de cet état et l'état relâché*$\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m^s_{\mathfrak{Symb}^{\mathfrak{s}}}}\rangle$ *est définie par :* $\Upsilon_{\mathfrak{Symb}_1} : \{1, \ldots, 3\} \mapsto \{1, \ldots, 2\}$ *tel que :* $\Upsilon_{\mathfrak{Symb}_1}(1) = 1, \Upsilon_{\mathfrak{Symb}_1}(2) = 2$*,* $\Upsilon_{\mathfrak{Symb}_1}(3) = 2$*. Le système d'équations*

associé est donné par :

$$\begin{cases} X^1+Y^1 = \mathfrak{Symb}^{\mathfrak{s}}{}_1.card_1(1) - \sum_D \mathfrak{Symb}_1.card_1(k') = 1-0 = 1. \\ \quad avec\ D = \{k' \in \Upsilon^{-1}_{\mathfrak{Symb}_1}(1) | (\mathfrak{Symb}_1.d_1(k') \neq 1 \wedge \mathfrak{Symb}_1.d_1(k')) \neq 2\} \\ X^2+Y^2 = \mathfrak{Symb}^{\mathfrak{s}}{}_1.card_1(2) - \sum_{D'} \mathfrak{Symb}_1.card_1(k') = 2-1 = 1 \\ \quad avec\ D' = \{k' \in \Upsilon^{-1}_{\mathfrak{Symb}_1}(2) | (\mathfrak{Symb}_1.d_1(k') \neq 1 \wedge \mathfrak{Symb}_1.d_1(k')) \neq 2\} \\ X^1+X^2 = |\mathcal{L}_{1,1}| = 1 \\ Y^1+Y^2 = |\mathcal{L}_{1,2}| = 1 \end{cases}$$

Le nombre de solutions de ce systèmes est 2 : $(X^1=1, Y^1=0, X^2=0, Y^2=1)$ *et* $(X^1=0, Y^1=1, X^2=1, Y^2=0)$. *Ce qui correspond à la cardinalité de l'ensemble* E. *La condition* (b) *est donc satisfaite.*

Si un ensemble d'états est regroupable, alors est possible de calculer l'état symbolique qui le représente suivant la définition suivante.

Définition 7.7 (Représentation d'un ensemble d'états regroupable)**.** *Soit* $\mathcal{C} = \bigcup_{i=1\ldots n} C_i$ *une famille de classes,* $E = \{\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}^1_{\mathfrak{Symb}^1}\rangle, \langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}^2_{\mathfrak{Symb}^2}\rangle, \ldots\}$ *un ensemble fini d'états symboliques regroupable par rapport à la partition locale* $\mathfrak{Part}^{\mathfrak{l}} = \mathfrak{Part}^{\mathfrak{l}'} \setminus \{\mathcal{L}'_{i,j'}, \mathcal{L}'_{i,j''}\} \cup \{\mathcal{L}_{i,r}\}$, *telle que* $\mathcal{L}_{i,r} = \mathcal{L}_{i,j'} \cup \mathcal{L}_{i,j''}$. *Soit* $\mathfrak{Symb}'$ *une partition symbolique quelconque de* Raf_E *et* $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'}\rangle \in E$. *L'état symbolique* $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}\rangle$ *représentant de* E *est défini par :*

- $\mathfrak{Symb}$ *est une partition symbolique de* $\mathfrak{Part}^{\mathfrak{l}}$ *telle que pour tout* $i \in \{i,\ldots,n\}$ *il existe une surjection* $\Phi_i : \{1,\ldots,|\mathfrak{Symb}'_i.\mathfrak{Dyn}_i|\} \mapsto \{1,\ldots,|\mathfrak{Symb}_i.\mathfrak{Dyn}_i|\}$ *qui vérifie,*
  - $\forall k', k'' \in \{1,\ldots,|\mathfrak{Symb}'_i.\mathfrak{Dyn}_i|\}$ *tels que* $\mathfrak{Symb}'_i.d_i(k') \neq j', \mathfrak{Symb}'_i.d_i(k'') \neq j', \mathfrak{Symb}'_i.d_i(k') \neq j'', \mathfrak{Symb}'_i.d_i(k'') \neq j''$, *on a :*
    - $k' \neq k'' \Rightarrow \Phi_i(k') \neq \Phi_i(k'')$,
    - $\mathfrak{Symb}'_i.card_i(k') = \mathfrak{Symb}_i.card_i(\Phi_i(k'))$.
  - $\forall k', k'' \in \{1,\ldots,|\mathfrak{Symb}'_i.\mathfrak{Dyn}_i|\}$ *tels que* $\mathfrak{Symb}'_i.d_i(k') = j', \mathfrak{Symb}'_i.d_i(k'') = j''$ *on a :*
    - $\Upsilon_{\mathfrak{Symb}'_i}(k') = \Upsilon_{\mathfrak{Symb}'_i}(k'') \Rightarrow \Phi_i(k') = \Phi_i(k'')$,
    - $\mathfrak{Symb}_i.card_i(\Phi_i(k')) = \sum_{t \in \{1,\ldots,|\mathfrak{Symb}'_i.\mathfrak{Dyn}_i|\} | \Upsilon_{\mathfrak{Symb}'_i}(t) = \Upsilon_{\mathfrak{Symb}'_i}(k')} \mathfrak{Symb}'_i.card_i(t)$.
- $\hat{m}$ *est une fonction qui associe à chaque place* $p \in P$ *un multi-ensemble de* $\mathfrak{Symb}_{J(p)}$ *et vérifiant :*

$$\forall p \in P, \forall Z \in \mathfrak{Symb}'_{J(p)}, \hat{m}(p)(\Pi^n_{i=1}\Pi^{e_i}_{j=1} Z_i^{w(i,j)}) = \hat{m}'(p)(\Pi^n_{i=1}\Pi^{e_i}_{j=1} Z_i^{\Phi_i(w(i,j))})$$

Exemple 7.8. *L'état symbolique représentant de l'ensemble E de l'exemple précédent est* $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$ *tel que :*
– $\mathfrak{Part}^{\mathfrak{l}} = \{\mathfrak{Part}_1\} = \{\{\mathcal{L}_{1,1}, \mathcal{L}_{1,2}\}\}$, *où* $\mathcal{L}_{1,1} = \{p_1, p_2\}$ *et* $\mathcal{L}_{1,2} = \{p_3\}$,
– $\mathfrak{Symb}$ *est définie par :* $|Z^1_{1,1}| = |Z^2_{1,1}| = |Z^3_{1,2}| = 1$,
– $\hat{m}$ *est défini par :* $\hat{m} = Idl(Z^1_1) + Rq(Z^2_1 + Z^3_1)$.

Algorithme général du regroupement symbolique. L'algorithme 4 montre le processus complet du processus de regroupement, appliqué à un ensemble quelconque d'états symboliques. Par souci de légèreté, cet algorithme est présenté pour le cas d'une famille de classes $\mathcal{C}$ telle que $|\mathcal{C}| = 1$. Par conséquent, la partition locale $\mathfrak{Part}^{\mathfrak{l}}$ de $\mathcal{C}$ est constituée des parties d'une seule classe. Ainsi, une partie $\mathcal{L}_{i,j}$ est simplement notée $\mathcal{L}_j$. La généralisation au cas de plusieurs classes de couleurs ne pose aucun problème théorique.

La fonction $Condition_{Néc}$ partitionne l'ensemble des états symboliques *Set* de manière à assurer que les éléments de chaque partition satisfont la condition nécessaire (point 1) de la définition 7.6 : tous les états symboliques de chaque partition doivent avoir le même état relâché.

La fonction $Condition_a$ partitionne l'ensemble *Set* (en entrée) de manière à assurer que les éléments de chaque partition satisfont la condition (a) de la définition 7.6. La fonction $Condition_b$ vérifie la satisfaction de la condition (b) par l'ensemble *Set* (en entrée). La fonction *IndexSuivant* cherche la prochaine partition locale $\tilde{\mathcal{L}}$ à tester.

Main constitue le cœur de l'algorithme de regroupement, elle commence par partitionner l'ensemble des états symbolique *Set* par rapport à la condition nécessaire (ligne 31). Chaque partition est ensuite découpée pour satisfaire la condition (a) (ligne 36). Pour chacune d'elles nous testons la condition (b). Pour cela nous avons besoin de deux structures Gp et NGp. Si la condition (b) est vérifiée (lignes 37-41) alors, nous calculons l'état symbolique représentant (suivant la définition 7.7) et nous ajoutons le résultat dans Gp, sinon nous stockons l'ensemble des états non regroupables (suivant les partitions d'indices (j, j')) dans NGp. Les éléments appartenant à Gp sont remis sans la pile $todo$, s'il reste une possibilité de regroupement par rapport à une nouvelle partition (lignes 42-47). Nous effectuons la même opération pour les éléments appartenant à NGp (lignes 48-53).

Données :
$InitialSet = \{\langle \mathfrak{Part}^{\iota}, \hat{m}^1_{\mathfrak{Symb}^1}\rangle, \langle \mathfrak{Part}^{\iota}, \hat{m}^2_{\mathfrak{Symb}^2}\rangle, \ldots\}$ un ensemble fini d'états symboliques
$todo$: pile de $\langle \mathfrak{Part}^{\iota'}$: une partition locale, $j \in \mathbb{N}, j' \in \mathbb{N}, E$: ensemble d'états symboliques$\rangle$

Condition$_{Néc}$(*Set*) :
début
  /* Calculer *PNes*, une partition de *Set* t.q. :
  - $\forall E \in PNes, \forall e \neq e' \in E$: $EtatRel(e_1) = EtatRel(e_2)$
  - $\forall E, E' \in PNes, \forall e \in E, \forall e' \in E'$: $EtatRel(e_1) \neq EtatSym(e_2)$
  */
  retourner *PNes*
fin

Condition$_a(\mathfrak{Part}^{\iota'}, j, j', Set)$:
début
  /* Calculer $PCond_a$, une partition de *Set* t.q :
  - $\forall E \in PCond_a, \forall e \neq e' \in E$: e et e′ satisfont la condition (a) (déf. 7.6) p.r.a $\mathcal{L}_j$ et $\mathcal{L}_{j'}$ de $\mathfrak{Part}^{\iota'}$.
  - $\forall E, E' \in PCond_a, \forall e \in E, \forall e' \in E'$: e et e′ ne satisfont pas la condition (a) (déf. 7.6) p.r.a $\mathcal{L}_j$ et $\mathcal{L}_{j'}$ de $\mathfrak{Part}^{\iota'}$.
  */
  retourner $PCond_b$
fin

Condition$_b(\mathfrak{Part}^{\iota'}, j, j', Set)$:
début
  si *Set satisfait la condition* (b) *de la déf. 7.6 p.r.a* $\mathcal{L}_j, \mathcal{L}_{j'}$ *de* $\mathfrak{Part}^{\iota'}$ **alors**
    retourner $Vrai$
  sinon
    retourner $Faux$
fin

IndexSuivant$(\mathfrak{Part}^{\iota}, j, j')$:
début
  si $j' = |\mathfrak{Part}^{\iota}|$ **alors**
    si $j' = |\mathfrak{Part}^{\iota}| - 1$ **alors**
      retourner $\langle -1, -1\rangle$
    sinon
      retourner $\langle j+1, j+2\rangle$
  sinon
    retourner $\langle j, j'+1\rangle$
fin

```
26 Main() :
27 début
28     Res = ∅
29     PNes = Condition_Néc(InitialSet)
30     pour tous les E ∈ PNes faire
31         todo.push(⟨𝔓art^l, 1, 2, E⟩)
32         tant que ¬todo.empty() faire
33             ⟨𝔓art^l', j, j', E'⟩ = todo.pop()
34             PCond_a = Condition_a(𝔓art^l', j, j', E')
35             Gp = NGp = ∅
36             pour tous les E'' ∈ PCond_a faire
37                 si Condition_b(𝔓art^l', j, j', E'') alors
                       /* Calcul de la représentation symbolique de
                       E'' suivant la déf. 7.7                    */
38                     ⟨𝔓art^l'', m̂''_Symb''⟩ = CalRepSymb(𝔓art^l', j, j', E'')
39                     Gp = Gp ∪ {⟨𝔓art^l'', m̂''_Symb''⟩}
40                 sinon
41                     NGp = NGp ∪ E''
42             si Gp ≠ ∅ alors
43                 ⟨s, s'⟩ = IndexSuivant(𝔓art^l'', j, j' − 1)
44                 si ⟨s, s'⟩ = ⟨−1, −1⟩ alors
45                     Res = Res ∪ Gp
46                 sinon
47                     todo.push(⟨𝔓art^l'', s, s', Gp⟩)
48             si NGp ≠ ∅ alors
49                 ⟨s, s'⟩ = IndexSuivant(𝔓art^l', j, j')
50                 si ⟨s, s'⟩ = ⟨−1, −1⟩ alors
51                     Res = Res ∪ NGp
52                 sinon
53                     todo.push(⟨𝔓art^l', s, s', NGp⟩)
54     retourner Res
55 fin
```

Algorithme 4 : Algorithme du regroupement symbolique :RegSymb(*Set*).

7.3.2.3 Inclusion symbolique

Nos états symboliques représentent des ensembles d'états ordinaires et nos approches génériques tirent profit des inclusions qui existent entre les ensembles d'états, il est donc nécessaire de définir une opération d'inclusion qui peut être effectuée au niveau symbolique, c'est-à-dire, entre les représentations symboliques des ensembles d'états.

Soient deux états symboliques $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$ et $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$. Pour pouvoir tester l'inclusion entre ces deux états symboliques, il faut impérativement les réécrire sous *des formes comparables*. Une solution évidente serait d'expliciter l'ensemble des états ordinaires, représentés par chacun de ces deux états symboliques, puis d'effectuer un test d'inclusion standard entre ensembles. Cependant, cette solution annule tout l'intérêt de l'approche symbolique.

La solution que nous proposons repose sur l'exploitation *des symétries communes* entre les états symboliques. En effet, en réécrivant deux états symboliques sur la base de leurs symétries communes, il devient possible *d'uniformiser* les représentations au niveau symbolique. Dans ce cas, le test d'inclusion symbolique revient à un test d'inclusion standard entre ensembles.

Les permutations communes entre deux sous-groupes $\mathcal{H}$ et $\mathcal{H}'$ sont obtenues par le sous-groupe $\mathcal{H} \cap \mathcal{H}'$. Sachant que dans notre approche les sous-groupes sont implicitement représentés par des partitions locales, $\mathfrak{Part}^{\mathfrak{l}}$ et $\mathfrak{Part}^{\mathfrak{l}'}$, alors nous pouvons représenter les symétries communes par la partition $\mathfrak{Part}^{\mathfrak{l}} \cap \mathfrak{Part}^{\mathfrak{l}'}$.

Ainsi, l'uniformisation de la représentation des deux états symboliques s'effectue par un raffinement symbolique de $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$ et $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ par rapport à $\mathfrak{Part}^{\mathfrak{l}} \cap \mathfrak{Part}^{\mathfrak{l}'}$. On obtient les deux ensembles $\text{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^{\mathfrak{l}} \cap \mathfrak{Part}^{\mathfrak{l}'})$ et $\text{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle, \mathfrak{Part}^{\mathfrak{l}} \cap \mathfrak{Part}^{\mathfrak{l}'})$.

Définition 7.8 (Inclusion symbolique)**.** *Un état symbolique* $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$ *est dit inclus dans un état symbolique* $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ *ssi :*

$$\text{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^{\mathfrak{l}} \cap \mathfrak{Part}^{\mathfrak{l}'}) \subseteq \text{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle, \mathfrak{Part}^{\mathfrak{l}} \cap \mathfrak{Part}^{\mathfrak{l}'})$$

Exemple 7.9. *Soit* $\mathcal{C}_1 = \{pr_1, pr_2, pr_3\}$ *une classe d'objet. Soient* $\mathfrak{Part}^{\mathfrak{l}} = \{\mathfrak{Part}^{\mathfrak{l}}_1\} = \{\{\mathcal{L}_{1,1}\}\}$ *avec* $\mathcal{L}_{1,1} = \{pr_1, pr_2, pr_3\}$ *et* $\mathfrak{Part}^{\mathfrak{l}'} = \{\mathfrak{Part}^{\mathfrak{l}'}_1\} = \{\{\mathcal{L}'_{1,1}, \mathcal{L}'_{1,2}\}\}$ *avec* $\mathcal{L}'_{1,1} = \{pr_1\}$ *et* $\mathcal{L}'_{1,2} = \{pr_2, pr_3\}$*, deux partitions locales de*

C.

Soit $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$ *un état symbolique avec :*
– $\mathfrak{Symb} = \{\mathfrak{Symb}_1\}$ *définie par :* $|Z^1_{1,1}| = 1, |Z^2_{1,1}| = 2$,
– $\hat{m} = Idl(Z^1_1) + Rq(Z^2_1)$.

Soit $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ *un deuxième état symbolique avec :*
– $\mathfrak{Symb}' = \{\mathfrak{Symb}'_1\}$ *définie par* $|Z^1_{1,1}| = 1, |Z^2_{1,2}| = 2$,
– $\hat{m}' = Idl(Z^1_1) + Rq(Z^2_1)$.

Pour savoir si $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ *est inclus dans* $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$*, il faut d'abord calculer les symétries communes :* $\mathfrak{Part}^{\mathfrak{l}^C} = \mathfrak{Part}^{\mathfrak{l}} \cap \mathfrak{Part}^{\mathfrak{l}'} = \{\{\mathcal{L}^c_{1,1}, \mathcal{L}^c_{1,2}\}\}$ *telle que :* $\mathcal{L}^c_{1,1} = \{p_1\}$ *et* $\mathcal{L}^c_{1,2} = \{p_2, p_3\}$. *On remarque ici que les partitions* $\mathfrak{Part}^{\mathfrak{l}^C}$ *et* $\mathfrak{Part}^{\mathfrak{l}'}$ *sont identique.*

Nous calculons ensuite les ensembles $\mathrm{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^{\mathfrak{l}^C})$ *et* $\mathrm{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle, \mathfrak{Part}^{\mathfrak{l}^C})$. *Puisque* $\mathfrak{Part}^{\mathfrak{l}^C} = \mathfrak{Part}^{\mathfrak{l}'}$*, alors le raffinement de* $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ *est lui-même et* $\mathrm{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^{\mathfrak{l}^C}) = \{\langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}^1_{\mathfrak{Symb}^1} \rangle, \langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}^2_{\mathfrak{Symb}^2} \rangle\}$ *:*
– $\mathfrak{Symb}^1$ *est définie par :* $|Z^1_{1,1}| = 1, |Z^2_{1,2}| = 2$ *;* $\hat{m}^1 = Idl(Z^1_1) + Rq(Z^2_1)$.
– $\mathfrak{Symb}^2$ *est définie par :* $|Z^1_{1,1}| = |Z^2_{1,1}| = |Z^3_{1,2}| = 1$ *;* $\hat{m}^2 = Idl(Z^2_1) + Rq(Z^1_1 + Z^3_1)$ *;*

Nous remarquons que $\langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}^1_{\mathfrak{Symb}^1} \rangle = \langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$*. Par conséquent,* $\mathrm{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle, \mathfrak{Part}^{\mathfrak{l}^C}) \subseteq \mathrm{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^{\mathfrak{l}^C})$.

7.3.2.4 Décomposition symbolique

La décomposition d'un état symbolique est la version symbolique de l'opération Decomp introduite par la proposition 5.3. Soient $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$ et $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ deux états symboliques tels que $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ est inclus dans $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$. La décomposition symbolique est une opération qui rend la différence entre ces deux états.

Comme dans le cas de l'inclusion, pour effectuer cette opération symboliquement, il faut uniformiser les représentations des deux états sur la base des symétries communes. Ceci est réalisé par le raffinement symbolique par rapport à la parti-

tion $\mathfrak{Part}^{l} \cap \mathfrak{Part}^{l'}$, ensuite une opération de différence standard est réalisée sur les ensembles résultants.

Définition 7.9 (Décomposition symbolique)**.** *Soient* $\langle \mathfrak{Part}^{l}, \hat{m}_{\mathfrak{Symb}} \rangle$ *et* $\langle \mathfrak{Part}^{l'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ *deux états symboliques tels que le deuxième est inclus dans le premier. La décomposition symbolique de* $\langle \mathfrak{Part}^{l}, \hat{m}_{\mathfrak{Symb}} \rangle$ *par rapport à* $\langle \mathfrak{Part}^{l'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ *est définie par l'ensemble :*

$$\text{RafSymb}(\langle \mathfrak{Part}^{l}, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^{l} \cap \mathfrak{Part}^{l'}) \setminus \text{RafSymb}(\mathfrak{Part}^{l'}, \hat{m}'_{\mathfrak{Symb}'}, \mathfrak{Part}^{l} \cap \mathfrak{Part}^{l'}).$$

Exemple 7.10. *Reprenons l'exemple précédent. Après l'étape de raffinement, nous avons trouvé que* $\langle \mathfrak{Part}^{l^c}, \hat{m}^1_{\mathfrak{Symb}^1} \rangle = \langle \mathfrak{Part}^{l'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$*. Donc, la différence entre les deux états symboliques* $\langle \mathfrak{Part}^{l}, \hat{m}_{\mathfrak{Symb}} \rangle$ *et* $\langle \mathfrak{Part}^{l'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ *est représentée par l'état symbolique* $\langle \mathfrak{Part}^{l^c}, \hat{m}^2_{\mathfrak{Symb}^2} \rangle$*.*

7.3.3 Franchissement symbolique synchronisé dans les CWN

Nous définissons maintenant une règle de franchissement symbolique pour les CWN, qui permet un calcul automatique des successeurs d'un état symbolique $\langle \mathfrak{Part}^{l}, \hat{m}_{\mathfrak{Symb}}, l \rangle$ [5]. Ce franchissement symbolique est basé sur la règle de franchissement symbolique des WN, augmentée par la synchronisation avec une transition de l'automate de contrôle.

Afin de préserver l'intérêt de l'approche symbolique, la synchronisation avec les transitions de l'automate de contrôle doit s'opérer au niveau symbolique, c'est à dire, en utilisant les instances de franchissement symbolique au lieu des instances ordinaires. Pour réaliser cette *synchronisation symbolique*, nous définissons, pour chaque transitions $l \xrightarrow{\gamma} l'$ de l'automate, un sous-groupe de permutations $\mathcal{H}_\gamma$ sur l'ensemble des événements qui laisse γ invariant. Ce sous-groupe est implicitement représenté par une *partition admissible associée à* γ*.*

Définition 7.10 (Partition admissible associée à γ)**.** *Soit* $\mathcal{C}$ *une famille de classes,* $\mathfrak{Part}^{l}$ *une partition locale de* $\mathcal{C}$ *et* $\gamma \subseteq \mathcal{E}$ *un ensemble d'instances de franchissements.* $\mathfrak{Part}^{l}$ *est une partition admissible pour* γ *ssi :*

$$\forall t \in T, \forall \tilde{c} \in \mathfrak{Part}^{l}_{J(t)}, \forall c_1, c_2 \in \tilde{c}, (t, c_1) \in \gamma \Leftrightarrow (t, c_2) \in \gamma.$$

[5] Dans cette partie, nous ignorons l'état q du $\mathcal{TGBA}$ considéré car il n'a pas d'influence.

On note $Adm(\gamma)$ l'ensemble des partitions admissibles de γ, et on appelle partition admissible maximale, la partition $\mathfrak{Part}^{\gamma} \in Adm(\gamma)$ qui vérifie :

$$\forall t \in T, \forall \tilde{c_1}, \tilde{c_2} \in \mathfrak{Part}^{\gamma}_{J(t)}, \forall c_1 \in \tilde{c_1}, \forall c_2 \in \tilde{c_2}, (t, c_1) \in \gamma \Leftrightarrow (t, c_2) \notin \gamma.$$

L'écriture symbolique de γ par rapport à $\mathfrak{Part}^{\iota} \in Adm(\gamma)$ est $\langle \mathfrak{Part}^{\iota}, \hat{\gamma} \rangle$ tel que :

$$\langle \mathfrak{Part}^{\iota}, \hat{\gamma} \rangle = \{(t, \tilde{c}) \mid t \in T \wedge \tilde{c} \in \mathfrak{Part}^{\iota}{}_{J(t)} \wedge \exists c \in \tilde{c}, (t, c) \in \gamma\}.$$

Dans le cas de l'utilisation de la partition maximale, $\mathfrak{Part}^{\gamma}$ alors $\langle \mathfrak{Part}^{\gamma}, \hat{\gamma} \rangle$ est simplement noté $\hat{\gamma}$.

Exemple 7.11. *Soit $\mathcal{C} = \{C_1\}$ une famille de classe et $\mathfrak{Part}^{\iota} = \{\mathfrak{Part}^{\iota}{}_1\} = \{\{\mathcal{L}_{1,1}\}\}$ avec $\mathcal{L}_{1,1} = \{pr_1, pr_2, pr_3\}$.*

Considérons la transition $\langle l_0, t_6[x < y], l_0 \rangle$ de l'automate de contrôle de la Fig.7.1. Dans ce cas, $\gamma = \{(t_6, (pr_1, pr_2)), (t_6, (pr_2, pr_3)), (t_6, (pr_1, pr_3))\}$. Suivant la définition 7.10, l'ensemble $Adm(\gamma)$ est réduit à une unique partition $\mathfrak{Part}^{\iota} = \{\mathfrak{Part}^{\iota}{}_1\} = \{\{\mathcal{L}_{1,1}, \mathcal{L}_{1,2}, \mathcal{L}_{1,3}\}\}$ avec $\mathcal{L}_{1,1} = \{pr_1\}$, $\mathcal{L}_{1,2} = \{pr_2\}$ et $\mathcal{L}_{1,3} = \{pr_3\}$, et donc $\mathfrak{Part}^{\gamma} = \mathfrak{Part}^{\iota}$.

Le raffinement de $\mathfrak{Part}^{\iota}$ par rapport à $\mathfrak{Part}^{\iota}{}_{\gamma}$ est donné par : $\mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\iota}{}_{\gamma} = \mathfrak{Part}^{\iota}{}_{\gamma}$. L'écriture symbolique de γ par rapport à $\mathfrak{Part}^{\iota}{}_{\gamma}$ est donc $\hat{\gamma} = \{(t_6, (\mathcal{L}_{1,1}, \mathcal{L}_{1,2})), (t_6, (\mathcal{L}_{1,1}, \mathcal{L}_{1,3})), (t_6, (\mathcal{L}_{1,2}, \mathcal{L}_{1,3}))\}$.

Définition 7.11 (Règle de franchissement symbolique par synchronisation)**.** *Soient $N_{\mathcal{A}_c}$ un CWN, t une transition de N et $\langle l, \gamma, l' \rangle$ une transition de $\mathcal{A}_c$. Soit $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l \rangle$ un état symbolique tel que $\mathfrak{Part}^{\iota} \in Adm(\gamma)$. L'instance de transition symbolique $(t, [\lambda, \mu])$ est franchissable à partir de l'état $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l \rangle$, notée $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l \rangle [[(t, [\lambda, \mu])\rangle\rangle_{\gamma}$, ssi :*
– *$\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}} \rangle [[(t, [\lambda, \mu])\rangle\rangle$ et*
– *$\exists (t, \tilde{c}) \in \langle \mathfrak{Part}^{\iota}, \hat{\gamma} \rangle$ tel que $\forall i, k$: $\mathcal{L}_{i, \mathfrak{Symb}_i . d_i(\lambda_i(k))} = \tilde{c}_i^k$.*

L'état symbolique obtenu par ce franchissement est $\langle \mathfrak{Part}^{\iota}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle$ tel que, $\langle \mathfrak{Part}^{\iota}, \hat{m}'_{\mathfrak{Symb}'} \rangle = \langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}} \rangle [[(t, [\lambda, \mu])\rangle\rangle$.

Cas général. On est souvent amené à calculer les franchissements synchronisés d'un état $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l \rangle$ par rapport à une transition $l \xrightarrow{\gamma} l'$ tel que $\mathfrak{Part}^{\iota} \notin Adm(\gamma)$. On doit alors effectuer un raffinement de l'état symbolique par rapport à

$\mathfrak{Part}^{\gamma}$ pour nous ramener au cas exigé par la définition.

Les successeurs symboliques valides de $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l \rangle$ par rapport à la transition $l \xrightarrow{\gamma} l'$ seront donc définis par l'ensemble des successeurs à partir de chaque élément de RafSymb$(\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma})$: seules les instances de franchissement symbolique appartenant à $\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{\gamma} \rangle$ seront acceptées. Ces différentes étapes sont illustrées dans la Fig. 7.3.

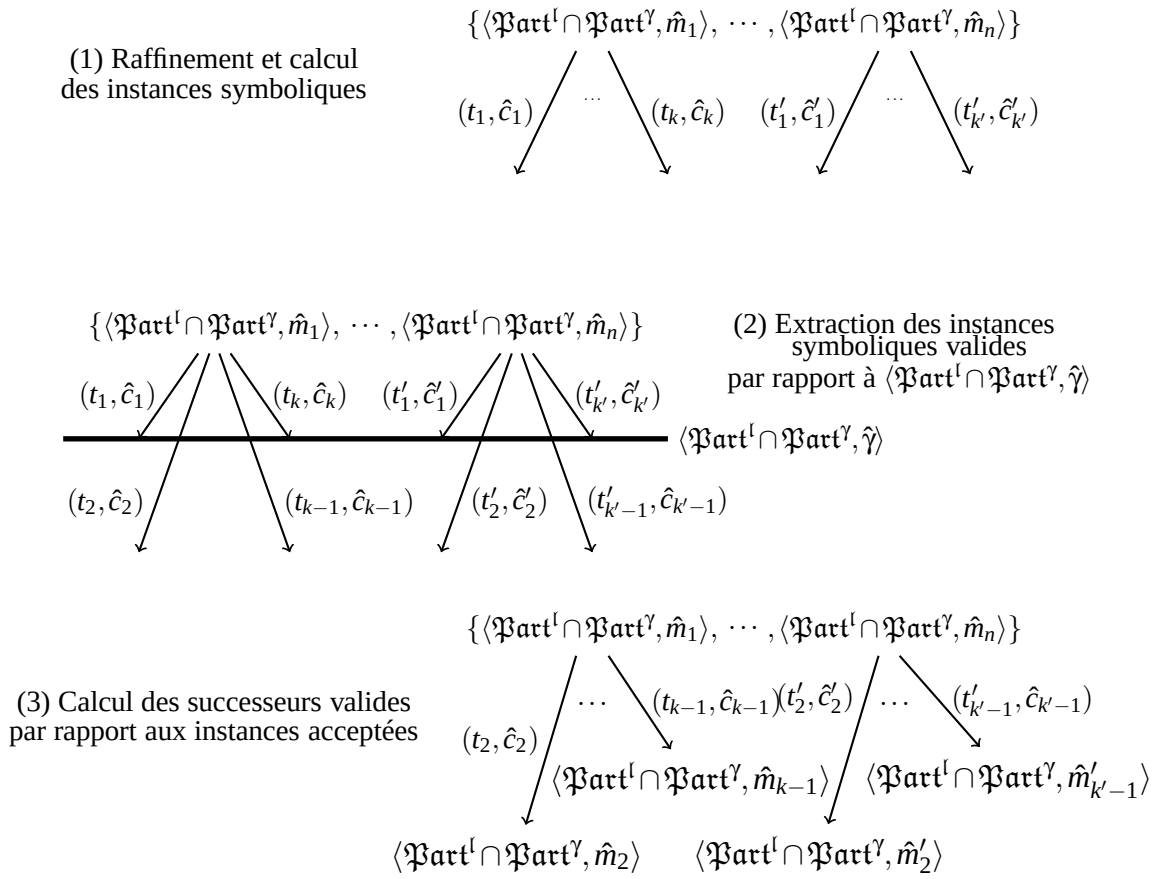


FIG. 7.3 – Franchissement symbolique synchronisé.

Exemple 7.12. *Soit l'état symbolique* $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l \rangle$ *où*
– $\mathfrak{Part}^{\iota} = \{\mathfrak{Part}^{\iota}_1\} = \{\{\mathcal{L}_{1,1}\}\}$ *avec* $\mathcal{L}_{1,1} = \{pr_1, pr_2, pr_3\}$,
– $\mathfrak{Symb}\{\mathfrak{Symb}_1\}$ *est défini par :* $|Z^1_{1,1}| = 2, |Z^2_{1,1}| = 1$,
– $\hat{m} = Wt(Z^1_1) + Sl(Z^2_1)$.

Le raffinement de $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}} \rangle$ *par rapport à* $\mathfrak{Part}^{\gamma}$ *de l'exemple précédent, produit les trois états symboliques* $\text{RafSymb}(\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}) = \{\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}^1_{\mathfrak{Symb}'} \rangle, \langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}^2_{\mathfrak{Symb}'} \rangle, \langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}^3_{\mathfrak{Symb}'} \rangle\}$ *tel que :*

- $\mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma} = \mathfrak{Part}^{\gamma}$,
- $\mathfrak{Symb}' = \{\mathfrak{Symb}'_1\}$ *définie par :* $|Z^1_{1,1}| = |Z^2_{1,2}| = |Z^3_{1,3}| = 1$,
- $\hat{m}^1 = Wt(Z^1_1 + Z^2_1) + Sl(Z^3_1)$,
- $\hat{m}^2 = Wt(Z^1_1 + Z^3_1) + Sl(Z^2_1)$,
- $\hat{m}^3 = Wt(Z^2_1 + Z^3_1) + Sl(Z^1_1)$.

Les instances symboliques associées à $\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}^1_{\mathfrak{Symb}'} \rangle$ *sont* $(t_6, (Z^3_1, Z^1_1))$ *et* $(t_6, (Z^3_1, Z^2_1))$. *Vis-à-vis de* $\langle \mathfrak{Part}^{\gamma}, \hat{\gamma} \rangle$, *aucune de ces deux instances n'est acceptée :*

- *pour* $\hat{c} = (t_6, (Z^3_1, Z^1_1))$, *nous avons* $\tilde{c} = (t_6, (\mathcal{L}_{1,\mathfrak{Symb}'_1.d_1(3)}, \mathcal{L}_{1,\mathfrak{Symb}'_1.d_1(1)})) = (t_6, (\mathcal{L}_{1,3}, \mathcal{L}_{1,1})) \notin \langle \mathfrak{Part}^{\gamma}, \hat{\gamma} \rangle$,
- *pour* $\hat{c} = (t_6, (Z^3_1, Z^2_1))$, *nous avons* $\tilde{c} = (t_6, (\mathcal{L}_{1,\mathfrak{Symb}'_1.d_1(3)}, \mathcal{L}_{1,\mathfrak{Symb}'_1.d_1(2)})) = (t_6, (\mathcal{L}_{1,3}, \mathcal{L}_{1,2})) \notin \langle \mathfrak{Part}^{\gamma}, \hat{\gamma} \rangle$.

Pour $\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}^2_{\mathfrak{Symb}'} \rangle$, *les instances sont* $(t_6, (Z^2_1, Z^1_1))$ *et* $(t_6, (Z^2_1, Z^3_1))$. *Seule deuxième instance est acceptée par* $\langle \mathfrak{Part}^{\gamma}, \hat{\gamma} \rangle$.

Enfin, l'état symbolique $\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}^3_{\mathfrak{Symb}'} \rangle\}$ *produit les instances* $(t_6, (Z^1_1, Z^3_1))$ *et* $(t_6, (Z^1_1, Z^2_1))$. *Dans ce cas, les deux instances sont acceptées par* $\langle \mathfrak{Part}^{\gamma}, \hat{\gamma} \rangle$ *:*

- *pour* $\hat{c} = (t_6, (Z^1_1, Z^3_1)))$, *nous avons* $\tilde{c} = (t_6, (\mathcal{L}_{1,\mathfrak{Symb}'_1.d_1(1)}, \mathcal{L}_{1,\mathfrak{Symb}'_1.d_1(3)})) = (t_6, (\mathcal{L}_{1,1}, \mathcal{L}_{1,3})) \in \langle \mathfrak{Part}^{\gamma}, \hat{\gamma} \rangle$,
- *pour* $\hat{c} = (t_6, (Z^1_1, Z^2_1))$, *nous avons* $\tilde{c} = (t_6, (\mathcal{L}_{1,\mathfrak{Symb}'_1.d_1(1)}, \mathcal{L}_{1,\mathfrak{Symb}'_1.d_1(2)})) = (t_6, (\mathcal{L}_{1,1}, \mathcal{L}_{1,2})) \in \langle \mathfrak{Part}^{\gamma}, \hat{\gamma} \rangle$.

Les états symboliques valides sont calculés par rapport aux instances acceptées, nous obtenons alors :

- $\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}^2_{\mathfrak{Symb}'} \rangle \xrightarrow{(t_6, (Z^2_1, Z^3_1))} \langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}^1_{\mathfrak{Symb}'} \rangle$;
- $\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}^3_{\mathfrak{Symb}'} \rangle \xrightarrow{(t_6, (Z^1_1, Z^3_1))} \langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}^1_{\mathfrak{Symb}'} \rangle$;
- $\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}^3_{\mathfrak{Symb}'} \rangle \xrightarrow{(t_6, (Z^1_1, Z^2_1))} \langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}^2_{\mathfrak{Symb}'} \rangle$.

7.3.4 Spécification des propriétés atomiques pour les CWN

Dans les CWN, l'information minimale que nous pouvons extraire d'un état du système est du type : *la couleur c marque la place p par une multiplicité* α_c. Puisque les domaines des classes du CWN sont finis, alors l'ensemble construit autour de cette information, pour toutes les places et toutes les couleurs, forme l'ensemble des propositions atomiques $\mathcal{AP}$.

Cependant, exprimer des propriétés temporelles sur la base de l'ensemble $\mathcal{AP}$ s'avère une opération très fastidieuse, car on doit gérer un nombre considérable de propositions atomiques, même quand il s'agit d'exprimer des contraintes simples sur le marquage des places.

Pour palier à ce problème, nous proposons un langage de spécification des propositions atomiques, exprimées sous forme de contraintes sur les marquages (des multi-ensembles de couleurs). Chaque contrainte peut être vue comme une écriture abrégée d'une conjonction de propositions atomiques de $\mathcal{AP}$, que nous appelons *contraintes atomiques*.

Pour exprimer ces contraintes, nous avons besoin de sélectionner certaines instances d'une classe à l'intérieur des domaines de couleurs des places.

Exemple 7.13. *Considérons deux places p_1 et p_2 avec le même domaine de couleur $C_1 \times C_1$. Nous voulons comparer le marquage de p_1 vis-à-vis de la première occurrence de C_1, avec le marquage de p_2 vis-à-vis de la deuxième occurrence de C_1. Par exemple, dans le marquage $m = p_1(\langle c_1, c_2\rangle) + p_2(\langle c_2, c_1\rangle)$, nous devons comparer $p_1(\langle c_1, -\rangle)$ avec $p_1(\langle -, c_1\rangle)$, i.e., on ignorant la deuxième occurrence de C_1 dans p_1 et la première occurrence de C_1 dans p_2.*

Soit une place p de domaine non neutre, $Cd(p) = \Pi_{i\in I} C_i^{e_i}$, $e_i \geqslant 0$. Dans ce qui suit, nous définissons une opération de sélection σ qui permet de choisir un ensemble d'occurrences de C_i dans $Cd(p)$, autant de fois que C_i apparaît dans $Cd(p)$.

Définition 7.12 (Sélection et restriction dans un domaine)**.** *Soit p une place et $Cd(p) = \Pi_{i\in I} C_i^{e_i}, e_i \geqslant 0$ son domaine.*

– *Une sélection dans $Cd(p)$ est une fonction $\sigma : I \to 2^{\mathbb{N}}$, vérifiant :*
Si $e_i = 0$ alors $\sigma(i) = \emptyset$ sinon $(\sigma(i) = \emptyset$ ou $\sigma(i) \subseteq \{1, \ldots, e_i\})$.
On note par $Sl(Cd(p))$ l'ensemble des sélections dans $Cd(p)$.

– *La restriction de $Cd(p)$ par rapport à $\sigma \in Sl(Cd(p))$ est définie par : $Cd(p)_{/\sigma} = \Pi_{i\in I} C_i^{e'_i}$ où $e'_i = |\sigma(i)|$. Si $\forall i, \sigma(i) = \emptyset$ alors $Cd(p)_{/\sigma} = \{\varepsilon\}$ (le domaine neutre).*

– *La restriction d'une couleur $c \in Cd(p)$ par rapport à $\sigma \in Sl(Cd(p))$, notée $c_{/\sigma}$, est définie par : $(\Pi_{i=1}^{n} \Pi_{j=1}^{e_i} c_i^j)_{/\sigma} = \Pi_{i=1}^{n} \Pi_{j\in\sigma(i)} c_i^j$. Si $\forall i, \sigma(i) = \emptyset$ alors $(\Pi_{i=1}^{n} \Pi_{j=1}^{e_i} c_i^j)_{/\sigma} = \varepsilon$ (la couleur neutre).*

Exemple 7.14. *Dans l'exemple précédent, nous avons $Cd(p_1) = Cd(p_2) = C_1^2$. L'ensemble $Sl(Cd(p_1))$ (Resp. $Sl(Cd(p_2))$) contient quatre selections :*
– $\sigma^1(1) = \emptyset$ *: aucune occurrence de C_1 n'est choisie ,*
– $\sigma^2(1) = \{1\}$ *: pour choisir la première occurrence de C_1,*
– $\sigma^3(1) = \{2\}$ *: pour choisir la deuxième occurrence de C_1,*
– $\sigma^4(1) = \{1,2\}$ *: pour choisir la les deux occurrences de C_1.*

La restriction de $Cd(p_1)$ par rapport à σ^1 est : $Cd(p_1)_{/\sigma^1} = C_1$. Celle de $Cd(p_2)$ par rapport à σ^2 : $Cd(p_2)_{/\sigma^2} = C_1$.

La restriction de la couleur $\langle c_1, c_2\rangle$ par rapport à σ^1 est $\langle c_1, c_2\rangle_{/\sigma^1} = \langle c_1\rangle$. Par rapport à σ^2, $\langle c_1, c_2\rangle_{/\sigma^2} = \langle c_2\rangle$.

Grâce à la définition précédente, nous pouvons maintenant donner la syntaxe et la sémantique du langage que nous proposons pour l'expression des contraintes atomiques sur un CWN.

Définition 7.13 (Contraintes atomiques sur un CWN)**.** *Soit p, p' deux places d'un CWN. Une contrainte atomique est une expression qui peut avoir les formes suivantes :*
– $\varepsilon_1 = \langle p, \sigma, op, ml\rangle$,
– $\varepsilon_2 = \langle p, \sigma, op, p', \sigma'\rangle$.
Où,
– $op \in \{<, \leq, >, \geq, =\}$,
– $ml \in Bag(Cd(p)_{/\sigma})$,
– $\sigma \in Sl(Cd(p))$,$\sigma' \in Sl(Cd(p'))$ *et* $Cd(p)_{/\sigma} = Cd(p')_{/\sigma'}$

La sémantique de ces propositions est la suivante :
ε_1 *: on prévoit que le marquage de p relativement à la sélection σ soit comparable au multi-ensemble ml, suivant l'opérateur op.*
ε_2 *: on prévoit que le marquage de p relativement à la sélection σ soit comparable*

au marquage de p' relativement à la sélection σ', suivant l'opérateur op.

Exemple 7.15. *Considérons le CWN de la Fig. 7.1. Nous voulons identifier les états tels que la place Rq, dont le domaine est $\mathcal{C}_1$, soit marquée uniquement par p_2. Ceci peut être exprimé par la contrainte atomique $\varepsilon = \langle Rq, \sigma, =, ml \rangle$ telle que $\sigma(1) = \{1\}$ et $ml = \langle p_2 \rangle$. La valeur de $\sigma(1)$ détermine l'occurrence de $\mathcal{C}_1$ dans $Cd(Rq)$ à laquelle on s'intéresse. Ici, Il n'y en a qu'une.*

Définition 7.14 (Satisfaction d'une contrainte atomique dans un état)**.** *Soient $\varepsilon_1 = \langle p, op, ml, \sigma \rangle$, $\varepsilon_2 = \langle p, op, p', \sigma, \sigma' \rangle$ des contraintes atomiques. On définit la satisfaction de $\varepsilon_1, \varepsilon_2$ sur un état (m, l) par les règles suivantes :*

1. $(m,l) \models \varepsilon_1 \Leftrightarrow \forall c \in Cd(p)_{/\sigma}, \sum_{c' \in Cd(p), c'_{/\sigma} = c} m(p)(c')\ op\ ml(c)$.
2. $(m,l) \models \varepsilon_2 \Leftrightarrow \forall c \in Cd(p)_{/\sigma}$,
$$\sum_{c' \in Cd(p), c'_{/\sigma} = c} m(p)(c')\ op\ \sum_{c' \in Cd(p'), c'_{/\sigma'} = c} m(p')(c').$$

La satisfaction que nous venons de définir suppose une représentation explicite des états du système, or notre approche est basée sur une représentation symbolique des classes d'états. Par conséquent, et pour bénéficier de l'apport du codage symbolique, il faut exprimer cette satisfaction en utilisant les représentations symboliques.

En fait, seule la contrainte atomique $\langle p, \sigma, op, ml \rangle$ nécessite un traitement spécial, car dans le cas général *ml* est un multi-ensemble sur $(Cd(p)_{/\sigma})$, faisant intervenir les couleurs pour l'expression de la contrainte. Or, dans un marquage symbolique les couleurs sont substituées par des sous-classes dynamiques, et nous devons donc trouver une expression symbolique pour *ml*.

La contrainte atomique $\langle p, \sigma, op, p', \sigma' \rangle$ ne faisant pas explicitement référence aux couleurs, ne nécessite pas de changement de représentation. Il faut juste définir la manière de la satisfaire sur un état symbolique.

Définition 7.15 (Partition admissible sur un multi-ensemble)**.** *Soit $\mathcal{C}$ une famille de classes, $\mathfrak{Part}^l$ une partition locale de $\mathcal{C}$ et $\mathcal{C}_J$ un domaine de couleurs. Soit $ml \in Bag(\mathcal{C}_J)$ un multi-ensemble sur $\mathcal{C}_J$. $\mathfrak{Part}^l$ est une partition admissible pour ml ssi :*

$$\forall \tilde{c} \in \mathfrak{Part}^l{}_J, \forall c_1, c_2 \in \tilde{c}, ml(c_1) = ml(c_2).$$

On note $Adm(ml)$ l'ensemble des partitions admissibles de ml, et on appelle partition admissible maximale, la partition $\mathfrak{Part}^{ml} \in Adm(ml)$ qui vérifie :

$$\forall \tilde{c}_1, \tilde{c}_2 \in \mathfrak{Part}_J^{ml}, \forall c_1 \in \tilde{c}_1, \forall c_2 \in \tilde{c}_2, ml(c_1) \neq ml(c_2).$$

L'écriture symbolique de ml par rapport à $\mathfrak{Part}^{\mathfrak{l}} \in Adm(ml)$ *est* $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{ml} \rangle$ *tel que :*

$$\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{ml} \rangle = \sum_{\tilde{c} \in \tilde{\mathfrak{Part}}^{\mathfrak{l}}_J} ml(c).\tilde{c}, \text{ où } c \in \tilde{c}.$$

Si la partition considérée est la partition maximale, alors $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{ml} \rangle$ *sera simplement notée* $\hat{ml}$.

Exemple 7.16. *Soit* $ml \in Bag(C_1)$ *un multi-ensemble tel que* $ml = \langle pr_2 \rangle$. *En appliquant la définition 7.15, les partitions admissibles* $Adm(ml)$ *de ml sont*
– $\mathfrak{Part}^{\mathfrak{l}} = \{\mathcal{L}_{1,1}, \mathcal{L}_{1,2}, \mathcal{L}_{1,3}\}$ *tel que* $\mathcal{L}_{1,1} = \{pr_1\}, \mathcal{L}_{1,2} = \{pr_2\}, \mathcal{L}_{1,3} = \{pr_3\}$.
– $\mathfrak{Part}^{\mathfrak{l}'} = \{\mathcal{L}'_{1,1}, \mathcal{L}'_{1,2}\}$ *tel que* $\mathcal{L}'_{1,1} = \{pr_1, pr_3\}$ *et* $\mathcal{L}'_{1,2} = \{pr_2\}$.
Dans ce cas, la partition symbolique maximale de ml est $\mathfrak{Part}^{ml} = \mathfrak{Part}^{\mathfrak{l}'}$. *L'écrite symbolique de ml par rapport à* $\mathfrak{Part}^{\mathfrak{l}'}$ *est* $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{ml} \rangle = \langle \mathcal{L}_{1,2} \rangle$.

Pour vérifier symboliquement une contrainte atomique, nous devons étendre les notions de sélection et de restriction, proposées dans la définition 7.12, aux partitions locales de domaine de couleurs, aux partitions symbolique de domaines de couleurs, aux n-uplets de parties, et aux n-uplets de sous-classes dynamiques. En fait, ceci est trivialement réalisé en substituant, dans la définition 7.12,le domaine de la place p ($Cd(p)$) par sa partition locale $\mathfrak{Part}^{\mathfrak{l}}_{J(p)}$ (Resp. symbolique $\mathfrak{Symb}_{J(p)}$, et la couleur $c \in Cd(p)$ par un n-uplets de parties $\tilde{c} \in \mathfrak{Part}^{\mathfrak{l}}_{J(p)}$ (Resp. un n-uplets de sous-classes dynamiques $Z \in \mathfrak{Symb}_{J(p)}$. Nous garderons le reste des notations inchangées.

Ainsi, la satisfaction d'une contrainte sur un état symbolique peut être définie da la manière suivante.

Définition 7.16 (Satisfaction d'une contrainte sur un état symbolique)**.** *Soient* $\varepsilon_1 = \langle p, \sigma, op, ml \rangle$, $\varepsilon_2 = \langle p, \sigma, op, p', \sigma' \rangle$ *des contraintes atomiques. Soit un état symbolique* $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}, l, q \rangle$, *tel que* $\mathfrak{Part}^{\mathfrak{l}} \in Adm(ml)$. *On définit la satisfaction symbolique de* ε_1, ε_2 *par* $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}, l, q \rangle$, *par les règles suivantes :*

1. $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l, q \rangle \models \varepsilon_1 \Leftrightarrow \forall \tilde{c} \in \mathfrak{Part}^{\mathfrak{l}}_{J(p)/\sigma}, \forall (\Pi_i^j \Pi_j^{e_i} Z_i^{w(i,j)}) \in \mathfrak{Symb}_{J(p)/\sigma}$ *t.q.* $\Pi_i^j \Pi_j^{e_i} \mathcal{L}_{i,\mathfrak{Symb}_i.d_i(w(i,j))} = \tilde{c}$,

$$\sum_{Z' \in \mathfrak{Symb}_{J(p)/\sigma}, Z'_{/\sigma} = Z} \hat{m}(p)(Z') \; op \; \langle \mathfrak{Part}^{\mathfrak{l}}, \hat{ml} \rangle (\tilde{c}).$$

2. $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l, q\rangle \models \varepsilon_2 \Leftrightarrow \forall Z \in \mathfrak{Symb}_{J(p)/\sigma}$,

$$\sum_{Z' \in \mathfrak{Symb}_{J(p)/\sigma}, Z'_{/\sigma}=Z} \hat{m}(p)(Z') \; op \sum_{Z' \in \mathfrak{Symb}_{J(p')/\sigma'}, Z'_{/\sigma}=Z} \hat{m}(p')(Z').$$

Cas général. Nous sommes souvent amenés à tester la satisfaction par un état symbolique $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l, q\rangle$, d'une contrainte atomique $\langle p, \sigma, op, ml\rangle$ telle que $\mathfrak{Part}^{\iota} \notin Adm(ml)$. Dans ce cas, la condition d'application de la définition 7.16 n'est plus satisfaite et donc, il n'est plus possible de tester la satisfaction de $\langle p, \sigma, op, ml\rangle$ directement sur $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l, q\rangle$.

Cependant, il est toujours possible de retrouver la condition exigée en opérant un raffinement symbolique sur l'état $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}\rangle$ par rapport à la partition $\mathfrak{Part}^{ml}$: $\text{RafSymb}(\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}\rangle, \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{ml}) = \{\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{ml}, \hat{m}'_{\mathfrak{Symb}'}\rangle, \ldots\}$. Dans ce cas, $(\mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{ml}) \in Adm(ml)$ et la satisfaction par l'état symbolique original est ramenée à la satisfaction symbolique par chaque état de son raffinement.

Exemple 7.17. *Soit un état symbolique* $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}\rangle$ *où :*
– $\mathfrak{Part}^{\iota} = \{\mathfrak{Part}^{\iota}_1\} = \{\{\mathcal{L}_{1,1}\}\}$ *avec* $\mathcal{L}_1 = \{pr_1, pr_2, pr_3\}$,
– $\mathfrak{Symb} = \{\mathfrak{Symb}_1\}$ *est définie par :* $|Z^1_{1,1}| = |Z^2_{1,1}| = |Z^3_{1,1}| = 1$,
– $\hat{m} = Id(Z^1_1) + Rq(Z^2_1) + Sl(Z^3_1)$.

Soit la contrainte atomique $\varepsilon = \langle Rq, \sigma, =, ml\rangle$ *telle que* $\sigma(1) = \{1\}$ *et* $ml = \langle p_2\rangle$. *La partition symbolique maximale associée à ml est* $\mathfrak{Part}_{ml} = \{\mathcal{L}'_{1,1}, \mathcal{L}'_{1,2}\}$ *tel que* $\mathcal{L}'_{1,1} = \{pr_1, pr_3\}$ *et* $\mathcal{L}'_{1,2} = \{pr_2\}$ *(voir exemple précédent).*

Puisque les deux partitions ne sont pas identiques, alors pour vérifier la satisfaction de ε *par* $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}\rangle$, *il faut procéder d'abord au raffinement de* $\mathfrak{Part}^{\iota}$ *par rapport à* $\mathfrak{Part}^{ml}$ *et nous obtenons,* $\mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{ml} = \mathfrak{Part}^{ml}$. *Par conséquent, l'écriture symbolique de ml par rapport à* $\mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{ml}$ *est* $\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}_{ml}, \hat{ml}\rangle = \langle \mathcal{L}'_{1,2}\rangle$.

Le raffinement de $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}\rangle$ *par rapport à* $\mathfrak{Part}^{\iota} \cap \mathfrak{Part}_{ml}$ *produit les trois états symboliques* $\text{RafSymb}(\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}\rangle, \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{ml}) = \{\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{ml}, \hat{m}^1_{\mathfrak{Symb}'}\rangle, \langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{ml}, \hat{m}^2_{\mathfrak{Symb}'}\rangle, \langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{ml}, \hat{m}^3_{\mathfrak{Symb}'}\rangle\}$ *tel que :*
– $\mathfrak{Symb}' = \{\mathfrak{Symb}'_1\}$ *définie par :* $|Z^1_{1,1}| = |Z^2_{1,2}| = |Z^3_{1,3}| = 1$,
– $\hat{m}^1 = Id(Z^1_1) + Rq(Z^3_1) + Sl(Z^2_1)$,

– $\hat{m}^2 = Id(Z_1^1) + Rq(Z_1^2) + Sl(Z_1^3)$,
– $\hat{m}^3 = Id(Z_1^3) + Rq(Z_1^1) + Sl(Z_1^2)$.

A présent, il est possible de vérifier la satisfaction de ε *symboliquement sur chaque état de* RafSymb$(\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^{\mathfrak{l}} \cap \mathfrak{Part}^{ml})$. *On note que le seul état symbolique qui vérifie* ε *est* $\langle \mathfrak{Part}^{\mathfrak{l}} \cap \mathfrak{Part}^{ml}, \hat{m}^2_{\mathfrak{Symb}'} \rangle$ *car la place Rq est marquée par la seule sous classe dynamique* Z_1^2 *dont la partie associée est* $\mathcal{L}'_{1,2}$ *(*$\mathfrak{Symb}'.d_i(2) = 2$*), et ceci correspond à* $\langle \mathfrak{Part}^{\mathfrak{l}} \cap \mathfrak{Part}^{ml}, \hat{ml} \rangle$.

7.3.5 Algorithmes de construction du produit synchronisé symbolique

En se basant sur les définitions précédentes, nous pouvons maintenant donner un algorithme général pour la construction de SSP. Deux cas sont à prévoir : le cas de la vérification d'une propriété temporelle et le cas de la vérification de la propriété d'accessibilité.

7.3.5.1 Cas d'une propriété temporelle

La construction du produit synchronise symbolique est réalisée *à la volée*, sous le contrôle d'un algorithme de test de vacuité (en l'occurrence, celui développé dans la section 5.3). Dans ce cas, il suffit de définir une procédure qui construit, *à la demande*, les successeurs symboliques d'un état état symbolique.

Dans cette procédure, $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l, q \rangle$ est l'état symbolique dont l'algorithme de test de vacuité veut connaître les successeurs, vis-à-vis de la transition $q \xrightarrow{P,F} q'$ du $\mathcal{TGBA}$ considéré, et respectant la transition $l \xrightarrow{\gamma} l'$ de l'automate de contrôle.

D'abord, les symétries communes sont calculées. Ceci est réalisé par un raffinement des différentes partitions : $\mathfrak{Part}^{\mathfrak{l}^c} = \mathfrak{Part}^{\mathfrak{l}} \cap \mathfrak{Part}^{\gamma} \cap \mathfrak{Part}^{P}$ (ligne 4). Ici, P est une conjonction de contraintes atomiques et $\mathfrak{Part}^{P}$ est calculée par l'intersection de toutes les partitions admissibles maximales de ses contraintes.

Pour chaque état symbolique issu du raffinement symbolique RafSymb $(\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l, q \rangle, \mathfrak{Part}^{\mathfrak{l}^c})$, nous testons la satisfaction symbolique des différentes contraintes atomiques de l'ensemble P et nous calculons les instances symboliques franchissables à partir de cet état, en respectant la synchronisation par rapport à l'automate de contrôle (ligne 6). Les successeurs sont ensuite construits et rajoutés à l'ensemble *Succ* (lignes 7 et 8). Une fois tous les successeurs connus, nous procédons au regroupement symbolique, afin de réduire la représentation de

l'ensemble *Succ* (ligne 9).

```
/* ⟨𝔓art^ι, m̂_Sψmb, l, q⟩ : un état symbolique                    */
/* l —γ→ l' : une transition de 𝒞𝒜                                 */
/* q —P,F→ q' : une transition du 𝒯𝒢ℬ𝒜                             */
```

1 *SuccesseursSymboliques*$(\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l, q\rangle, l \xrightarrow{\gamma} l', q \xrightarrow{P,F} q')$:
2 **début**
3 $Succ = \emptyset$
4 $\mathfrak{Part}^{\iota_C} = \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma} \cap \mathfrak{Part}^{P}$
5 **pour tous les** $\langle \mathfrak{Part}^{\iota_C}, \hat{m}^{c}_{\mathfrak{Symb}^c}, l, q\rangle \in \text{RafSymb}(\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l, q\rangle, \mathfrak{Part}^{\iota_C})$ **faire**
6 **si** $\langle \mathfrak{Part}^{\iota_C}, \hat{m}^{c}_{\mathfrak{Symb}^c}, l, q\rangle \models \varepsilon, \forall \varepsilon \in P \wedge \exists (t, \hat{c}), \langle \mathfrak{Part}^{\iota_C}, \hat{m}^{c}_{\mathfrak{Symb}^c}\rangle [[(t, \hat{c})\rangle\rangle_{\gamma}$ **alors**
7 $\langle \mathfrak{Part}^{\iota_C}, \hat{m}'_{\mathfrak{Symb}'}\rangle = \langle \mathfrak{Part}^{\iota_C}, \hat{m}^{c}_{\mathfrak{Symb}^c}\rangle [[(t, \hat{c})\rangle\rangle$
8 $Succ = Succ \cup \{\langle \mathfrak{Part}^{\iota_C}, \hat{m}'_{\mathfrak{Symb}'}, l', q'\rangle\}$
9 **retourner** RegSymb(*Succ*)
10 **fin**

Algorithme 5 : Calcul des successeurs symboliques dans SSP.

7.3.5.2 Cas de la propriété d'accessibilité

Si l'on s'intéresse uniquement à la propriété d'accessibilité alors l'utilisation de l'algorithme 6 est plus efficace, car il n'est plus concerné par la recherche de cycles (donc pas de décomposition). Nous appellerons la structure construite SSP_{Acc}.

Dans cet algorithme, une attention particulière est donnée à la procédure d'ajout d'un état symbolique dans la liste des états à traiter : un état symbolique représentant un grand nombre d'états ordinaires et ne permettant qu'un nombre restreint de franchissements asymétriques est plus prioritaire que les autres.

7.3.5.3 Optimisation de la recherche des états

Chercher si un état est inclus ou inclut un autre état est une opération coûteuse, qui agit sur les performances du processus de vérification (du point de vue temps de calcul). Par conséquent, il faut trouver une organisation des états, qui permet d'accélérer cette opération : une organisation acceptable est donc celle qui permet

/* $N = \langle P, T, \mathcal{C}, Cd, Pre, Post, Inh, \phi, \pi \rangle$: un WN symétrique */
/* $\mathcal{A}_c = \langle L, l_0, \mathcal{E}, \mathcal{R} \rangle$: un automate de contrôle */
/* $\langle \mathfrak{Part}^{\mathfrak{l}}{}_0, \hat{m}^0_{\mathfrak{Symb}}, l_0 \rangle$: un état symbolique initial */

ConstructionSSP$_{Acc}$() :
début
  SSP$_{Acc}$.AjouterNœud ($\langle \mathfrak{Part}^{\mathfrak{l}}{}_0, \hat{m}^0_{\mathfrak{Symb}}, l_0 \rangle$)
  ATraiter.Inserer($\langle \mathfrak{Part}^{\mathfrak{l}}{}_0, \hat{m}^0_{\mathfrak{Symb}}, l_0 \rangle$)
  tant que *ATraiter* $\neq \emptyset$ **faire**
    $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l \rangle$=ATraiter.Retirer()
    pour tous les (l', γ) *t.q.* $l \xrightarrow{\gamma} l' \in \mathcal{R}$ **faire**
      $Succ = \emptyset$
      $\mathfrak{Part}^{\mathfrak{l}^c} = \mathfrak{Part}^{\mathfrak{l}} \cap \mathfrak{Part}^{\gamma}$
      pour tous les
      $\langle \mathfrak{Part}^{\mathfrak{l}^c}, \hat{m}^c_{\mathfrak{Symb}^c}, l \rangle \in$ RafSymb($\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l \rangle, \mathfrak{Part}^{\mathfrak{l}^c}$) **faire**
        pour tous les $(t, \hat{c})$ *t.q.* $\langle \mathfrak{Part}^{\mathfrak{l}^c}, \hat{m}^c_{\mathfrak{Symb}^c} \rangle [[(t, \hat{c})]\rangle_{\gamma}$ **faire**
          $\langle \mathfrak{Part}^{\mathfrak{l}^c}, \hat{m}'_{\mathfrak{Symb}'} \rangle = \langle \mathfrak{Part}^{\mathfrak{l}^c}, \hat{m}^c_{\mathfrak{Symb}^c} \rangle [[(t, \hat{c})]\rangle$
          si $\nexists \langle \mathfrak{Part}^{\mathfrak{l}''}, \hat{m}''_{\mathfrak{Symb}''}, l' \rangle \in$ *SSP$_{Acc}$.Nœuds() t.q.*
          $\langle \mathfrak{Part}^{\mathfrak{l}^c}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle \subseteq \langle \mathfrak{Part}^{\mathfrak{l}''}, \hat{m}''_{\mathfrak{Symb}''}, l' \rangle$ **alors**
            $Succ = Succ \cup \{\langle \mathfrak{Part}^{\mathfrak{l}^c}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle\}$
          sinon
            SSP$_{Acc}$.AjouterArc($\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l \rangle \rightarrow \langle \mathfrak{Part}^{\mathfrak{l}''}, \hat{m}''_{\mathfrak{Symb}''}, l' \rangle$)
      pour tous les $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle \in$ RegSymb($Succ$) **faire**
        SSP$_{Acc}$.AjouterNœud($\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle$)
        SSP$_{Acc}$.AjouterArc($\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l \rangle \rightarrow \langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle$)
        ATraiter.Inserer($\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle$)
  retourner *SSP$_{Acc}$*
fin

Algorithme 6 : Construction de la structure *SSP$_{Acc}$*.

de limiter la recherche à un nombre réduit d'états.

Nous utiliserons pour cela le sous-groupe $\mathcal{G}_s$, associé au WN d'un CWN. Ce sous-groupe définit la plus large relation d'équivalence entre les marquages du WN, celle qui ne prend en compte aucune asymétrie : si m est un marquage de WN alors $m \in [\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^{s}_{\mathfrak{Symb}^{\mathfrak{s}}} \rangle]$, où $\mathfrak{Part}^{\mathfrak{s}}$ est la partition associée à $\mathcal{G}_s$ et $\hat{m}_s$ est la représentation symbolique de m par rapport à $\mathfrak{Part}^{\mathfrak{s}}$.

Pour tout $\mathcal{H} \subseteq \mathcal{G}_s$, $\mathcal{H}.m$ est l'orbite de m par rapport à $\mathcal{H}$ (définition 3.1) et on a $\mathcal{H}.m \subseteq \mathcal{G}_s.m$: l'orbite d'un marquage par rapport à un groupe contient toujours celle par rapport à l'un de ses sous-groupes. Donc, Pour tout état symbolique $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$ tel que $m \in [\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle]$, on a toujours $[\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle] \subseteq [\langle \mathfrak{Part}^{\mathfrak{s}}, \hat{m}^{s}_{\mathfrak{Symb}^{\mathfrak{s}}} \rangle]$.

Ainsi, tout état $\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ susceptible d'inclure un état $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle$ vérifie la condition nécessaire : $[\langle \mathfrak{Part}^{\mathfrak{l}'}, \hat{m}'_{\mathfrak{Symb}'} \rangle] \subseteq [\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle]$.

Pour alléger le poids de la recherche, nous pouvons donc organiser les états en ensembles tels que les états de chaque ensemble référencent un même état $\langle \tilde{\mathcal{C}}_s, \hat{m}_s \rangle$. Pour plus d'efficacité, les états symboliques à l'intérieur de chaque ensemble sont réparties suivant les états l et q.

7.4 Outils et expérimentations

Toutes les approches symboliques que nous avons présentées ont été implémentées en réutilisant le noyau de GreatSPN[6]. Initialement, cette outil a été développé pour la construction du SRG des WN, incluant une gestion efficace des représentations symboliques [CG95].

L'extension que nous avons apportée consiste en l'ajout du module *DySy* (gestionnaire des symétries dynamiques), ainsi que les différents modules permettant l'intégration des approches ESRG et SSP (la partie encadrée de la Fig.7.4).

L'algorithme de test de vacuité que nous avons présenté est implémenté dans Spot[7]. Nous avons connecté les deux outils pour comparer différentes techniques.

[6] `http://www.di.unito.it/~greatspn/`
[7] `http://spot.lip6.fr/`

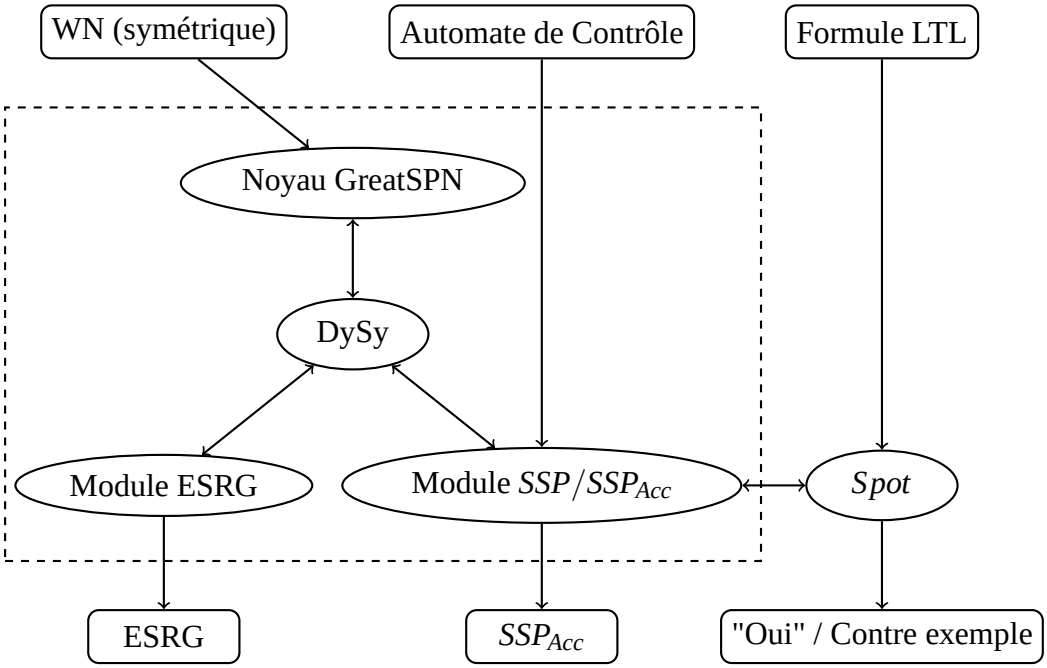


FIG. 7.4 – Architecture abstraire de l'outil de vérification réalisé.

Le tableau 7.1 présente quelques résultats obtenus sur le modèle de la section critique distribuée de la Fig. 7.1. Ce tableau compare les approches SRG, ESRG et SSP_{Acc} dans le cadre de la verification de la propriété d'accessibilité. Chaque ligne de la colonne *Modèle* référence le nombre de processus utilisé lors du test. *st*. et *t*. représentent respectivement le nombre d'états symboliques et le temps nécessaire à la construction de chaque structure.

	SRG(RG)		ESRG		SSP_{Acc}	
modèle	ét.	t.	ét.	t.	ét.	t.
WCS(3)	139	0	54	0	28	0
WCS(5)	2709	2	441	1	96	0
WCS(7)	50159	41	4918	25	253	3
WCS(9)	911017	1147	57211	939	559	45
WCS(11)	16378179	—	639056	54074	1090	1830

TAB. 7.1 – $SRG(RG)$ vs. $ESRG$ vs. SSP_{Acc}

On note ici le gain exponentiel dans le nombre d'états construits ainsi que dans le temps nécessaire à la construction de la strcuture SSP_{Acc} par rapport aux deux autres structure.

Le tableau 7.2 présente quelques mesures sur deux modèles paramètrables : WCS et PO [HTMK+04]. Le deuxième est le modèle d'un inter-géciel schizophrène, dans lequel nous augmentons le nombre de ressources et deux threads (2-2,2-3 et 3-2).

				SP+TEC			SSP+TEC		
modèle		*n*		ét.	tr.	T	ét.	tr.	T
WCS(3)	asym.	28	produit non vide	28	80	0.05	25	58	0.06
WCS(4)	asym.	28	produit non vide	78	250	0.06	77	202	0.14
WCS(5)	asym.	28	produit non vide	290	979	0.13	416	1309	2.76
PO(2,2)	sym.	18	produit non vide	252	431	0.13	690	1307	0.95
PO(2,3)	sym.	18	produit non vide	292	511	0.17	770	1441	1.48
PO(3,2)	sym.	22	produit non vide	1173	2235	0.65	2184	4730	7.40
WCS(3)	asym.	22	produit vide	99	279	0.06	94	255	0.13
WCS(4)	asym.	22	produit vide	434	1485	0.13	602	2063	1.60
WCS(5)	asym.	22	produit vide	1889	7430	0.60	4224	17744	144
PO(2,2)	sym.	32	produit vide	2484	5482	0.91	866	1817	2.16
PO(2,3)	sym.	32	produit vide	3253	7200	1.56	952	2030	3.68
PO(3,2)	sym.	28	produit vide	4617	10651	2.48	1334	2848	4.97

	SSP+NSIEC			SSP+IEC			SSP+AEC		
modèle	ét.	tr.	T	ét.	tr.	T	ét.	tr.	T
WCS(3)	26	51	0.06	24	45	0.05	21	39	0.05
WCS(4)	66	176	0.17	43	96	0.10	29	57	0.06
WCS(5)	294	1118	6.44	106	287	0.95	39	82	0.07
PO(2,2)	738	1505	1.10	738	1505	1.10	738	1505	1.10
PO(2,3)	750	1550	1.69	750	1550	1.69	750	1550	1.69
PO(3,2)	1400	3031	3.75	1400	3031	3.75	1392	2982	3.71
WCS(3)	91	250	0.14	73	194	0.13	30	70	0.07
WCS(4)	568	1980	2.17	297	940	1.07	64	177	0.15
WCS(5)	3905	16719	107	1370	4815	23.7	136	428	0.46
PO(2,2)	864	1814	2.14	865	1814	2.14	864	1813	2.14
PO(2,3)	868	1830	3.08	868	1831	3.09	868	1830	3.08
PO(3,2)	1294	2784	4.78	1294	2784	4.78	1294	2783	4.79

TAB. 7.2 – États (ét.) et transitions (tr.) explorées par chaque algorithme sur différents modèles, et temps (T) passé. Moyennes sur *n* propriétés.

Chacun de ces modèles a été synchronisé avec 50 automates (propriétés) : le tableau est décomposé en deux parties. Dans la première partie nous avons regroupé les résultats où le produit est vide (l'algorithme de test de vacuité ne trouve pas de d'exécution acceptante), tandis que la deuxième regroupe ceux dont le produit

contient une exécution acceptante. Cette séparation est nécessaire pour l'interprétation des résultats, car dans le cas d'un produit vide, le test de vacuité doit parcourir (en fait, c'est une construction à la volée) la totalité des états de l'automate, alors que pour un produit non vide, l'algorithme se termine dès la découverte de la première SCC acceptante.

Les abréviations dans les entêtes référencent la manière dont le produit a été construit ainsi que l'algorithme de test de vacuité utilisé. SP est le produit synchronisé de la définition 2.12 tandis que SSP est le produit synchronisé de la définition 4.4. Les états de SP ne sont pas des ensembles, donc cet automate est vérifié par un test de vacuité traditionnel (TEC), similaire à l'algorithme 1 mais sans les inclusions et les décompositions. IEC désigne le test de vacuité de l'algorithme 1. NSIEC est le même algorithme sans les lignes 40–43 (i.e., sans inclusion dans la pile de recherche). Finalement, AEC désigne le test de vacuité approximatif de la section 5.4.

Nous observons que bien que SP soit beaucoup plus rapide que SSP, il visite beaucoup plus d'états et donc exige beaucoup plus de mémoire. Les différentes versions de notre algorithme de test de vacuité peuvent être comparées sur les quatre colonnes (SSP) : sur le modèle WCS, l'ajout des tests d'inclusion dans l'ensembles des SCC retirées (NSIEC) réduit la taille de l'automate exploré (en comparaison avec TEC), et l'ajout des tests d'inclusion dans la pile de recherche (IEC) réduit l'automate encore plus.

Par conséquent, même si l'opération de décomposition est coûteuse (en terme de temps), elle est indispensable pour la réduction de la consommation mémoire du processus de vérification. La dernière colonne montre que la l'approche approximative est plus rapide et construit moins d'états que toutes les autres approches. Nous noterons ici, que sur ces modèles, l'approche approximative ne provoque pas de faux contre-exemples. Sur le modèle PO, les nouveaux algorithmes de test de vacuité ne sont pas meilleurs que le test de vacuité classique car le modèle est *symétrique* et les tests d'inclusion ne se produisent que très rarement.

La figure suivante indique la consommation mémoire utilisée pour la vérification du modèle WCS par les algorithmes SP+TEC et SSP+AEC.

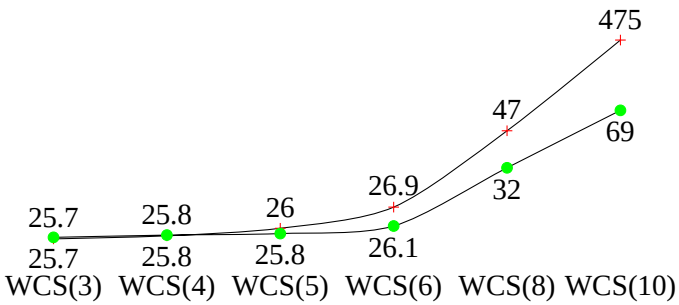


FIG. 7.5 – Moyenne de la mémoire utilisé (en MB, sur une échelle logarithmique) par nos implémentations de SP+TEC (au-dessus) et SSP+AEC (en-dessous) pour la vérification du modèle WCS.

Deuxième partie

Évaluation de performances des systèmes concurrents finis

Chapitre 8

Évaluation de performances et réseaux stochastiques bien formés

Le comportement quantitatif des systèmes concurrents est généralement approché par l'utilisation des processus aléatoires comme modèle mathématique de représentation. Cependant, le choix d'un modèle particulier est souvent guidé par l'exactitude et la complexité de la solution numérique qu'il apporte. Dans ce contexte, les processus markoviens, et spécialement les processus markoviens à espace d'états discret (appelés *chaînes de Markov*), se trouvent être d'une grande utilité car en plus d'être représentatifs du comportement d'un grand nombre de systèmes réels, ils permettent l'obtention d'une solution exacte par la simple résolution d'une équation matricielle.

8.1 Le modèle stochastique

Une exécution d'un système à événements discrets se caractérise par une suite (*a priori* infinie) d'événements $\{e_1, e_2, \ldots\}$ séparés par des intervalles de temps. Seuls les événements peuvent changer l'état du système.
Formellement, le comportement stochastique d'un système concurrent est déterminé par deux familles de variables aléatoires :

– $X_0, \ldots, X_n, \ldots$ à valeurs dans l'espace (discret) des états du système, noté S. Dans la suite, nous supposerons que cet espace est fini. X_0 représente l'état initial du système et X_n $(n > 0)$ l'état courant après le $n^{ième}$ événement. L'occurrence d'un événement ne modifie pas nécessairement l'état du système, par conséquent X_{n+1} peut être égal à X_n ;
– $T_0, \ldots, T_n, \ldots$ à valeurs dans $\mathbb{R}^+$ où T_0 représente l'intervalle de temps avant le premier événement et T_n $(n > 0)$ représente l'intervalle de temps entre le $n^{ième}$ et le $(n+1)^{ième}$ événement. Notons que cet intervalle peut être nul (par

exemple une suite d'instructions considérées comme instantanées au regard de transactions de base de données avec des entrées/sorties).

Lorsque la distribution initiale X_0 est concentrée en un état s, on dira que le processus démarre en s (c'est-à-dire $Pr(X_0 = s) = 1$).

A priori, aucune restriction n'est imposée sur ces familles de variables aléatoires. Cependant, pour les catégories de processus que nous étudierons, un système ne peut exécuter une infinité d'actions en un temps fini.

8.1.1 Les chaînes de Markov à temps discret

Une chaîne de Markov à temps discret (DTMC pour *Discrete Time Markov Chain*) est un processus stochastique qui possède les caractéristiques suivantes :

– l'intervalle de temps entre les instants T_n est une constante de valeur 1 ;
– l'état suivant un état atteint ne dépend que de cet état et les probabilités de transition restent constantes[1] au cours du temps :

$$Pr(X_{n+1} = s_j \mid X_0 = s_{i_0}, \ldots, X_n = s_i) =$$
$$Pr(X_{n+1} = s_j \mid X_n = s_i) = p_{ij} =_{def} P[i, j].$$

Ainsi, on définit une chaîne de Markov homogène $\mathcal{C}$ par un espace d'états S, une matrice P représentant les probabilités de transition entre les états, *i.e.*, $P[i, j] = p_{ij}$ et π_0 une distribution initiale de probabilité sur ces états. On utilisera la notation $\mathcal{C} = \langle S, P, \pi_0 \rangle$.

8.1.1.1 Comportement transitoire et stationnaire d'une DTMC

Pour présenter les résultats les plus importants concernant l'analyse numérique d'une DTMC, nous rappelons plusieurs définitions concernant les propriétés structurelles d'une DTMC $\mathcal{C} = \langle S, P, \pi_0 \rangle$.

Définition 8.1 (Irréductibilité)**.** *$\mathcal{C}$ est irréductible ssi de tout état i on peut atteindre tout état j (en un nombre fini d'étapes) :*

$$\forall i, j \in S, \exists n \text{ tel que } p_{ij}^{(n)} > 0$$

Où $p_{ij}^{(n)} = \sum_{k \in S} p_{ik}^{(n-1)} p_{kj}$ et $p_{ij}^{(1)} = p_{ij}$.

[1] D'où le terme de chaîne *homogène* utilisé dans les études sur les chaînes de Markov en toute généralité.

Définition 8.2 (Périodicité)**.** *Un état i est périodique si on ne peut y revenir qu'après un nombre d'étapes multiple de $k > 1$:*

$$\exists k > 1 \text{ tel que } p_{ii}^{(n)} = 0 \text{ pour n non multiple de } k.$$

La période de l'état j est alors le plus petit entier k vérifiant cette propriété.

Définition 8.3 (Récurrence)**.** *Soit $f_{ii}^{(n)}$ la probabilité que le premier retour en i ait lieu n étapes après l'avoir quitté. Soit f_{ii}, la probabilité de revenir en i après l'avoir quitté : $f_{ii} = \sum_{n=1}^{\infty} f_{ii}^{(n)}$. Soit M_i, le nombre moyen d'étapes de retour en i : $M_i = \sum_{n=1}^{\infty} n\, f_{ii}^{(n)}$.*

Un état i est dit :
- *transitoire si $f_{ii} < 1$.*
- *récurrent si $f_{ii} = 1$. De plus il est*
  - *récurrent nul si $M_i = \infty$.*
  - *récurrent non nul si $M_i < \infty$.*

Définition 8.4 (Ergodicité)**.** *Un état est ergodique s'il est apériodique et récurrent non nul. $\mathcal{C}$ est ergodique si tous ses états sont ergodiques.*

Un des principaux objectifs de l'étude d'une DTMC, concerne l'évaluation de la probabilité de l'occurrence d'un état i de la chaîne à une certaine étape n : $\pi_i(n) = Pr(X_n = i)$ (*analyse du régime transitoire*). On s'intéresse aussi souvent à l'existence et l'évaluation du *régime permanent* de la DTMC. Ce régime est caractérisé par une distribution de probabilité sur les états telle qu'une fois cette distribution atteinte, elle demeure la distribution du processus pour toutes les étapes suivantes.

Formellement, il s'agit de répondre aux deux questions suivantes :

1. $\lim_{n\to\infty} \pi(n)$ existe-t-elle ?
2. Si oui, alors comment la calculer ?

La condition nécessaire et suffisante pour garantir l'existence de la précédente limite est que la DTMC soit *ergodique*. Dans ce cas, le vecteur π des probabilités des états en régime permanent est obtenu par la résolution du système d'équations suivant :

$$\begin{cases} \pi.P = \pi \\ \sum_{i \in S} \pi_i = 1 \end{cases} \tag{8.1}$$

1ex2ex Une fois la distribution stationnaire calculée, plusieurs indices de performances peuvent être obtenus. Par exemple,
– Le temps[2] moyen de récurrence de l'état i.
– Le temps moyen passé par le processus dans un état i.
– Le temps moyen passé par le processus dans un état i entre deux visites successives à l'état j.

8.1.2 Les processus semi-markoviens

Nous nous restreindrons ici à une notion de processus semi-markovien moins générale que la définition habituelle, car elle est largement suffisante pour les système que nous considérons.

Un processus semi-markovien (SMP, pour Semi-Markovian Processes) est un processus stochastique qui possède les caractéristique suivante :
– l'intervalle de temps entre les instants T_n est une variable aléatoire qui ne dépend que de l'état X_n. Autrement dit :

$$Pr(T_n \leq \tau \mid X_0 = s_{i_0}, \ldots, X_n = s_i, T_0 \leq \tau_0, ..., T_{n-1} \leq \tau_{n-1}) = Pr(T_n \leq \tau \mid X_n = s_i) = Pr(D_i \leq \tau);$$

où D_i est une variable aléatoire quelconque, de distribution d'espérance finie, notée d_i.
– l'état suivant un état atteint ne dépend que de cet état et les probabilités de transition restent constantes au cours du temps :

$$Pr(X_{n+1} = s_j \mid X_0 = s_{i_0}, \ldots, X_n = s_i) = Pr(X_{n+1} = s_j \mid X_n = s_i) = p_{ij} =_{def} P[i, j].$$

On remarque ici que les changements d'états X_n constituent une DTMC. Cette chaîne est appelée *la chaîne de Markov incluse (EMC, pour Embedded Markov Chain).*

8.1.2.1 Conditions d'existence et calcul d'une distribution stationnaire

On se limitera ici à spécifier une condition suffisante d'existence d'une distribution stationnaire qui couvre les cas les plus fréquents de stationnarité. Nous commençons par supposer que la chaîne de incluse est ergodique avec une distribution

[2]Ici, le temps est mesuré en nombre d'étapes.

$pi'(\pi'_k = \pi'[k])$ solution de l'équation 8.1.

L'hypothèse d'ergodicité assure que chaque état s_i se répète indéfiniment. Cependant chaque visite à s_i donne lieu à un séjour d'une moyenne d_i. Il se peut alors que, bien que le nombre moyen de visite avant un retour soit fini, la moyenne du temps de retour soit infinie : le nombre moyen de visites à s_k entre deux visites à s_i vaut π'_k/π'_i. Par conséquent, le temps moyen de retour en s_i est égale à :

$$d_i + \sum_{k \neq i} d_k . \frac{\pi'_k}{\pi'_i} = \frac{1}{\pi'_i} . \sum_k d_k . \pi'_k$$

Autrement dit, l'existence d'une distribution stationnaire est assurée si $\sum_k d_k . \pi'_k$ est finie.

Avec le même raisonnement, on voit que le ratio π_k/π_i doit correspondre au temps de séjour moyen en s_k entre deux retours en s_i, divisé par le temps de séjour moyen en s_i, soit :

$$\frac{\pi_k}{\pi_i} = \frac{d_k . \frac{\pi'_k}{\pi'_i}}{d_i} = \frac{d_k . \pi'_k}{d_i . \pi'_i}$$

Ceci conduit à la distribution stationnaire.

$$\pi_k = \frac{\pi'_k . d_k}{\sum_{k'} \pi'_{k'} . d_{k'}}$$

Notons que cette manière de procéder autorise certaines distributions D_i à être concentrées en 0.

Puisque l'analyse d'un processus semi-markovien se ramène à l'analyse de la chaîne incluse (puis la reintroduction du temps par biais du temps de séjour moyen) alors, toutes les discussions qui vont suivre portent uniquement sur les chaînes de Markov à temps discret.

8.2 L'agrégation markovienne

Lorsque l'espace des états de la chaîne de Markov est trop important, on est alors confronté au problème de *l'explosion combinatoire* et par conséquent la complexité de la résolution numérique devient très élevée. Dans ce cas, on fait appel à des méthodes d'analyse efficaces pour traiter des espaces d'états dont la taille peut atteindre plusieurs millions d'états. Parmi ces méthodes, l'agrégation markovienne vise à diminuer la taille du processus en regroupant les états en classes.

Cependant, pour que le processus agrégé puisse être analysé, ce regroupement doit préserver la propriété markovienne du processus : la probabilité de passer d'une classe à une autre classe d'états ne doit pas dépendre de l'état, mais uniquement de la classe dans laquelle on se trouve.

Définition 8.5 (Agrégation markovienne)**.** *Soit $\mathcal{C} = \langle S, P, \pi_0 \rangle$ une DTMC associée au processus stochastique $\{X_n\}_{n\in\mathbb{N}}$ et $\{S_i\}_{i\in I}$ une partition de S. Soit Y_n une variable aléatoire définie par $Y_n = i \Leftrightarrow X_n \in S_i$ alors :*

1. *U est fortement agrégée p.r.a $\{S_i\}_{i\in I}$ ssi $\forall \pi_0, \{Y_n\}_{n\in N}$ est une DTMC.*
2. *U est faiblement agrégée p.r.a $\{S_i\}_{i\in I}$ ssi $\exists \pi_0, \{Y_n\}_{n\in N}$ est une DTMC.*

Agrégation markovienne forte. Une condition *d'agrégation forte* a été définie dans [KS60], assurant que le processus agrégé est markovien quelle que soit la distribution des probabilités initiales.

Théorème 8.1 (Condition d'agrégation forte)**.** *Soit $\mathcal{C} = \langle S, P, \pi_0 \rangle$ une DTMC et $\{S_i\}_{i\in I}$ une partition de S, alors U est fortement agrégeable p.r.a. $\{S_i\}_{i\in I}$ ssi*

$$\forall i \neq j \in I, \forall s, s' \in S_i, \sum_{s''\in S_j} P(s, s'') = \sum_{s''\in S_j} P(s', s'')$$

La matrice de transition de la chaîne agrégée peut être déduite directement de la matrice originale, suivant la proposition ci-dessous.

Proposition 8.1. *Soit $\mathcal{C} = \langle S, P, \pi_0 \rangle$ une DTMC fortement agrégeable par rapport à la partition $\{S_i\}_{i\in I}$. La matrice de probabilité de transitions $\widehat{U}$, associée au processus agrégé est définie par :*

$$\forall i, j \in I, \forall s \in S_i, \widehat{P}(i, j) = \sum_{s'\in S_j} U(s, s')$$

Il est à noter que les probabilités des états de la chaîne originale ne peuvent pas être obtenues à partir de celles des classes d'états, dans la chaîne agrégée. L'utilisateur doit donc se contenter de l'information contenue dans les probabilités des classes, qui n'est significative que si la sémantique des classes est elle-même significative.

Agrégation markovienne exacte. Si la caractérisation de l'agrégation forte est assez simple, celle de l'agrégation faible est beaucoup plus difficile à trouver et à vérifier. Cependant, un cas particulier de cette agrégation est relativement simple : *l'agrégation exacte* [Sch84].

Théorème 8.2 (Condition d'agrégation exacte)**.** *Soit $\mathcal{C} = \langle S, P, \pi_0 \rangle$ une DTMC et $\{S_i\}_{i \in I}$ une partition de S, alors U est exactement agrégeable p.r.a. $\{S_i\}_{i \in I}$ ssi*

$$\forall i, j \in I, \forall s, s' \in S_i, \sum_{s'' \in S_j} P(s'', s) = \sum_{s'' \in S_j} P(s'', s')$$

En outre, l'agrégation exacte possède des propriétés importantes : comme pour l'agrégation forte, la matrice de transition de la chaîne agrégée est calculée directement à partir de la matrice de la chaîne originale ; ayant une distribution initiale équiprobable sur chaque sous-ensemble de la partition, la distribution à n'importe quel instant est aussi équiprobable ; si la DTMC est ergodique, sa distribution stationnaire est équiprobable. En d'autres termes, en connaissant la matrice de transition de la chaîne agrégée, on peut calculer sa distribution stationnaire, et déduire (par équiprobabilité) la distribution stationnaire de la chaîne originale. La proposition suivante résume ces résultats [Sch84].

Proposition 8.2. *Soit $\mathcal{C} = \langle S, P, \pi_0 \rangle$ une DTMC, exactement agrégeable par rapport à la partition $\{S_i\}_{i \in I}$. Soit $\widehat{U}$ la matrice de transition associée au processus agrégé, alors :*

1. $\forall i, j \in I, \forall s \in S_j, \widehat{P}(i, j) = \frac{|S_j|}{|S_i|} \times (\sum_{s' \in S_i} P(s', s))$.
2. *Si $\forall i \in I, \forall s, s' \in S_i, \pi_0(s) = \pi_0(s')$ alors $\forall n, \forall i \in I, \forall s, s' \in S_i, \pi_n(s) = \pi_n(s')$, où π_n est la distribution de probabilité à l'étape n.*
3. *Si $\mathcal{C}$ est ergodique et π est la distribution stationnaire, alors*
$$\forall i \in I, \forall s, s' \in S_i, \pi(s) = \pi(s')$$.

Nous constatons que l'agrégation markovienne soulève deux problèmes : d'une part elle fournit des conditions pour vérifier que la partition des états en classe conduit à un processus markovien, mais elle ne fournit pas de méthodes pour construire cette partition. D'autre part, l'information obtenue n'est exploitable que si les classes d'états ont une signification vis-à-vis du comportement du système étudié.

Il faut donc, pour une utilisation efficace de la méthode, que le formalisme employé pour décrire la chaîne de Markov permette de résoudre, au moins partiellement, ces problèmes. La section suivante étudie ces problèmes dans le cadre des

réseaux stochastiques bien formés.

8.3 Application au réseaux stochastiques bien formés

Le modèle des réseaux de Petri a été introduit afin de modéliser et valider des systèmes synchronisés. Cependant,la validation du comportement du système inclut une étape d'évaluation de performances. Par exemple, l'étude d'un protocoles de communication passe obligatoirement par une étude des performances pour mesurer le sur-coût temporel dû au protocol. L'introduction du temps dans les réseaux de Petri est donc apparue nécessaire pour compléter les informations fournies par l'étude du modèle qualitatif.

Parmi les extensions temporisées des réseaux de Petri, les réseaux de Petri stochastiques [FN85], obtenus en associant aux transitions un temps de franchissement distribué suivant une loi exponentielle, présentent la propriété essentielle suivante : *le graphe de marquages accessibles d'un réseau de Petri stochastique est isomorphe à une chaîne de Markov [Mol81]*. Par conséquent, les résultats connus sur les chaînes de Markov sont réutilisables dans ce contexte.

Cependant, la complexité des système étudiés s'est rapidement trouvée limitée par le pouvoir de représentation des réseaux de Petri stochastiques. Les réseaux colorés se sont alors présentés comme un formalisme permettant d'élargir la classe des systèmes que l'on savait représenter de manière concise.

Les méthodes d'analyse des réseaux de Petri stochastiques colorés se sont donc elle aussi orientées vers une utilisation des symétries du modèle pour construire des classes d'états. Cependant, ces classes ne présentent d'intérêt que si elles simplifient effectivement la résolution du processus, et donc en particulier si elles préservent son caractère markovien. L'agrégation markovienne est donc rapidement devenue comme une technique de base pour l'analyse des réseaux stochastiques colorés.

Plusieurs méthodes ont été développées sur les réseaux colorés pour créer des processus markoviens agrégés ([Zin87, GC88, CL88]). Comme dans l'analyse qualitative, le regroupement des états se fait en général sur la base d'une relation d'équivalence *définie par l'utilisateur* et qui dépend des fonctions de couleurs figurant dans le modèle. Cependant, compte-tenu de la généralité des classes de réseaux colorés étudiés, la définition de cette relation est parfois difficile à obtenir, ce qui conduit généralement à des erreurs nécessitant la reprise de toute la phase

d'agrégation.

Grâce à ses fonctions de couleurs structurées, le modèle des *réseaux bien formés stochastiques* a permis de surmonter ce problème. Par l'introduction du temps *qui respecte les symétries qualitatives* du système, le graphe de marquages symboliques (voir section 6.3) est isomorphe à une chaîne de Markov agrégée [DH89] : les classes d'états et arcs sont construits de manière à respecter les symétries du système et à vérifier les conditions d'agrégation forte et exacte simultanément.

8.3.1 Les réseaux de Petri stochastiques bien formés

Un réseau stochastique bien formé (SWN, pour Stochastic WN) est un réseau bien formé dont les transitions possèdent un attribut supplémentaire qui correspond à un délai de franchissement. Le délai de franchissement associé à une transition peut être soit distribué suivant une loi exponentielle, et la transition est dite *temporisée*, soit nul, et la transition est dite *immédiate*. Lorsque deux transitions temporisées se trouvent simultanément franchissables, on sélectionne pour chacune d'elles un délai de franchissement suivant la distribution associée, et celle qui obtient la plus petite valeur est franchie la première (politique de course). En revanche, lorsque deux transitions immédiates sont simultanément franchissables, la connaissance d'une information supplémentaire est nécessaire pour spécifier complètement le comportement du système. Cette information est matérialisée par l'association d'un poids à chaque transition immédiate. La probabilité de franchir une transition est alors proportionnelle au poids de la transition.

Cependant, la concurrence se produit non seulement entre les transitions mais aussi entre les instances de couleurs d'une même transition. Une solution possible, permettant de respecter les symétries du modèle qualitatif sous-jacent et les apports de sa technique de réduction, consiste à associer à chaque transition une valeur qui ne dépend pas des couleurs mais des sous-classes statiques auxquelles elles appartiennent. De cette manière, les symétries détectées au niveau du modèle qualitatif s'étendent naturellement au modèle quantitatif.

Nous commençons par définir la notion de *partition statique d'un marquage,* qui permet de formaliser les réseaux stochastiques bien formés.

Définition 8.6 (Partition statique d'un marquage)**.** *La partition statique d'un marquage m pour la place p, notée* $\widetilde{m}(p)$ *est un élément de Bag*$(\mathfrak{Part}_{J(p)})$*, défini par :*

$$\widetilde{m}(p)(\tilde{c}) = \sum_{c'/\tilde{c'}=\tilde{c}} m(p)(c')$$

$\tilde{m}(p)(\tilde{c})$ donne le nombre de jetons de $m(p)$ appartenant au produit cartésien de sous-classes statiques $\tilde{c}$.

Définition 8.7 (Un réseau stochastique bien formé (SWN))**.** *Un SWN est un couple (WN, θ) tel que :*
– *WN est réseau bien formé,*
– *θ est un vecteur défini sur T ($\theta(t)$ est noté θ_t), tel que :*

$$\theta_t : \mathfrak{Part}_{J(t)} \times \Pi_{p \in P} Bag(\mathfrak{Part}_{J(p)}) \rightarrow \Re^+$$

où $\theta_t(\tilde{c}, \tilde{m})$ désigne :

1. *si $\pi(t) = 0$ (la transition t est temporisée), le taux de franchissement associé à l'instance (t,c) dans le marquage m. Ce délai de franchissement est une variable aléatoire distribuée suivant une loi exponentielle de moyenne $\frac{1}{\theta_t(\tilde{c},\tilde{m})}$*
2. *si $\pi(t) > 0$ (la transition t est immédiate), le poids associé à l'instance (t,c) dans le marquage m. Cette pondération détermine la probabilité de franchir l'instance (t,c) lorsque d'autres transitions immédiates sont franchissables simultanément à partir de m. Cette probabilité est donnée par :*

$$\frac{\theta_t(\tilde{c}, \tilde{m})}{\sum_{(t',c') \ franchissable\ en\ m} \theta_{t'}(\tilde{c}', \tilde{m})}$$

Le taux de franchissement (ou le poids) associé à une transition ne dépend donc que des sous-classes statiques et jamais des couleurs directement. Ceci est suffisant pour affirmer que l'ensemble des arcs ordinaires représentés par un arc symbolique ont des valuations identiques. Ainsi, nous définissons le taux de franchissement associé à une instance symbolique comme suit.

Définition 8.8 (Taux de franchissement d'une instance symbolique)**.** *Le taux de franchissement de la transition t à partir du marquage symbolique $\hat{m}_{\mathfrak{Symb}}$, pour l'instance symbolique $(t, \hat{c})$ avec $\hat{c} = \Pi_i^n \Pi_j^{e_i} Z_i^{\lambda_i(j), \mu_i(j)}$ est donné par : $\theta_t(\tilde{c}, \tilde{m})$ tel que $\Pi_i^j \Pi_j^{e_i} \mathcal{D}_{i, \mathfrak{Symb}_i . d_i(\lambda(j))} = \tilde{c}$ et $m \in [\hat{m}_{\mathfrak{Symb}}]$.*

Ainsi, la symétrie observée au niveau qualitatif du modèle est préservée au niveau quantitatif. L'agrégation markovienne peut donc être utilisée pour la résolution du processus aléatoire associé au SWN.

8.3.2 SRG d'un SWN et agrégation markovienne

La coexistence dans un SWN de transitions immédiates et temporisées engendre, lors de la construction du RG d'un SWN, deux types de nœuds (marquages) :
– *Les marquages tangibles*, où le système séjourne pendant un temps non nul.
– *Les marquages évanescents*, que le système quitte dès qu'il y entre.

Par conséquent, le processus engendré par un SWN est un processus *semi-markovien*. On doit alors, pour analyser le système, travailler sur l'EMC ($\mathcal{E}$), *isomorphe à son RG*. L'espace d'états S de $\mathcal{E}$ est l'ensemble des marquages accessibles, et la matrice de transition P est définie par :

$$P(m_i,m_j)=\frac{\sum_{(t,c)\ menant\ de\ m_i\ à\ m_j}\theta_t(\tilde{c},\tilde{m}_i)}{\sum_{(t',c')\ franchissable\ en\ m_i}\theta_{t'}(\tilde{c}',\tilde{m}_i)}\ ,\forall\ m_i\neq m_j\in S. \qquad (8.2)$$

Il a été prouvé, dans [Dut91], que $\mathcal{E}$ vérifie les conditions d'agrégation forte et exacte, par rapport à une partition de l'ensemble S en classes d'équivalence. Ces classes d'équivalence sont construites en utilisant la relation induite par le sous-groupe des symétries admissibles $\mathcal{G}$.[3] En d'autre termes, *le SRG du SWN est isomorphe à une chaîne agrégée de* $\mathcal{E}$. La matrice de transition de la chaîne agrégée est définie par :

$$\hat{P}(\hat{m}_i,\hat{m}_j)=\frac{\sum_{(t,\hat{c})\ \text{menant de}\ \hat{m}_i\ \text{à}\ \hat{m}_j}\theta_t(\tilde{c},\tilde{m}_i)\cdot N(\hat{c})}{\sum_{(t',\hat{c}')\ \text{franchissable en}\ \hat{m}_i}\theta_{t'}(\tilde{c}'),\tilde{m}_i)\cdot N(\hat{c}')},\forall\ \hat{m}_i\neq\hat{m}_j\in\hat{S} \qquad (8.3)$$

Où $m_i\in[\hat{m}_i]$ et $N(\hat{c})$ est le nombre d'instances ordinaires représentées par l'instance symbolique $(t,\hat{c})$: considérons que λ et μ sont les fonctions associées à l'instance symbolique $\hat{c}$, alors $N([\lambda,\mu])$ est donné par la formule suivante :

$$N([\lambda,\mu])=\prod_{i=1}^{h}\prod_{j=1}^{|\mathfrak{Part}_i|}\prod_{q=1}^{|\mathcal{D}_{i,j}|}\frac{|Z_{i,j}^q|!}{(|Z_{i,j}^q|-\beta_{i,j}^q)!} \qquad (8.4)$$

Où h est le nombre de classes non ordonnées, $\beta_{i,j}^q=sup(\mu_i(e)|\lambda_i(e)=Z_{i,j}^q)$ est le nombre d'instanciations différentes dans la sous-classe dynamique $Z_{i,j}^q$.

Le gain de la technique SRG est très fortement lié à la présence de symétries globales dans le modèle. Or, en pratique, il arrive souvent que dans certaines situations, les composants d'un système se comportent de manière asymétrique. L'approche SRG ne peut dans ce cas contribuer à diminuer l'espace d'états car

[3] voir section 6.3.1.

les asymétries, même exceptionnelles, éliminent toute possibilité d'exploitation de symétries globales. C'est pour cette raison qu'une extension du SRG à été proposée, l'ESRG (voir section 6.5). Nous allons voir que ce graphe peut aussi servir de base à la construction d'une chaîne agrégée.

8.3.3 ESRG d'un SWN et agrégation markovienne

L'utilisation de l'ESRG pour évaluer les performances du système s'avère délicate, en particulier parce que certaines informations sur la structure du graphe d'état peuvent être masquées.

Entre autres, la structure de l'ESRG ne permet pas instantanément de savoir si le graphe d'états est fortement connexe. L'ESRG ne permet donc pas de décider directement si le processus stochastique sous-jacent est ergodique, ou non. Une technique incrémentale a été présentée dans [CDFI99], consistant à raffiner les informations contenues dans l'ESRG jusqu'à décider l'ergodicité du processus stochastique sous-jacent.

Aussi, l'ESRG n'est en général pas isomorphe à une chaîne de Markov agrégée, mais les auteurs de [CDFI00] ont présenté une technique qui permet de dériver une chaîne fortement agrégée à partir d'un ESRG ergodique par raffinements successifs. Dans le pire des cas, cette chaîne agrégée est isomorphe au SRG.

Algorithme de raffinement de l'ESRG. Après la construction de l'ESRG, les nœuds du graphe sont partitionnés en agrégats. Chaque agrégat contient un RSM plus l'ensemble E' des SM présents dans le graphe tel que : $\forall SM \in E', [SM] \subseteq [RSM]$. On note que ces agrégats ont été construits sur des critères qualitatifs (voir section 6.5). Chaque paire d'agrégats est connectée par des *arcs génériques* et/ou des *arcs instanciés*.

Si deux agrégats sont connectés uniquement par un *arc générique*, alors la condition d'agrégation forte est satisfaite localement par l'agrégat source [CDFI00]. Par contre, s'ils sont connectés par des *arcs instanciés*, il peuvent ne pas satisfaire la condition d'agrégation forte, du faite de l'asymétrie engendrée par certains de ces franchissements. Dans ce cas, il faut diviser l'agrégat source en agrégats plus petits pour satisfaire, localement, la condition d'agrégation.

Cette division n'affecte pas les successeurs de l'agrégat divisé, car les flux sortants de chacun des marquages vers les autres agrégats ne sont pas modifiés. Par contre, les prédécesseurs de cet agrégat sont affectés par la division, car les mar-

quages qu'ils représentent peuvent être connectés à diffrents sous-agrégats issus de l'agrégat divisé. Dans ce cas, la division ce propage vers les prédécesseurs pour maintenir/garantir la satisfaction la condition d'agrégation.

Lors de l'exécution de l'algorithme, les connexions entre agrégats sont classées *vérifiée* ou *à vérifier* selon que la condition d'agrégation est satisfaite localement sur une connexion, c'est-à-dire que le flux sortant représenté par cette connexion est identique pour tous les marquages d'un agrégat, ou non. L'initialisation de l'algorithme consiste donc à marquer *à vérifier* les couples $\langle ag, ag' \rangle$ connectés par un franchissement asymétrique, puisque tous les franchissements symétriques vérifient la condition d'agrégation.

Si la condition d'agrégation n'est pas satisfaite pour un couple $\langle ag, ag' \rangle$, alors ag est divisé en agrégats de grain plus fin. La division d'un agrégat ag peut casser la condition d'agrégation pour les prédécesseurs de ag. L'algorithme marque donc *à vérifier* tous les couples $\langle ag'', ag \rangle$ tel que ag'' est un prédécesseur de ag. Enfin, après la division de ag, toutes les connexions $\langle ag, - \rangle$ sont marquées *vérifiée*. La procédure décrite ci-dessus est itérée jusqu'à ce qu'il n'y ait plus de couple marqué *à vérifier*.

8.4 Conclusion

Dans ce chapitre, nous avons présenté les chaînes de Markov comme modèle pour l'évaluation de performances des systèmes concurrents fini. Ce modèle pose le problème de *l'explosion combinatoire* lorsque l'espace des états de la chaîne de Markov est trop important.

L'une des méthodes qui permettent la résolution de ce problème est l'agrégation markovienne. Elle consiste en un regroupement des états en classes, de manière à dériver une chaîne (agrégée) de taille réduite, mais équivalente à celle de départ.

L'application de la méthode d'agrégation sur le modèle des réseaux stochastique bien formés (SWN) a permit l'études de systèmes dont les comportements sont représentés par des graphes de marquages de tailles importantes. Ceci est réalisé *via* le graphe de marquages symbolique (SRG), car les symétries définies sur le modèle qualitatif (le WN sous-jacent) sont "trivialement" étendues sur le modèle quantitatif.

Cependant, le problème reste posé dans le cas des systèmes globalement asymétriques (modélisé par un SWN), car la chaîne agrégée dérivée de ces systèmes

à une taille prochaine de la chaîne originelle. Une première solution a été proposé par l'utilisation de l'ESRG. Elle consiste à raffiner les informations qu'il stocke jusqu'à l'obtention d'une chaîne agrégée représentant le processus originelle. Deux problèmes se posent avec cette solution : (1) nous avons besoin de deux étapes. La première pour la construction de l'ESRG et la deuxième pour dériver la chaîne agrégée ; (2) sur la chaîne agrégée (dérivée), nous ne pouvons dériver que les probabilités des classes d'états et pas celles des états à l'intérieur des classes. Par conséquent, nous n'avons qu'une étude approximative de processus stochastique sous-jacent.

Dans le prochain chapitre, nous proposons une nouvelle approche pour la modélisation et l'évaluation de performances des systèmes globalement asymétriques. Cette approche est dérivée directement de celle proposée dans la partie qualitative de cette thèse. Son avantage (en plus d'être complètement automatisable) est qu'elle peut engender une chaîne de Markov de taille réduite pour les systèmes globalement asymétriques **en une seule passe**. De plus, les **probabilités des états** sont calculer directement sur la chaîne agrégée.

Chapitre 9

Symétries partielles et agrégation Markovienne[1]

Les SWN ont démontré leur efficacité pour la modélisation et l'analyse quantitative des systèmes concurrents. Cependant, cette efficacité est très dépendante du degré de symétrie (globale) qu'offre le système. En effet, l'existence d'asymétries locales dans le modèle est souvent un obstacle de taille pour le regroupement des états et des franchissements en classes, ce qui réduit considérablement l'intérêt de l'agrégation pour le traitement de ce type de systèmes.

Dans le cadre de la vérification qualitative, nous avons proposé une solution efficace à ce problème (voir chapitre 6). Cette solution repose sur une représentation séparée des asymétries, via l'utilisation des *automates de contrôle*. Le graphe quotient est alors obtenu par une synchronisation des transitions de cet automate avec les transitions du WN (symétrique).

Dans ce chapitre, nous adaptons notre approche qualitative à l'analyse quantitative. Nous montrons que cette approche peut être utilisée pour dériver un graphe quotient isomorphe à une chaîne de Markov agrégée.

9.1 Modèle pour les DTMCs partiellement symétriques

Le modèle des systèmes *partiellement symétriques* que nous présentons ici est défini par une DTMC, obtenue par la synchronisation d'une DTMC *symétrique* avec

[1] Les idées développées dans ce chapitre ont fait l'objet d'une publication [BDSH05].

un automate de contrôle.

La formalisation de cette synchronisation nécessite l'étiquetage des transitions de la DTMC par les événements du système étudié : soit $\mathcal{C} = \langle S, P, \pi_0 \rangle$ une DTMC et $\mathcal{E}$ l'ensemble des événements du système. On associe à chaque transition entre deux états s, s' un événement dans $\mathcal{E}$, noté $\Lambda(s, s')$. Les automates de contrôle que nous considérons sont ceux de la définition 4.2 : $\mathcal{A}_c = \langle L, l_0, \mathcal{E}, \mathcal{R} \rangle$.

Exemple 9.1. *La Fig.9.1 représente une DTMC étiquetée et son automate de contrôle. Les lettres en gras représentent les événements du système :* $\mathcal{E} = \{a, b, c, d\}$*) et* $0 \leq p \leq 1$ *est une probabilité. Les nombres associés aux états représentent la distribution initiale de probabilités.*

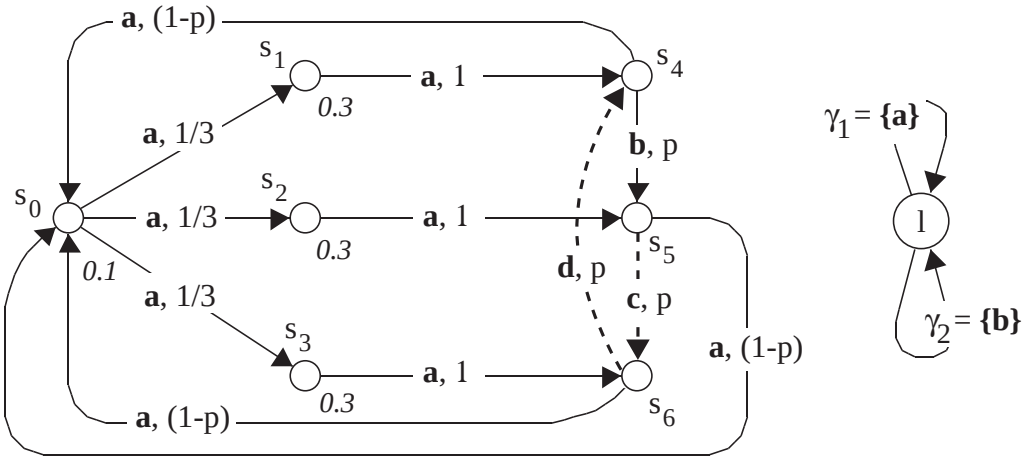


FIG. 9.1 – Une DTMC étiquetée et son automate de contrôle

Dans le produit synchronisé[2] défini ci-dessous, la DTMC représente le composant "actif", et l'automate est utilisé pour "inhiber" les comportements de ce composant en les limitant à ceux réellement produits par le système.

Définition 9.1 (DTMC contrôlée par un automate)**.** *Soit* $\mathcal{C} = \langle S, P, \pi_0 \rangle$ *une DTMC et* $\mathcal{A} = \langle L, l_0, \mathcal{E}, \mathcal{R} \rangle$ *un automate de contrôle. Soit la fonction* $\Delta : S \times S \times L \times L \rightarrow \{0, 1\}$*. Le produit synchronisé de* $\mathcal{C}$ *et* $\mathcal{A}$ *est une DTMC,* $\mathcal{C}_\mathcal{A} = \langle S \times L, P', \pi'_0 \rangle$ *définie par :*
– $\forall s, \pi'_0(s, l_0) = \pi_0(s) \wedge \forall l \neq l_0, \pi'_0(s, l) = 0$

[2]Nous utilisons aussi le terme *contrôle* pour désigner ce produit synchronisé.

– $\forall s,s' \in S, \forall l,l' \in L$,
si $l \xrightarrow{\gamma} l' \wedge \Lambda(s,s') \in \gamma$ *alors* $\Delta(s,s',l,l') = 1$ *sinon* $\Delta(s,s',l,l') = 0$.

$$\textit{On a alors, } P'((s,l),(s',l')) = \frac{\Delta(s,s',l,l') \times P(s,s')}{\sum_{(s'',l'') \in S \times L} \Delta(s,s'',l,l'') \times P(s,s'')}$$

Dans l'exemple 9.1, l'automate de contrôle interdit les transitions qui ne sont pas étiquetées par a ou b. Ainsi, $\mathcal{C}_{\mathcal{A}}$ est obtenu à partir de $\mathcal{C}$ en retirant les arcs en pointillé et en normalisant les valuations des arcs restant.

Du point de vue théorique, la spécification des symétries du système repose sur la théorie des groupes opérant sur les ensembles d'états et d'événements (voir définition 3.1). Ceci permet l'introduction des notions de DTMC symétrique et partiellement symétrique comme suit.

Définition 9.2 (DTMC symétrique et partiellement symétrique)**.** *Une DTMC* $\mathcal{C} = \langle S, P, \pi_0 \rangle$ *est symétrique par rapport à* $\mathcal{G} \subseteq \mathfrak{S}(S \cup \mathcal{E})$ *ssi :* $\forall g \in \mathcal{G}, \forall s \neq s' \in S$,

1. $\pi_0(g.s) = \pi_0(s)$,
2. $P(g.s, g.s') = P(s,s')$ *et*
3. $\Lambda(g.s, g.s') = g.\Lambda(s,s')$.

Soient $\mathcal{C}$ *une DTMC symétrique par rapport à* $\mathcal{G}$ *et* $\mathcal{A}_c$ *un automate de contrôle de* $\mathcal{C}$*, alors* $\mathcal{C}_{\mathcal{A}_c}$ *est dite partiellement symétrique par rapport à* $\mathcal{G}$.

On associe à chaque arc $l \xrightarrow{\gamma} l'$ *de* $\mathcal{A}_c$ *un sous-groupe* $\mathcal{H}_\gamma \subseteq \mathcal{G}$ *qui vérifie :*

$$\forall g \in \mathcal{H}_\gamma, \forall e \in \mathcal{E}, e \in \gamma \Leftrightarrow g.e \in \gamma.$$

La taille du sous-groupe $\mathcal{H}_\gamma$, associé à $l \xrightarrow{\gamma} l'$, est un indicateur de la symétrie de la transition : quand $\mathcal{H}_\gamma = \mathcal{G}$, la transition est dite *totalement symétrique*, tandis que pour $\mathcal{H}_\gamma = \{id\}$, la transition est dite *totalement asymétrique*.

Considérons à nouveau la Fig.9.1, soit $\mathcal{G} = \{id, g, g \bullet g\}$, tel que r est définie par :

$$\begin{array}{llll} g.s_0 = s_0 & g.s_1 = s_2 & g.s_2 = s_3 & g.s_3 = s_1 \\ g.s_4 = s_5 & g.s_5 = s_6 & g.s_6 = s_4 & \\ g.a = a & g.b = c & g.c = d & g.d = b \end{array}$$

Il est facile de vérifier que $\mathcal{G}$ est un groupe et que la DTMC est symétrique par rapport à $\mathcal{G}$. Les sous-groupes associés aux étiquettes de $\mathcal{A}_c$ sont : $\mathcal{H}_{\gamma_1} = \mathcal{G}$ et $\mathcal{H}_{\gamma_2} = \{id\}$.

9.1.1 Agrégation d'une DTMC partiellement symétrique

Ayant une DTMC $\mathcal{C}_{\mathcal{A}_c}$ partiellement symétrique par rapport à $\mathcal{G}$, notre méthode tente de construire une DTMC plus petite (mais équivalente). Cependant, pour prouver la correction de la construction, nous introduisons d'abord une DTMC $\mathcal{C}^G_{\mathcal{A}_c}$ qui est plus grande que $\mathcal{C}_{\mathcal{A}_c}$.

Dans la DTMC $\mathcal{C}^G_{\mathcal{A}_c}$, les états de $\mathcal{C}_{\mathcal{A}_c}$ sont répliqués en instances, et les instances sont organisées en ensembles. Par construction, le même état de l'automate apparaît dans toutes les instances qui appartiennent au même ensemble. Pour chaque sous-ensemble R, on note (s,l,R) l'instance de (s,l) qu'il contient.

Intuitivement, pour toute paire d'états (s,l,R) et (s',l,R) de $\mathcal{C}^G_{\mathcal{A}_c}$, chaque chemin menant à (s,l,R) peut être transformé, par l'application d'une action de $\mathcal{G}$, en un chemin menant à (s',l,R).

Définition 9.3. *Soit $\mathcal{C}_{\mathcal{A}_c} = \langle S \times L, P', \pi'_0 \rangle$ une DTMC partiellement symétrique par rapport à $\mathcal{G}$, alors la DTMC $\mathcal{C}^G_{\mathcal{A}_c} = \langle S'', P'', \pi''_0, \rangle$ est définie récursivement par :*

1. *L'ensemble des états S'' est l'union des triplets définis à partir de $R \subseteq S$ et un état $l \in L$ par $\{(s,l,R) \mid s \in R\}$,*
2. *$\forall s \in S, \forall l \in L, \forall R \subseteq S$,*
   si ($l = l_0 \wedge R$ est une orbite par $\mathcal{G} \wedge s \in R$) alors $\pi''_0(s,l,R) = \pi'_0(s,l_0)$ $(= \pi_0(s))$ sinon $\pi''_0(s,l,R) = 0$,
3. *Les sous-ensembles d'états "initiaux" sont $\{(s,l,R)\}$ tels que R est l'orbite de s par $\mathcal{G} \wedge \pi''_0(s,l,R) > 0$,*
4. *Si $\{(s,l,R)\}$ est un sous-ensemble d'états et $\exists s^* \in R, \exists s'^* \in S, \exists l \xrightarrow{\gamma} l' \wedge \Lambda(s^*, s'^*) \in \gamma$ alors l'ensemble $\{(s',l',R')\}$ avec $R' = (\mathcal{G}_R \cap H_\gamma).s'^*$ est un autre ensemble d'états,*
5. *$\forall g \in \mathcal{G}_R \cap \mathcal{H}_\gamma$, soit $s = g.s^*$ et $s' = g.s'^*$ alors $P''((s,l,R),(s',l',R')) = P'((s,l),(s',l'))$*

Note 9.1. *La construction précédente des sous-ensembles ne dépend pas du choix de s^* et s'^* : prenons $s' \in (\mathcal{G}_R \cap \mathcal{H}_\gamma).s'^*$, ceci implique que $s' = g.s'^*$ avec $g \in \mathcal{G}_R \cap \mathcal{H}_\gamma$. Définissons $s = g.s^*$, alors $s \in R (\supseteq \mathcal{G}_R.s^*)$ et $\Lambda(s,s') \in \gamma$. Ainsi, il est trivial de montrer que $(\mathcal{G}_R \cap \mathcal{H}_\gamma).s' = (\mathcal{G}_R \cap \mathcal{H}_\gamma).s'^*$.*

Exemple 9.2. *La Fig.9.2 décrit la DTMC $\mathcal{C}^G_{\mathcal{A}_c}$ de l'exemple 9.1. Les rectangles pointillés représentent les sous-ensembles d'états. Ces sous-ensembles sont construits suivant le point (1) de la définition 9.3 : $\{(s_0,l_0,S_0)\}$,*

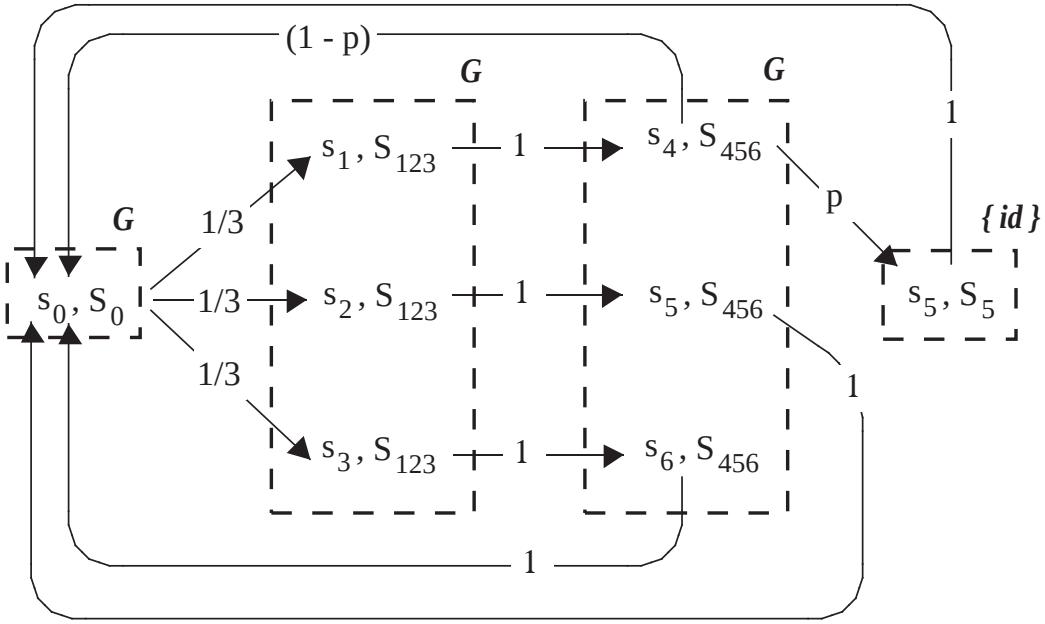


FIG. 9.2 – La DTMC $\mathcal{C}^{G}_{\mathcal{A}_c}$

$\{(s_1,l_0,S_{123}),(s_2,l_0,S_{123}),(s_3,l_0,S_{123})\}$, $\{(s_4,l_0,S_{456}),(s_5,l_0,S_{456}),(s_6,l_0,S_{456})\}$ *et* $\{(s_5,l_0,S_5)\}$. *L'état* l_0 *n'est pas représenté dans la figure. Les points (2) et (3) indiquent que les sous-ensembles initiaux sont ceux associés aux orbites* $S_0=\{s_0\}$ *et* $S_{123}=\{s_1,s_2,s_3\}$.

Un groupe est associé avec chaque sous-ensemble. Il donne une indication sur la manière d'utiliser (4) et (5) pour la construction des sous-ensembles : le groupe est $\mathcal{G}$ *pour les sous-ensembles initiaux et reste* $\mathcal{G}$ *pour les sous-ensembles construits jusqu'à la synchronisation avec* γ_2. *A ce moment, le groupe est réduit à l'identité par l'intersection avec* $\mathcal{H}_{\gamma_2}$, *et le sous-ensemble construit contient un seul état. Par conséquent, il y a deux instances de* s_5 *dans la DTMC résultante, chacune associée à une orbite différente.*

En fait, le processus stochastique attendu est construit en ignorant les instances et en mémorisant les représentants des sous-ensembles.

Définition 9.4 (Processus stochastique associé à $\mathcal{C}^{G}_{\mathcal{A}_c}$)**.** *Soit* $\mathcal{C}_{\mathcal{A}_c}$ *une DTMC partiellement symétrique par rapport à* $\mathcal{G}$. *Le processus stochastique* $(\mathcal{C}^{G}_{\mathcal{A}_c})^{lp}$ *est défini par :* $X^{lp}_n=(l,R)\Leftrightarrow X''_n\in\{(s,l,R)\}$.

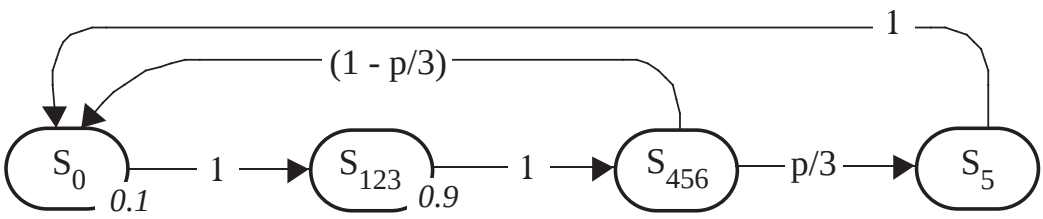


FIG. 9.3 – DTMC $(\mathcal{C}_{\mathcal{A}}^{G})^{lp}$

Le processus résultant pour l'exemple est montré dans la Fig.9.3. La distribution de probabilités initial et les probabilités de transition sont calculées suivant la proposition 8.2.

La proposition suivante constitue le noyau théorique de notre méthode. Elle énonce que $(\mathcal{C}_{\mathcal{A}_c}^{G})^{lp}$ est obtenue à partir de $\mathcal{C}_{\mathcal{A}_c}$ par l'inverse d'une agrégation forte suivie d'une agrégation exacte.

Proposition 9.1. *Soit $\mathcal{C}_{\mathcal{A}_c}$ une DTMC partiellement symétrique par rapport à $\mathcal{G}$, alors :*

- *Soit $(s_0,l_0)\ldots,(s_n,l_n)$ l'espace d'états de $\mathcal{C}_{\mathcal{A}_c}$. $\mathcal{C}_{\mathcal{A}_c}$ est une **agrégation forte** de $\mathcal{C}_{\mathcal{A}_c}^{G}$ par rapport à la partition $\{(s_0,l_0,R)\}_{R\supseteq\{s_0\}},\ldots,\{(s_n,l_n,R)\}_{R\supseteq\{s_n\}}$*
- *Soit $\{(l_0,R_0),\ldots,(l_k,R_k)\}$ l'espace d'états de $(\mathcal{C}_{\mathcal{A}_c}^{G})^{lp}$. $(\mathcal{C}_{\mathcal{A}_c}^{G})^{lp}$ est une **agrégation exacte** de $\mathcal{C}_{\mathcal{A}_c}^{G}$ par rapport à la partition $\{(s,l_0,R_0)\}_{s\in R_0},\ldots,$ $\{(s,l_k,R_k)\}_{s\in R_k}$*

Démonstration. Soit (s,l) un état de $\mathcal{C}_{\mathcal{A}_c}$ et soit (s,l,R) une instance de cet état dans $\mathcal{C}_{\mathcal{A}_c}^{G}$. On montre qu'il y a une application bijective des transitions sortant de (s,l) vers celles sortant de (s,l,R). Grâce à la note 9.1, on suppose que (s,l,R) est examiné pendant la recherche des successeurs de $\{(s'',l,R) \mid s''\in R\}$ dans la Définition 9.3. Alors, $\exists s',\exists l \xrightarrow{\gamma} l'$ t.q. $\Lambda(s,s')\in\gamma \Leftrightarrow \exists R',\exists s'\in R',\exists l\xrightarrow{\gamma} l'$ t.q. $\Lambda(s,s')\in\gamma$ avec $R'=(G_R\cap H_\gamma).s'$. Puisque cette application préserve les probabilités de transitions, alors la condition de la proposition 8.5 pour l'agrégation forte est garantie.

Soient (s_1,l,R) et (s_2,l,R), deux états de $\mathcal{C}_{\mathcal{A}_c}^{G}$. On montre qu'il y a une application bijective entre les transitions entrant dans (s_1,l,R) et dans (s_2,l,R). Soit (v_1,l',R') telle que $\exists l' \xrightarrow{\gamma} l$ et $\Lambda(v_1,s_1)\in\gamma$. Grâce à la même note ci-dessus, $\exists g\in(\mathcal{G}_{R'}\cap H_\gamma)$ t.q. $s_2=g.s_1$. On définit $v_2=g.v_1$, alors $v_2\in R'$ et $\Lambda(v_2,s_2)\in\gamma$. Ceci implique l'existence de l'application cherchée. Puisque cette application préserve les pro-

babilités de transitions, la condition de la propositions 8.5 pour l'agrégation exacte est garantie. □

Nous pouvons à présent décrire notre méthode générique. D'abord, nous supposons que la DTMC $\mathcal{C}_{\mathcal{A}_c}$ associée à un modèle de haut niveau $\mathcal{M}$ est partiellement symétrique. nous supposons aussi que nous sommes capables de calculer directement $(\mathcal{C}^G_{\mathcal{A}_c})^{lp}$ à partir de $\mathcal{M}$. Notons par π_n la distribution de $\mathcal{C}_{\mathcal{A}_c}$ à l'étape n et $\pi_n^{(lp)}$ la distribution calculée de $(\mathcal{C}^G_{\mathcal{A}_c})^{lp}$ à l'étape n. Alors, grâce à la proposition 8.2, nous avons

$$\pi_n(s,l) = \sum_{s \in R} (1/|R|) \times \pi_n^{(lp)}(l,R).$$

Cette égalité reste valable aussi pour la distribution stationnaire. La section suivante va montrer que les assertions précédentes sont satisfaites dans le modèle des SWN.

Bien que théoriquement difficile, nous pouvons donner des indications sur la décroissance de la complexité en espace en utilisant notre approche. Dans la DTMC agrégée, les états originaux ont été substitués par des sous-ensembles. Notez que ces sous-ensembles peuvent avoir des intersections non vides. Cependant ces sous-ensembles sont toujours les orbites d'un état par un sous-groupe de $\mathcal{G}$. Donc, plus ces sous-groupes sont larges, la meilleure est la méthode. Notez qu'à chaque nouvelle construction d'un sous-ensemble, le groupe est réduit (par intersection avec $\mathcal{H}_\gamma$), puis il est agrandi en substituant implicitement à $\mathcal{G} \cap \mathcal{H}_\gamma$ le sous-groupe stabilisateur du sous-ensemble. Interpréter ce phénomène au niveau du modèle revient à dire que le facteur de réduction de la complexité est important chaque fois que l'effet d'une asymétrie disparaît à court terme.

9.2 Application aux réseau de Petri stochastiques bien formés

L'approche que nous allons développer ici (ré)utilise et étend les SWN dans le but d'automatiser la construction de la DTMC $(\mathcal{C}^G_{\mathcal{A}_c})^{lp}$. On appelle *graphe de marquage symbolique dynamique (DSRG)*, la structure symbolique qui représente cette DTMC. Elle est basée sur une représentation symbolique de chaque nœud (l,R) et une règle de franchissement symbolique appliquée directement sur cette représentation.

9.2.1 Le modèle des SWN contrôlés

Dans notre approche, le comportement symétrique du système est capturé par un SWN (symétrique) et les asymétries sont représentées par un automate de contrôle $\mathcal{A}_c$. Nous appellerons ce nouveau modèle : *le modèle des SWN contrôlés (CSWN).*

La définition des automates de contrôle est identique à celle introduite dans la partie qualitative de cette thèse, c'est à dire, $\mathcal{A}_c = \langle L, l_0, \mathcal{E}, \mathcal{R} \rangle$ avec $\mathcal{E} = \{(t,c) | t \in T \ \wedge \ c \in Cd(t)\}$.

Exemple 9.3. *La Fig.9.4 présente un exemple de CSWN.*

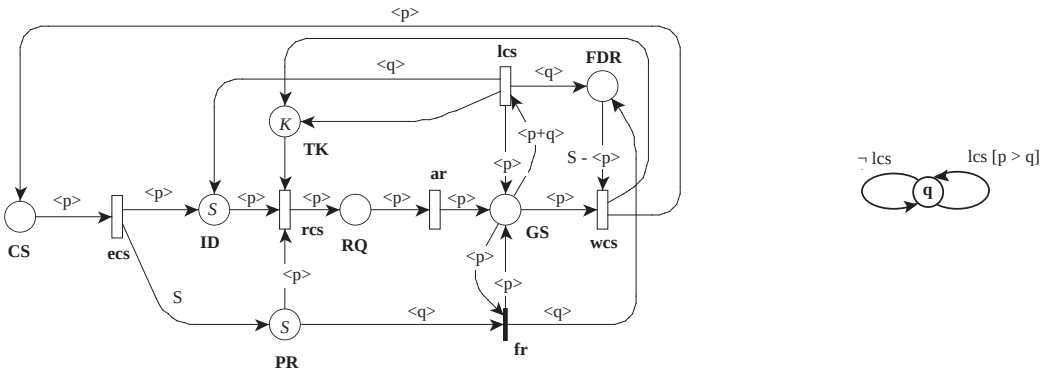


FIG. 9.4 – Modèle CSWN de l'algorithme distribué d'accès à la section critique

Ce modèle représente une version modifiée de l'algorithme d'accès à la section critique, déjà introduit dans la Fig.6.5. La modification consiste en un contrôle du nombre de processus candidats simultanés pour l'accès (modélisé par le nombre K de jetons dans la place TK).

L'automate de contrôle gère les accès à la section critique suivant l'identité des processus. Les étiquettes sur l'automate de contrôle sont définies par :
– $lcs[p > q] = \{(lcs, (p_i, p_j)) | (p_i, p_j) \in Cd(lcs) \wedge j > i\}$;
– $\neg lcs = \{(t,c) | t \neq lcs \wedge c \in Cd(t)\}$;

La sémantique associée à ce modèle est celle d'un chaîne de Markov partiellement symétrique $\mathcal{C}_{\mathcal{A}_c}$: elle obtenue par la synchronisation du RG (ismorphe à une DTMC symétrique $\mathcal{C}$) du SWN avec l'automate de contrôle $\mathcal{A}_c$.

9.3 Le graphe de marquages symbolique dynamique (DSRG)

Les représentations symboliques que nous allons utiliser pour la construction du *DSRG* sont extraites directement de celles déjà définies pour la construction de la structure SSP_{Acc}, dans la partie qualitative de cette thèse (section 7.1). On rappelle que chaque nœud de SSP_{Acc} est un triplet $\langle \mathfrak{Part}^l, \hat{m}_{\mathfrak{Symb}}, l \rangle$.

Dans ce cas, chaque état (l,R) de $(\mathcal{C}^G_{\mathcal{A}_c})^{lp}$ est représenté par un triplet $\langle \mathfrak{Part}^l, \hat{m}_{\mathfrak{Symb}}, l \rangle$ tel que $R = [\langle \mathfrak{Part}^l, \hat{m}_{\mathfrak{Symb}} \rangle]$ et appelé *état symbolique*.

Exemple 9.4. *L'état symbolique* $\langle \mathfrak{Part}^l, \hat{m}_{\mathfrak{Symb}}, l \rangle$ *défini par :*
– $\mathfrak{Part}^l = \{\mathfrak{Part}_1\} = \{\{\mathcal{L}_{1,1}, \mathcal{L}_{1,2}\}\}$ *telle que* $\mathcal{L}_{1,1} = \{pr_1, pr_2\}$ *et* $\mathcal{L}_{1,2} = \{pr_3\}$,
– $\mathfrak{Symb} = \{Sym_1\}$ *définie par* $|Z^1_{1,1}| = |Z^2_{1,1}| = |Z^3_{1,2}| = 1$,
– $\hat{m} = ID(Z^1_1) + RQ(Z^2_1 + Z^3_1)$,
– *et l'état de l'automate de contrôle l,*
représente le nœud (l,R) *tel que* $R = \{m_1, m_2\}$ *avec* $m_1 = ID(pr_1) + RQ(pr_2 + pr_3)$ *et* $m_2 = ID(pr_2) + RQ(pr_1 + pr_3)$.

9.3.1 Franchissement symbolique synchronisé et calcul des probabilités de transitions de la chaîne agrégée

La construction d'une structure symbolique nécessite la définition d'une règle de franchissement symbolique qui répond à deux exigences : d'une part, elle doit être applicable directement sur les états symboliques pour produire d'autres états symboliques. D'autre part, elle doit assurer la satisfaction de la condition d'agrégation exacte pour tout nouvel état symbolique construit.

Franchissent symbolique synchronisé. Le franchissement symbolique que nous avons introduit dans la section 7.3.3 répond à la première exigence. Dans sa version la plus générale, ce franchissement est opéré comme suit : soit un état symbolique $\langle \mathfrak{Part}^l, \hat{m}_{\mathfrak{Symb}} \rangle$ et une transition $l \xrightarrow{\gamma} l'$, de l'automate de contrôle. D'abord, nous construisons l'ensemble RafSymb$(\langle \mathfrak{Part}^l, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^l \cap \mathfrak{Part}^\gamma)$, puis pour chaque élément de cet ensemble, nous calculons ses instances de franchissements symboliques. Les successeurs valides sont alors obtenus par le franchissement des instances qui *appartiennent* à $\langle \mathfrak{Part}^l \cap \mathfrak{Part}^\gamma, \hat{\gamma} \rangle$.

Cependant, cette démarche n'est pas suffisante pour garantir une satisfaction (locale) de la condition d'agrégation exacte entre $\langle \mathfrak{Part}^l, \hat{m}_{\mathfrak{Symb}} \rangle$ et ses successeurs.

En effet, par ce procédé nous garantissons que tous les marquages représentés par un élément $\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}'_{\mathfrak{Symb}'} \rangle \in \text{RafSymb}(\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma})$ ont des franchissements symétriques par rapport à $l \xrightarrow{\gamma} l'$, mais n'assure pas que ces mêmes marquages ont des franchissements symétriques par rapport à n'importe quelle transition $l \xrightarrow{\gamma'} l''$ ($l' \neq l''$). Or, ce dernier point est nécessaire pour établir que les marquages du successeur symbolique $\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}''_{\mathfrak{Symb}''} \rangle$ de $\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ ont tous les mêmes probabilités en entrée (à partir des marquages de $\langle \mathfrak{Part}^{\iota} \cap \mathfrak{Part}^{\gamma}, \hat{m}'_{\mathfrak{Symb}'} \rangle$).

Pour palier à ce problème, un solution évidente consiste à s'assurer que tous les états représentés par un état symbolique, issu d'un raffinement de $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l \rangle$, permettent les mêmes franchissements (à une permutation près) par rapport à l'ensemble $l \xrightarrow{\gamma_i} l_i$. Ceci est possible en optant pour un raffinement de $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l \rangle$ qui prend en compte les asymétries de l'ensemble des transitions $l \xrightarrow{\gamma_i} l_i$ en une seule fois : nous devons donc construire l'ensemble $\text{RafSymb}(\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}} \rangle, \bigcap_i \mathfrak{Part}^{\gamma_i} \cap \mathfrak{Part}^{\iota})$. Ainsi, si $\langle \bigcap_i \mathfrak{Part}^{\gamma_i} \cap \mathfrak{Part}^{\iota}, \hat{m}'_{\mathfrak{Symb}'} \rangle$ est un successeur de $\langle \bigcap_i \mathfrak{Part}^{\gamma_i} \cap \mathfrak{Part}^{\iota}, \hat{m}^k_{\mathfrak{Symb}^k} \rangle \in \text{RafSymb}(\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}} \rangle, \bigcap_i \mathfrak{Part}^{\gamma_i} \cap \mathfrak{Part}^{\iota})$, alors $\langle \bigcap_i \mathfrak{Part}^{\gamma_i} \cap \mathfrak{Part}^{\iota}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle$ est un successeur de $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l \rangle$ qui satisfait la condition d'agrégation exacte (localement).

Calcul des probabilités de transitions. Posons $\mathfrak{Part}^{\iota^C} = \bigcap_i \mathfrak{Part}^{\gamma_i} \cap \mathfrak{Part}^{\iota}$. D'après la première équation de la proposition 8.2, la probabilité de transition entre deux états symboliques, $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l \rangle$ et $\langle \mathfrak{Part}^{\iota^C}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle$ dépend de leurs cardinalités, et de la probabilité de transition vers n'importe quel état $(m', l') \in [\langle \mathfrak{Part}^{\iota^C}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle]$ à partir des états représentés par $\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l \rangle$: $\sum_{(m,l) \in [\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l \rangle]} P'((m,l),(m',l'))$.

Considérons la contribution du franchissement d'une transition t à la somme $\sum_{(m,l) \in [\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}}, l \rangle]} P'((m,l),(m',l'))$. Après l'étape du raffinement symbolique, soit une instance symbolique est possible pour tous les marquages qui appartiennent à la même représentation symbolique, issue du raffinement, soit aucune n'est possible (vis-à-vis de la synchronisation avec l'ensemble $l \xrightarrow{\gamma_i} l_i$). Supposons qu'une telle instantiation est possible à partir d'un état symbolique $\langle \mathfrak{Part}^{\iota^C}, \hat{m}^k_{\mathfrak{Symb}^k} \rangle \in \text{RafSymb}(\langle \mathfrak{Part}^{\iota}, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^{\iota^C})$ et mène à l'état $\langle \mathfrak{Part}^{\iota^C}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle$.

Le taux de sortie global de $\langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle$, causé par le franchissement de l'instance symbolique $(t, \hat{c})$, est $\theta_t.N(\hat{c}).|[\langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}^k_{\mathfrak{Symb}^k} \rangle]|$, où $N(\hat{c})$ est le nombre d'instances ordinaires représenté par l'instance symbolique de $(t, \hat{c})$ et θ_t est le taux associé à t.[3]

Puisque tous les états représentés par $\langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle$ ont le même taux en entrée [4], alors nous pouvons écrire : $\forall (m', l') \in [\langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle]$

$$\sum_{(m,l) \in [\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l \rangle]} P((m,l),(m',l')) = \frac{1}{|[\langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}'_{\mathfrak{Symb}'} \rangle]|} \times$$

$$\sum_{\langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}^k_{\mathfrak{Symb}^k} \rangle \in \mathrm{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^{\mathfrak{l}^C})} \frac{\sum_{(t,\hat{c}) \text{ menant de } \langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}^k_{\mathfrak{Symb}^k} \rangle \text{ à } \langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}'_{\mathfrak{Symb}'} \rangle} \theta_t \cdot N(\hat{c}) \cdot |[\langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}^k_{\mathfrak{Symb}^k} \rangle]|}{\sum_{(t',\hat{c}') \text{ franchissable en } \langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}^k_{\mathfrak{Symb}^k} \rangle} \theta_{t'} \cdot N(\hat{c}')}$$

Ainsi, la probabilité de transition entre $\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l \rangle$ et $\langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle$ est donnée par la formule :

$$\hat{P}(\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}}, l \rangle, \langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}'_{\mathfrak{Symb}'}, l' \rangle) = \frac{1}{|[\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle]|} \times$$

$$\sum_{\langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}^k_{\mathfrak{Symb}^k} \rangle \in \mathrm{RafSymb}(\langle \mathfrak{Part}^{\mathfrak{l}}, \hat{m}_{\mathfrak{Symb}} \rangle, \mathfrak{Part}^{\mathfrak{l}^C})} \frac{\sum_{(t,\hat{c}) \text{ menant de } \langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}^k_{\mathfrak{Symb}^k} \rangle \text{ à } \langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}'_{\mathfrak{Symb}'} \rangle} \theta_t \cdot N(\hat{c}) \cdot |[\langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}^k_{\mathfrak{Symb}^k} \rangle]|}{\sum_{(t',\hat{c}') \text{ franchissable en } \langle \mathfrak{Part}^{\mathfrak{l}^C}, \hat{m}^k_{\mathfrak{Symb}^k} \rangle} \theta_{t'} \cdot N(\hat{c}')}$$

[3] Pour simplifier, nous considérons que θ_t est constante pour toutes les instances symboliques de t.
[4] Propriété du franchissement symbolique des SWN [CDFH93].

9.3.2 Optimisation pour les CSWN

Dans cette section, nous montrons que le formalisme des CSWN permet l'optimisation de la méthode générique. Une première optimisation consiste en un regroupement des représentations symboliques obtenues après un franchissement symbolique synchronisé, tout en préservant la condition de l'agrégation exacte. Cette optimisation est faisable car les probabilités de transitions de la DTMC agrégée sont calculées à la volée. Après à cette optimisation, il apparaît que le choix de l'instance symbolique suivante à franchir affecte la taille de la DTMC agrégée. Ainsi, notre seconde optimisation tente de minimiser la taille par une heuristique.

9.3.2.1 Regroupement des états symboliques

En fait, cette optimisation a été déjà proposée dans la section 7.1. Cependant, les conditions de ce regroupement ne concernaient que la préservation des chemins du graphe. Pour l'évaluation de performance, on ne peut pas se baser uniquement sur des critères qualitatifs pour opérer le regroupement des états. En effet, les probabilités de transitions doivent être prises en compte.

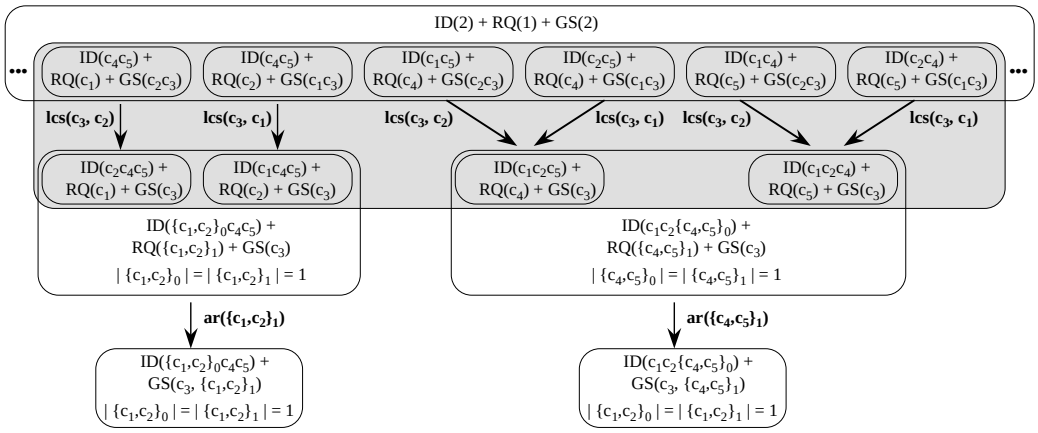


FIG. 9.5 – Franchissement et regroupement

Pour illustrer le problème qui peut apparaître, nous allons utiliser le modèle de la Fig. 9.4. Pour simplifier, nous considérons un taux uniforme pour toutes les instances de la transition *lcs*. Soit $P = 5$ le nombre des processus et $K = 3$ le nombre maximum de candidats simultanés. La Fig. 9.5 représente les distributions de jetons dans des places significatives du SWN et les franchissements des transitions que nous allons détailler dans cette section. La notation $\{c_i, c_j\}_k$ définit une partition de l'ensemble $\{c_i, c_j\}$: par exemple, l'état symbolique de gauche, $RQ(\{c_1, c_2\}_1)$ avec $|\{c_1, c_2\}_1| = 1$, signifie que l'une des deux objets c_1 ou c_2 est

dans la place *RQ* tandis que l'autre objet est dans la place *ID*.

On considère la séquence de franchissements à partir de la représentation symétrique où deux processus sont au repos, un troisième a envoyé une requête et deux autres exécutent une selection. La synchronisation du SWN et l'automate de contrôle pour le franchissement de la transition *lcs* se termine par un raffinement complet du marquage symbolique source, car la seule symétrie compatible avec l'étiquette $lcs[y < x]$ est l'identité ($\mathcal{H}_{lcs[y<x]} = \{id\}$). Pour chacun des marquages symboliques obtenus par le raffinement du marquage symbolique source il y a une seule instance symbolique de *lcs* franchissable. Un sous-ensemble des marquages symboliques obtenus par le franchissement de *lcs* est présenté dans la Fig. 9.5. A ce moment, il convient d'appliquer le regroupement sur les marquages symboliques obtenus par ces franchissements. Par rapport aux conditions d'application du regroupement qualitative (section 7.3.2.2), il faut aussi vérifier que les marquages à regrouper ont la même probabilité en entrée à partir du marquage symbolique source, pour garantir le respect de la condition de l'agrégation exacte. En ne considérant que les marquages symbolique représentés, comme nous avons considéré un taux uniforme pour transition *lcs*, la seule possibilité que nous avons est de grouper les deux marquages symboliques de droite, et aussi les deux de gauche. La partie ombrée est donc enlevée, pour ne stocker que la partie claire.

A partir des états symboliques construits, nous pouvons franchir la transition *ar*. Il n'y a aucune restriction associée à cette transition dans l'automate de contrôle, et donc on peut utiliser le franchissement classique des SWN.

9.3.2.2 Stratégie globale de construction

Nous montrons ici que l'optimisation précédente nécessite une stratégie efficace pour le choix de la prochaine transition à franchir, dans le but d'optimiser la taille de la DTMC agrégée.

Par exemple, ce qui peut se produire est que quelques états représentés par un état symbolique sont atteints par un autre franchissement. La Fig. 9.6 présente un exemple d'une telle situation. Considérons l'état symbolique avec deux processus dans la place *ID* et trois autres dans la place *GS*. La transition *lcs* est franchissable pour certains état représenté par cet état symbolique. En utilisant un processus de franchissement similaire à celui utilisé dans la Fig. 9.5, les différentes instances retenues mènent aux états avec deux processus dans la places *GS*, excepté (c_1c_2), et les trois autres dans la place *ID*. La configuration avec (c_1c_2) dans la place *GS* ne peut pas avoir lieu, car quelque soit le troisième objet de la configuration de

départ, il est plus prioritaire que c_1 et c_2 et par conséquent l'un de ces deux objets est systématique éliminé.

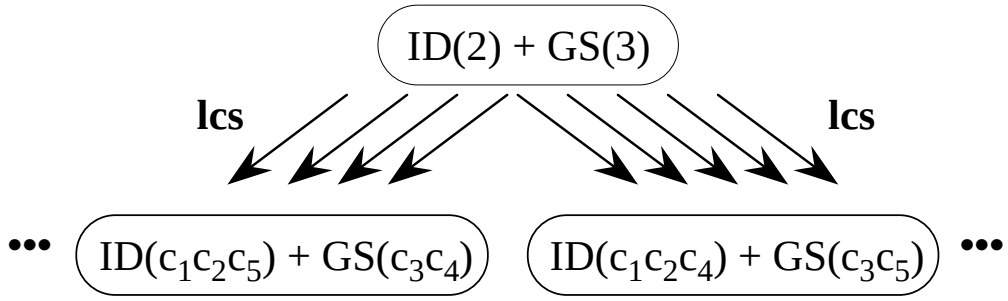


FIG. 9.6 – Construction des marquages inclus

Cette configuration symbolique à été déjà atteinte par le franchissement symbolique illustré dans la Fig. 9.5. Lors de ce franchissement on avait obtenu un seul état symbolique pour les états avec *GS* contenant (c_3c_4) ou (c_3c_5) (état symbolique de droite dans la Fig. 9.5), car ils sont issus d'un franchissement symétrique qui garantit des probabilités d'entrées identiques.

Ceci n'est plus vérifié dans la séquence que nous considérons actuellement : les états obtenus ne peuvent plus être groupés dans un seul état symbolique, car le nombre d'instances qui les atteignent sont différentes. On doit alors rajouter deux nouveaux états symboliques au DSRG. On est donc dans la situation ou un état apparaît plusieurs fois dans le DSRG.

Cependant, si nous avions considéré les séquences de franchissement dans l'ordre inverse, on aurait pu alors éviter cette redondance.

Ainsi, pour éviter autant que possible la construction de sous-classes d'états redondantes, nous proposons de traiter le franchissement des transitions asymétriques d'abord, pour favoriser la construction des états symboliques les plus petits (en termes de nombre d'états représentés).

9.4 Outil et expérimentations

L'approche symboliques que nous avons présenté a été implémenté en ré-utilisant le noyau de GreatSPN[5]. Initialement, cette outil a été développé pour la construction du SRG des WN, incluant une gestion efficace des représentations symbo-

[5] `http://www.di.unito.it/~greatspn/`

liques [CG95].

Par rapport à l'architecture proposée dans la partie qualitative de cette thèse (section 7.4), il a suffit de rajouter un module qui traite des différentes spécificités du modèle stochastique.

Dans cette section, nous considérons le modèle de la Fig. 9.4. Une étude attentive de la structure du modèle montre que sa complexité est fortement liée aux valeurs des paramètres P et K : P est le nombre des processus dans le système ; K est le nombre de processus pouvant prétendre un accès à la section critique simultanément ($K \leq P$). Par exemple, en se focalisant sur les place RQ et GS, on peut noter que K agit sur le nombre de jetons dans ces deux places, tandis que P élargit les possibilités de choix des identités des jetons qu'elles contiennent.

Comparons les effets de l'augmentation des valeurs de P et K sur les méthodes (S)RG et DSRG.

TAB. 9.1 – Taille des (S)RG et DSRG p.r.a. P and K

P \ K	3			5		
	(S)RG	DSRG	Ratio	(S)RG	DSRG	Ratio
3	45	23	1.66	–	–	–
5	441	49	9.00	573	186	3.08
7	3704	83	44.63	6231	772	8.07
9	28159	125	225.27	59281	1805	32.82
14	860371	199	4323.47	7210715	6148	1172.85

P \ K	7		
	(S)RG	DSRG	Ratio
3	–	–	–
5	–	–	–
7	6849	2150	3.18
9	73549	11150	6.60
14	17176671	68476	250.84

La Table 9.1 résume les experimentations. Les colonnes notées (S)RG (resp. DSRG) montrent le nombre des nœuds construits dans les chaque structure, pour P et K donnés.

En augmentant P et en fixant K, nous observons que le (S)RG se développe de manière exponentielle, tandis que le DSRG a une progression *presque linéaire*. Ceci est expliqué par le fait que la complexité induite par les différentes possibilités de sélectionner K processus concurrents sont explicitement représentées dans le (S)RG, alors qu'elles ont une représentation symbolique dans le DSRG. Plus précisément, dans le DSRG, aucune asymétrie entre les processus n'est prise en compte tant que la transition *lcs* n'est pas activée. De plus, l'opération de regroupement symbolique permet de "récupérer" une partie des symétries perdues lors d'un franchissement asymétrique.

Pour une valeur fixe de P, nous observons que la taille des deux structures augmente de manière exponentielle. Cependant, cette progression se stabilise quand les valeurs de P et K sont proches. Ceci est très prononcé pour le $(S)RG$, tandis que moins évident pour le DSRG. En effet, l'effet contextuel de l'opération de regroupement symbolique fait que cette progression reste incontrôlable.

Néanmoins, nous pouvons comparer le gain relative apporté par notre approche (voir les colonnes *Ratio*) : le ratio entre les deux structure progresse de manière exponentielle par rapport à P et ceci prouve l'efficacité de la méthode.

Enfin, nous noterons que la construction du DSRG est gourmande en termes de consommation en temps. Par exemple, pour les valeurs $P = 9$ et $K = 5$, la construction nécessite 275 unités de temps pour le DSRG, alors que 126 suffisent pour le (S)RG. En fait, 47% du temps total de construction est passée dans la comparaison entre ensemble de couleurs et ce pourcentage reste constant pour toutes les constructions. La résolution de ce problème est discuté dans la conclusion générale.

Chapitre 10

Conclusion Générale

Dans ce travail, nous nous sommes intéressés à la vérification qualitative et l'évaluation de performance des systèmes finis. Plus particulièrement, nous avons cherché à optimiser des solutions heuristiques pour la vérification d'un modèle de système concurrent.

Les optimisations adoptées et développées dans cette thèse sont basées sur les symétries (partielles) du modèle étudié. Contrairement aux approches classiques, nous avons choisi de séparer la représentation des comportements symétrique et asymétrique. Ainsi, le comportement réel du système est obtenu par *un produit synchronisé* des deux représentations Cette manière de procéder permet la construction d'un *produit synchronisé quotient* dans lequel les (a)symétries sont réévaluées dynamiquement durant le processus de vérification (ou d'évaluation de performances).

Cette approche n'a d'intérêt que lorsque nous avons le moyen de prendre en compte les symétries au niveau des propriétés à étudier.

Pour l'étude qualitative, les propriétés que nous spécifions sont des formules de logique linéaire. Nous obtenons ainsi une simplicité d'expression et une couverture importante des propriétés qualitatives des systèmes. En se plaçant au niveau de la sémantique des propriétés étudiées, *i.e., les automates de Büchi,* il est possible de détecter et d'évaluer automatiquement des symétries sur les états de tels automates et de les utiliser dans le processus de vérification développé.

Pour l'étude quantitative, nous nous sommes intéressés au processus stochastique associé au modèle étudié. Les symétries du modèle permettent de dériver automatiquement une chaîne de Markov agrégée. De plus, cette chaîne embarque toutes les informations nécessaires à l'étude du processus original, sans pour autant dé-

velopper la chaîne de Markov ordinaire.

Ces approches théoriques n'ont un sens pratique que si le processus de vérification et d'évaluation est complètement automatique : à partir de la spécification du système et de la propriété à vérifier, le processus de vérification (ou d'évaluation de performances) doit construire directement une structure réduite sans passer par la construction d'un graphe d'états complet.

Pour réaliser ces objectifs, nous avons fait le choix de représenter la partie symétrique par *un réseau de Petri bien formé symétrique*, qui représente le comportement *nominal* du système (autorisant certaines configurations du système qui ne sont pas "réelles"). Ce choix est justifié par l'existence d'une approche symbolique sur ce modèle qui permet la construction automatique d'un graphe quotient. La partie asymétrique est représentée par un automate fini basé sur les événements, appelé *automate de contrôle*, qui permet de restreindre les comportements du modèle symétrique à ceux réellement exécutés par le système. Ainsi, le comportement réel du système est obtenu par *un produit synchronisé* des deux composants.

Les expérimentations menées sur cette approche ont montré que dans le cas, très courant, des systèmes partiellement symétriques, la réduction de l'espace d'états nécessaire pour la vérification peut atteindre un facteur exponentiel. Ainsi, l'explosion combinatoire bien connue des techniques exploratoires sur les espaces d'états peut dans les cas pratiques être partiellement maîtrisée.

Pour mener ces expérimentations, nous avons réutilisé et étendu le noyau et les fonctionalités de base de l'outil GreatSPN. En effet, l'approche symbolique classique des réseaux de Petri bien formés a été complètement développée dans l'outil GreatSPN (dans une optique d'évaluation de performances). Ainsi, nous avons fait le choix de ne pas toucher (ou très légèrement) aux structures internes de l'outil et d'intégrer nos algorithmes sous la forme de modules externes, interagissant avec le noyau *via* des interfaces bien définies. Cette manière de procéder nous a permis une évaluation concrète et rapide de nos approches (sur des exemples de la littérature). Néanmoins, ceci reste une approche *ad-hoc*. En effet, nos *benchmarks* montrent que les temps de calcul sont élevés compte tenu de la taille des structures produites. L'étude approfondie que nous avons menée pour établir les causes principales de cette surestimation ont montré qu'une bonne partie du calcul est liée à la nécessité de s'adapter de façon récurrente aux structures internes de GreatSPN.

Pour palier à ce problème et pour ne pas perdre les acquis de l'outil, nous optons (dans un futur très proche) pour une refonte de GreatSPN et de ses extensions

suivant une approche modulaire.

Pour espérer une exploitation de ces technique en milieu industriel, différents problèmes restent ouverts :

– définir un language simple et rigoureux pour l'expression des contraintes sur le modèle introduit (les réseaux bien formés contrôlés). En effet, l'expression des contraintes sur les franchissements est effectuée, pour l'instant, au moyen d'un automate de contrôle spécifié manuellement. Pour éviter ce désagrément et les risques d'erreurs, nous espérons le développement d'un langage, simple et rigoureux, de spécification des contraintes à partir duquel l'automate sera généré automatiquement ;
– définir une caractérisation structurelle des modèles sur lesquels nos méthodes peuvent apporter des réductions significatives. En effet, une analyse structurelle automatique va permettre à l'utilisateur de prédire l'utilité de l'application de nos approches sans avoir à lancer les constructions à l'avance. L'intuition, ici, est d'identifier *les points de synchronisation* entres les comportements symétrique et asymétrique du modèle. Ces points vont garantir que, une fois que les objets du système ont fini leur comportement asymétrique, ils reviennent tous à un comportement symétrique ;
– étendre la partie évaluation de performances au *model-checking probabiliste*. Ceci à pour objectif d'établir des propriétés très précises sur les comportement temporel et probabiliste des objets du système.

Bibliographie

[AHl98] K. Ajami, S. Haddad, and J.-M. lié. Exploiting symmetry in linear time temporal logic model checking : One step beyond. In *Proceedings of the 4th International Conference on Tools and Algorithms for Construction and Analysis of Systems TACAS'98*, pages 52–67, London, UK, 1998. Springer-Verlag.

[AMO99] Mark D. Aagaard, Thomas F. Melham, and John W. O'Leary. Xs are for trajectory evaluation, booleans are for theorem proving. September 1999.

[BCM+90] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking : 10^20 states and beyond. In *LICS*, pages 428–439, 1990.

[BCMP75] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *J. ACM*, 22(2) :248–260, 1975.

[BDL06] S. Baarir and A. Duret-Lutz. Emptiness check of powerset Büchi automata. Technical report 2006/003, Universit ? Pierre et Marie Curie, LIP6-CNRS, Paris, France, October 2006.

[BDL07] Souheib Baarir and Alexandre Duret-Lutz. Emptiness check of powerset Büchi automata. In *Proc. of the ACSD*, June 2007.

[BDSH05] S. Baarir, C. Dutheillet, and J.M. Ilié S. Haddad. On the use of exact lumpability in partially symmetrical well-formed nets. In *Proceedings of International Conference on the Quantitative Evaluation of Systems*, pages 23–32, September 2005. Best paper award.

[Ber86] G. Berthelot. Checking properties of nets using transformation. In *Advances in Petri Nets 1985, covers the 6th European Workshop on Applications and Theory in Petri Nets-selected papers*, pages 19–40, London, UK, 1986. Springer-Verlag.

[Ber87] G. Berthelot. Transformations and decompositions of nets. In *Proceedings of an Advanced Course on Petri Nets : Central Mo-*

dels and Their Properties, Advances in Petri Nets 1986-Part I, pages 359–376, London, UK, 1987. Springer-Verlag.

[Beu03] B. Beutel. *Integration of the Petri Net Analysator TimeNET into the Model Analysis Environment MOSEL*. PhD thesis, Univ. of. Erlangen, Dept. of Comp. Science, Lehrstuhl 4, 2003.

[BFBI06] M. Beccuti, G. Franceschinis, S. Baarir, and J.M. Ilié. Efficient lumpability check in partially symmetric systems. In *Proceedings of International Conference on the Quantitative Evaluation of Systems*, pages 211–220, September 2006.

[BHI04] S. Baarir, S. Haddad, and J-M. Ilié. Exploiting Partial Symmetries in Well-formed nets for the Reachability and the Linear Time Model Checking Problems. In *Proceedings of WODES'04 - IFAC Workshop on Discrete Event Systems, part of 7th CAAP*, Reims - France, 2004. Springer Verlag.

[BHR84a] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3) :560–599, 1984.

[BHR84b] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3) :560–599, 1984.

[BIDL04] S. Baarir, J.M. Ilié, and A. Duret-Lutz. Improving reachability analysis for partially symmetric high level petri nets. In *Proceedings of the Poster Session of the 12th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'04)*, pages 5–8, Volendam, The Netherlands, October 2004.

[BM89] F. Baccelli and A.M. Makowski. Queueing models for systems with synchronization constraints. 77, Jan. 1989.

[BMT89] F. Baccelli, W. A. Massey, and D. Towsley. Acyclic fork-join queuing networks. *J. ACM*, 36(3) :615–642, 1989.

[BMvE98] A. Basu, J. G. Morrisett, and T. von Eicken. Promela++ : A language for constructing correct and efficient protocols. In *INFOCOM (2)*, pages 455–462, 1998.

[Bra85] A. Brandwajn. Equivalence and decomposition in queueing systems - a unified approach. *Perform. Eval.*, 5(3) :175–186, 1985.

[Bro91] Manfred Broy. Towards a formal foundation of the specification and description language SDL. *Formal Aspects of Computing*, 3(1) :21–57, 1991.

[BSS95a] E. Brinksma, G. Scollo, and C. Steenbergen. Lotos specifications, their implementations and their tests. pages 468–479, 1995.

[BSS95b] E. Brinksma, G. Scollo, and C. Steenbergen. Lotos specifications, their implementations and their tests. pages 468–479, 1995.

[Büc62] J. Richard Büchi. On a decision method in restricted second order arithmetic. In *Proceedings of the International Congress on Logic, Methodology, and Philosophy of Science, Berkley, 1960*, pages 1–11. Standford University Press, 1962.

[Buc95] P. Buchholz. A notion of equivalence for stochastic petri nets. In *Proceedings of the 16th International Conference on Application and Theory of Petri Nets*, pages 161–180, London, UK, 1995. Springer-Verlag.

[BZ83] D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2) :323–342, 1983.

[Cap00] L. Capra. *Expoiting partial symmetries in SWN models*. PhD thesis, Univerità degli studi di Torino, 2000.

[CDFH93] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic well-formed coloured nets for symmetric modelling applications. *IEEE Transactions on Computers*, 42(11) :1343–1360, nov 1993.

[CDFI99] L. Capra, C. Dutheillet, G. Franceschinis, and J-M. Ilié. Towards performance analysis with partially symmetrical swn. In *Proceedings of MASCOTS'99 - 7th Int. Symposium on Modeling, Analysis and Simulation*, college Park, MD, USA, 1999.

[CDFI00] L. Capra, C. Dutheillet, G. Franceschinis, and J-M. Ilié. Exploiting partial symmetries for Markov chain aggregation. *E. Notes in Theoretical Computer Science*, 39(3), 2000.

[CDLP05] J-M. Couvreur, A. Duret-Lutz, and D. Poitrenaud. On-the-fly emptiness checks for generalized Büchi automata. In *Proceedings of SPIN'05*, volume 3639 of *LNCS*, pages 143–158. Springer-Verlag, August 2005.

[CDS99] J. Campos, S. Donatelli, and M. Silva. Structured solution of asynchronously communicating stochastic modules. *Software Engineering*, 25(2) :147–165, 1999.

[CE81] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, pages 52–71, 1981.

[CEPA+02] J. M. Couvreur, E. Encrenaz, E. Paviot-Adet, D. Poitrenaud, and P. Wacrenier. Data decision diagrams for Petri net analysis. In *Proceedings of the 23th International Conference on Application and Theory of Petri Nets (ICATPN'02)*, volume 2360 of *Lecture Notes in Computer Science*, pages 101–120, Adelaide, Australia, June 2002. Springer Verlag.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2) :244–263, 1986.

[CG95] G. Chiola and R. Gaeta. Efficient Simulation of Parallel Architectures Exploiting Symmetric Well-formed Petri Net Models. In *Sixth International Workshop on Petri nets and Performance Models*, Durham, NC, USA, October 1995. IEEE Computer Society Press.

[CGP00] Jean-Michel Couvreur, Sébastien Grivet, and Denis Poitrenaud. Designing a LTL model-checker based on unfolding graphs. In *Proc. of the ICATPN'00*, volume 1825 of *LNCS*. Springer-Verlag, June 2000.

[CL88] DC. Marinescu C. Lin. Stochastic heigh-level petri nets ans applications. In *IEEE Transactions on Computers*, volume 37, pages 815–825, Jul 1988.

[Cou90] J.M. Couvreur. The general computation of flows of colored nets. 1990.

[Cou99] J-M. Couvreur. On-the-fly verification of temporal logic. In *Proceedings of the World Congress on Formal Methods in the Development of Computing Systems (FM'99)*, volume 1708 of *LNCS*, pages 253–271, Toulouse, France, 1999. Springer-Verlag.

[CP95] S. Christensen and L. Petrucci. Modular state space analysis of coloured Petri nets. In *Proc. 16th Int. Conf. Application and Theory of Petri Nets (ICATPN'95), Turin, Italy, June 1995*, volume 935, pages 201–217. Springer, 1995.

[CTM05] J. M. Couvreur and Y. Thierry-Mieg. Hierarchical decision diagrams to exploit model structure. In *FORTE*, pages 443–457, 2005.

[Dal90] Y. Dallery. Approximate analysis of general open queuing networks with restricted capacity. *Perform. Eval.*, 11(3) :209–222, 1990.

[DH89] C. Dutheillet and S. Haddad. Aggregation of states in colored stochastic petri nets : Application to multiprocessor architecture. In *Proceedings of the 3th Workshop on Petri Nets and Performance Models*, pages 40–49, Kyoto, Japan, 1989.

[DLT97] Y. Dallery, Z. Liu, and D. Towsley. Properties of fork/join queueing networks with blocking under various operating mechanisms. *IEEE Trans. on Robotics and Automation*, 13(4) :503–518, 1997.

[Don94] S. Donatelli. Superposed generalized stochastic petri nets : Definition and efficient solution. In *Proceedings of the 15th International Conference on Application and Theory of Petri Nets*, pages 258–277, London, UK, 1994. Springer-Verlag.

[Dut91] C. Dutheillet. *Symétries dans les Réseaux Colorés : Définition, Analyse et Appliction à l'Evalution de Performance*. PhD thesis, Université Paris-6, 1991.

[ECL93] O. Grumberg E.M. Clarke and D.E. Long. Verification tools for finite-state concurrent systems. In *REX School/Symposium*, pages 124–175, 1993.

[Eme95] E. A. Emerson. Automated temporal reasoning about reactive systems. In *Banff Higher Order Workshop*, pages 41–101, 1995.

[ES96] E. A. Emerson and A. P. Sistla. Symmetry and model checking. *Formal Methods in System Design*, 9(1/2) :105–131, 1996.

[ES97] E. A. Emerson and A. P. Sistla. Utilizing symmetry when model-checking under fairness assumptions : An automata-theoretic approach. *ACM Transactions on Programming Languages and Systems*, 19(4) :617–638, July 1997.

[ET99] E. A. Emerson and R. J. Trefler. From asymmetry to full symmetry : New techniques for symmetry reduction in model checking. In *Proceedings of CHARME99*, Lecture Notes in Computer Science, pages 142–156, Bad Herrenalb - Germany, September 1999. Springer Verlag.

[FN85] G. Florin and S. Natkin. *Les réseaux de Petri Stochastiques : Théorie, Thechniques de calcul et Application*. PhD thesis, Université Paris-6, 1985.

[GC88] T. Demaria G. Chiola, G. Bruno. Introducing a color formalism into generalized stochastic petri nets. In *Proceedings of 9th International Conference on Application and Theory of Petri Nets*, pages 202–215, Venise, Italie, June 1988.

[GL81] HJ. Genrich and K. Lautenbach. System modelling with high-level petri nets. *Theretical Computer Science*, 13 :109–136, 1981.

[GL02] D. Giannakopoulou and F. Lerda. From states to transitions : Improving translation of LTL formulæ to Büchi automata. In D.A. Peled and M.Y. Vardi, editors, *Proceedings of the 22nd IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE'02)*, volume 2529 of *Lecture Notes in Computer Science*, pages 308–326, Houston, Texas, November 2002. Springer-Verlag.

[GMKN00] D. Goldsman, W.S. Marshall, S.H. Kim, and B. L. Nelson. Ranking and selection for steady-state simulation. In *WSC '00 : Proceedings of the 32nd conference on Winter simulation*, pages 544–553, San Diego, CA, USA, 2000. Society for Computer Simulation International.

[GR85] M. G. Gouda and L. E. Rosier. Priority networks of communicating finite state machines. *SIAM J. Comput.*, 14(3) :569–584, 1985.

[GS90] H. Garavel and J. Sifakis. Compilation and verification of Lotos specifications. In Logrippo, R. L. Probert, and H. Ural, editors, *Proc. 10th International Symposium on Protocol Specification, Testing and Verification*, Amsterdam, 1990. Elsevier (North-Holland).

[GTW02] E. Grädel, W. Thomas, and T. Wilke. *Automata, logic, and infinite games*, volume 2500 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[GV05] J. Geldenhuys and A. Valmari. More efficient on-the-fly LTL verification with Tarjan's algorithm. *Theoretical Computer Science*, 345(1) :60–82, November 2005. Conference paper selected for journal publication.

[Had87] S. Haddad. *Une Catégorie particulière de Réseaux de Patri de Haut Niveau : Définition, Propriétés et Réductions.* PhD thesis, Universté de Paris VI, France, 1987.

[Had88] S. Haddad. A reduction theory for coloured nets. In *Proceedings of the European Workshop on Application and Theory of Petri Nets*, pages 209–235, 1988.

[HE93] B. Hofmann and W. Effelsberg. Efficient implementation of estelle specifications. Technical report, 1993.

[HIA00] S. Haddad, J.M. Ilié, and K. Ajami. A model checking method for partially symmetric systems. In *Proceedings of FORTE/PSTV'00*,

pages 121–136, Pisa, Italy, October 2000. Kluwer Academic Publishers.

[HIK04] S. Haddad, J-M. Ilié, and K. Klai. Design and evaluation of a symbolic and abstraction-based model checker. In *Proc. of ATVA'04*, volume 3299 of *LNCS*, pages 198–210. Springer-Verlag, October 2004.

[HITZ95] S. Haddad, J.M. Ilié, M. Taghelit, and B. Zouari. Symbolic Reachability Graph and Partial Symmetries. In *Proceedings of the* 16^{th} *Intern. Conference on Application and Theory of Petri Nets*, volume 935 of *LNCS*, pages 238–257, Turin, Italy, June 1995. Springer Verlag.

[HJJK86] HJ. Huber, AM. Jensen, LO Jespen, and K.Jensen. Reachability trees for high level petri nets. *Theretical Computer Science*, 45, 1986.

[HM95] S. Haddad and P. Moreaux. Evaluation of high level petri nets by means of aggregation and decomposition. In *PNPM '95 : Proceedings of the Sixth International Workshop on Petri Nets and Performance Models*, page 11, Washington, DC, USA, 1995. IEEE Computer Society.

[HM96] S. Haddad and P. Moreaux. Asynchronous composition of high level petri nets : A quantitative approach. In *Proceedings of the 17th International Conference on Application and Theory of Petri Nets*, pages 192–211, London, UK, 1996. Springer-Verlag.

[HR98] H. Hermanns and M. Ribaudo. Exploiting symmetries in stochastic process algebras, 1998.

[HTMK$^+$04] J. Hugues, Y. Thierry-Mieg, F. Kordon, L. Pautet, S. Barrir, and T. Vergnaud. On the formal verification of middleware behavioral properties. In *Proceedings of FMICS'04*, volume 133 of *ENTCS*, pages 139–157. Elsevier, September 2004.

[HV03] S. Haddad and F. Vernadat. *Vérification et Mise en Oeuvre des Réseaux de Petri - Dépliages pour la Vérification de Propriétés Temporelles*, chapter 1, pages 31–96. Hermès Science Publications, 2003. ISBN 2-7462-0445-2.

[IA97] J.-M. Ilié and K. Ajami. Model Checking through Symbolic Reachability Graph. In *Proceedings of Theory and Practice of Software Development TAPSOFT'97 - part of 7th CAAP*, volume 1214 of *LNCS*, pages 213–224, Lille - France, 1997. Springer Verlag.

[IBB+04] J.M. Ilié, S. Baarir, M. Beccuti, S. Donatelli, C. Dutheillet, G. Franceschinis, R. Gaeta, and P. Moreaux. Extended SWN Solvers in GreatSPN. In *Tool paper for the 1st Int. Conf. on Quantitative Evaluation of Systems (QEST'04)*, LNCS, Twente, Netherland, 2004. Springer Verlag.

[Jac63] J. R. Jackson. Jobshop-like queueing systems. *Management Sciences*, 10(1) :131–142, 1963.

[Jen82] K. Jensen. High-level petri nets. In *Proceedings of the 3rd European Workshop on Application and Theory of Petri Nets*, Varenna - Italy, 1982.

[Kle75] L. Kleinrock. *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, 1975.

[KS60] J.G. Kemeny and J.L. Snell. Finite Markov chains. New York, NY, 1960. D. Van Nostrand-Reinhold.

[Mar79] R. A. Marie. An approximate analytical method for general queueing networks. *IEEE Trans. Software Eng.*, 5(5) :530–538, 1979.

[McM93] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[Mil89a] R. Milner. *Communication and concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[Mil89b] R. Milner. *Communication and concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[Mol81] M. K. Molloy. *On the integration of delay and throughput measures in distributed processing models*. PhD thesis, 1981.

[Mur89] T. Murata. Petri nets : Properties, analysis and applications. *Procedings of the IEEE*, 77(4) :541–580, April 1989.

[NM94] M. Notomi and T. Murata. Hierarchical reachability graph of bounded petri nets for concurrent-software analysis. *IEEE Trans. Softw. Eng.*, 20(5) :325–336, 1994.

[Pet62] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Bonn : Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

[Rei91] W. Reisig. Petri nets and algebric specifications. *Teoretical Computer Science*, 80 :1–34, 1991.

[Sch84] P. J. Schweitzer. Aggregation methods for large Markov chains. In *Proceedings of the International Workshop on Computer Performance and Reliability*, pages 275–286. North-Holland, 1984.

[SE97] M. Silva and E.Teruel. Petri nets for the design and operation of manufactoring systems. *European Journal of Control*, 3(3), 1997.

[SE05] S. Schwoon and J. Esparza. A note on on-the-fly verification algorithms. In *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05)*, Lecture Notes in Computer Science. Springer-Verlag, April 2005. To appear.

[SF86] D. Mailles S. Fdida, G. Pujolle. Réseaux de files d'attente avec sémaphores. *Techniques et Sciences Informatiques (TSI)*, 5(3) :187–196, 1986.

[Tar71] R. Tarjan. Depth-first search and linear graph algorithms. In *Conference records of the 12th Annual IEEE Symposium on Switching and Automata Theory*, pages 114–121. IEEE, October 1971. Later republished as [Tar72].

[Tar72] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2) :146–160, 1972.

[TMBDLK04] Y. Thierry-Mieg, S. Baarir, A. Duret-Lutz, and F. Kordon. Nouvelles techniques de model-checking pour la vérification de systèmes complexes. *Génie Logiciel*, 69 :17–23, 2004.

[TMIB06] Y. Thierry-Mieg, J.M. Ilié, and S. Baarir. *Méthodes formelles pour les systèmes répartis et coopératifs-Vérifications efficace des systèmes finis*, chapter 8, pages 171–207. Hermès Science Publications, 2006. ISBN : 2746214474.

[TMIP04] Y. Thierry-Mieg, J. M. Ilié, and D. Poitrenaud. A symbolic symbolic state space representation. In David de Frutos-Escrig and Manuel Núñez, editors, *Proceedings of the 24th International Conference on Formal Techniques for Networked and Distributed Systems (FORTE'04)*, volume 3235 of *Lecture Notes in Computer Science*, pages 276–291. Springer, September 2004.

[Val93] A. Valmari. Compositional state space generation. In *P12th International Conference on Applications and Theory of Petri Nets*, pages 427–457, London, UK, 1993. Springer-Verlag.

[Val97] R. Valette. Some issues about Petri net application to manufacturing and process supervisory control. In *ICATPN*, pages 23–41, 1997.

[VAM96] F. Vernadat, P. Azema, and F. Michel. Covering step graph. In *Application and Theory of Petri Nets*, pages 516–535, 1996.

[Var96] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In Faron Moller and Graham M. Birtwistle, editors, *Proceedings of the 8th Banff Higher Order Workshop (Banff'94)*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266, Banff, Alberta, Canada, 1996. Springer-Verlag.

[WG93] P. Wolper and P. Godefroid. Partial-order methods for temporal verification. In *CONCUR '93 : Proceedings of the 4th International Conference on Concurrency Theory*, pages 233–246, London, UK, 1993. Springer-Verlag.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1–2) :72–99, 1983.

[ZE99] C. Zlatea and T. Elrad. A design methodology for mobile distributed applications based on unity formalism and communication-closed layering. In *PODC '99 : Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, page 284, New York, NY, USA, 1999. ACM Press.

[Zin87] A. Zinié. *Les réseaux de Petri Stochastiques Colorés. Application à l'analyse des systèmes Répartis en Temps Réel*. PhD thesis, Université Paris-6, 1987.