Titre: Title:	Approches formelles pour la modélisation et la vérification du contrôle d'accès et des contraintes temporelles dans les systèmes d'information
Auteur: Author:	Hind Rakkay
Date:	2009
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Rakkay, H. (2009). Approches formelles pour la modélisation et la vérification du contrôle d'accès et des contraintes temporelles dans les systèmes d'information [Ph.D. thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/123/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/123/
Directeurs de recherche: Advisors:	Hanifa Boucheneb
Programme: Program:	Génie informatique

UNIVERSITÉ DE MONTRÉAL

APPROCHES FORMELLES POUR LA MODÉLISATION ET LA VÉRIFICATION DU CONTRÔLE D'ACCÈS ET DES CONTRAINTES TEMPORELLES DANS LES SYSTÈMES D'INFORMATION

HIND RAKKAY
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INFORMATIQUE)
AVRIL 2009

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

APPROCHES FORMELLES POUR LA MODÉLISATION ET LA VÉRIFICATION DU CONTRÔLE D'ACCÈS ET DES CONTRAINTES TEMPORELLES DANS LES SYSTÈMES D'INFORMATION

présentée par: RAKKAY Hind

en vue de l'obtention du diplôme de: Philosophiæ Doctor

a été dûment acceptée par le jury d'examen constitué de:

M. QUINTERO Alejandro, Doct., président

Mme. BOUCHENEB Hanifa, Doctorat, membre et directrice de recherche

Mme. NICOLESCU Gabriela, Doct., membre

M. DEBBABI Mourad, Ph.D., membre

À mes parents pour leur soutien et leur confiance

À mes beaux parents pour leurs encouragements

À tous ceux qui m'aiment et ensoleillent ma vie

REMERCIEMENTS

Les travaux présentés dans cette thèse ont été effectués au Laboratoire de Vérification Formelle des Systèmes Temps-Réel de l'École Polytechnique de Montréal.

Je tiens donc, à adresser mes vifs remerciements au professeur Hanifa Boucheneb pour son encadrement, son écoute et sa contribution à titre de directrice de cette thèse.
J'exprime ma profonde gratitude au professeur Alejandro Quintero, pour l'honneur qu'il me fait en présidant mon jury de thèse, ainsi qu'à Mme. Gabriela Nicolescu, Professeur à l'École Polytechnique et M. Mourad Debbabi, Professeur à l'Université Concordia pour l'honneur qu'ils me font en participant à mon jury de thèse.

Je voudrais aussi vivement remercier le professeur Olivier H. Roux de l'Université IRCCyN de Nantes, les professeurs du département de Génie Informatique de l'École Polytechnique ainsi que le professeur Gerard Buzaglo du département de Génie Industriel et Mathématiques pour son soutien et les opportunités qu'il m'a offertes en enseignement.
Mes remerciements vont naturellement à toutes mes collègues du laboratoire, à mes amis de Poly, en France, au personnel de l'École Polytechnique, au comité IEEE-Poly, à la section de IEEE-Montréal, à mes étudiants des cours Calcul Différentiel et Algèbre Vectorielle, à Marie-Carline pour sa jovialité et son sens de l'humour, ainsi qu'aux collègues de ma sœur à SNC-Lavalin pour leur précieux coup de main.

Dans ces moments importants, je pense très fort à ma famille ici, au Maroc et en Tunisie. Particulièrement mes parents, mes beaux parents ainsi que mon frère et mes soeurs : Amine, Emna, Meriem et Salima. Un grand merci aussi à mon très cher fiancé Amine d'avoir toujours été là, d'un bon conseil et d'une aide précieuse. Mon salut à son adorable poisson Kiki ;) Je remercie tous mes proches pour leur soutien.
À tous un grand merci du fond du cœur.

RÉSUMÉ

Nos travaux de recherche s'inscrivent dans un cadre qui vise à développer des approches formelles pour aider à concevoir des systèmes d'information avec un bon niveau de sûreté et de sécurité. Précisément, il s'agit de disposer d'approches pour vérifier qu'un système fonctionne correctement et qu'il implémente une politique de sécurité qui répond à ses besoins spécifiques en termes de confidentialité, d'intégrité et de disponibilité des données. Notre recherche s'est ainsi construite autour de la volonté de développer, valoriser et élargir l'utilisation des réseaux de Petri en tant qu'outil de modélisation et le model-checking en tant que technique de vérification. Notre principal objectif est d'exprimer la dimension temporelle de manière quantitative pour vérifier des propriétés temporelles telles que la disponibilité des données, la durée d'exécution des tâches, les deadlines, etc. Tout d'abord, nous proposons une extension du modèle TSCPN (*Timed Secure Colored Petri Net*), initialement présenté dans mon mémoire de maîtrise. Le modèle TSCPN permet de modéliser et de raisonner sur les droits d'accès aux données exprimés via une politique de contrôle d'accès mandataire, i.e. Modèle de Bell-LaPadula. Ensuite, nous investigons l'idée d'utiliser les réseaux de Petri colorés pour représenter les politiques de contrôle d'accès à base de rôles (*Role Based Access Control* - RBAC). Notre objectif est de fournir des guides précis pour aider à la spécification d'une politique RBAC cohérente et complète, appuyée par les réseaux de Petri colorés et l'outil CPNtools. Finalement, nous proposons d'enrichir la classe des réseaux de Petri temporels par une nouvelle extension qui permet d'exprimer plus d'un seul type de contraintes temporelles. Il s'agit du modèle $TA_{WS}PN$ (*Timed Arc Petri net - Weak and Strong semantics*). Notre but étant d'offrir une grande flexibilité dans la modélisation de systèmes temporisés complexes sans complexifier les méthodes d'analyse classiques. En effet, le modèle $TA_{WS}PN$ offre une technique de model-checking, basée sur la construction de graphes des zones (Gardey et al., 2003), comparables à celles des autres extensions temporelles des réseaux de Petri.

ABSTRACT

Our research is integrated within a framework that aims to develop formal approaches to help in the design of information systems with a good level of safety and security. Specifically, these approaches have to verify that a system works correctly and that it implements a security policy that meets its specific needs in terms of data confidentiality, integrity and availability. Our research is thus built around the aim to develop, enhance and expand the use of Petri nets as a modeling tool and the *Model-checking* as a verification technique. Our main objective is to express the temporal dimension in order to check quantitative temporal properties such as data availability, task execution duration, deadlines, etc.

First, we propose an extension of the TSCPN (*Timed Secure Colored Petri Net*) model, originally presented in my master's thesis. This model allows representing and reasoning about access rights, expressed via a mandatory access control policy, i.e. Bell-LaPadula model. In a second step, we investigate the idea of using colored Petri nets to represent role based access control policies (RBAC). Our goal is to provide specific guidelines to assist in the specification of a coherent and comprehensive RBAC, supported by colored Petri nets and CPNtools. Finally, we propose to enrich the class of time Petri nets by a new extension that allows to express more than one kind of time constraint, named $TA_{WS}PN$ (*Timed-Arc Petri net Weak and Strong semantics*). Our goal is to provide great flexibility in modeling complex systems without complicating the conventional methods of analysis. Indeed, the $TA_{WS}PN$ model offers a model-checking technique based on the construction of zone graphs (Gardey et al., 2003), comparable to those of other extensions of timed Petri nets.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES NOTATIONS ET DES SYMBOLES

ACL	Access Control List
ARBAC	Administration of Role Based Access Control
ASP	Algebra of Communicating Processes
CCS	Calculus of Communicating Systems
CSP	Communication Sequencial Processes
CTL	Computation Tree Language
DAC	Discretionary Access Control
DBM	Difference Bound Matrix
GTRBAC	Generalized Temporal Role Based Access Control
ITCPN	Interval Timed Colored Petri net
ITSEC	Information Technology Security Evaluation Criteria
LTL	Linear Temporal Language
LTS	Labeled Transition System
MAC	Mandatory Access Control
MLS	Multi-level Security
ORBAC	Organization Role Based Access Control
RBAC	Role Based Access Control
SCG	State Class Graph
SI	Système d'Information
SICSS	Système dInformation et de Communication en Santé et social
SPIN	Simple Promela Interpreter
STS	Strong Time Semantics
TBAC	Task Based Authorization Controls
TMAC	Team-based Access Control
WTS	Weak Time Semantics

XACML	Extensible Access Control Markup Language
ZBG	Zone Based Graph

LISTE DES TABLEAUX

CHAPITRE 1

INTRODUCTION GÉNÉRALE

1.1 Contexte de la recherche

Aujourd'hui, les progrès technologiques, la disponibilité des ordinateurs et de réseaux informatiques puissants ainsi que la nécessité d'augmenter l'efficacité des opérations, ont amené l'ensemble des organisations à privilégier les systèmes d'information (SI). L'information constitue ainsi l'une des principales richesses de plusieurs organisations. Il est donc essentiel de pouvoir protéger cette richesse. La protection (Lampson, 1974) des SI présente un enjeu de grande envergure pour la société. En effet, une simple faille de sécurité dans le système peut entraîner une divulgation d'informations, une altération de données, un déni de service et une utilisation illicite des ressources. Les conséquences sont parfois très lourdes : "des renseignements inexacts sur un dossier médical ou un fichier de crédit peuvent ruiner des réputations ou occasionner un mauvais diagnostic de maladie, des renseignements recueillis dans les banques de données peuvent porter préjudice à la vie privée des gens, et des renseignements spécialisés peuvent être utilisés par des groupes d'intérêt afin d'obtenir un contrôle injuste sur d'autres groupes[1]".

Le recours aux systèmes d'information dans de nombreuses activités critiques ajoute donc une contrainte forte en termes de sécurité et de sûreté de fonctionnement qu'il importe de maîtriser. Dans la présente thèse, nous traitons de la mise en oeuvre des méthodes formelles dans le cadre de la sécurité des systèmes d'information.

Dans l'intention de contrer les risques d'erreurs et défaillances dans un SI, une politique

[1] http://www.thecanadianencyclopedia.com/

de sécurité est généralement définie pour fixer un cadre dans lequel doit évoluer chaque utilisateur du système. Le *Information Technology Security Evaluation Criteria* ITSEC (ITSEC, 1991), appelé “Livre orange” est un standard de sécurité émis par le Conseil de sécurité des Etats-Unis *National Computer*. Il définit une politique de sécurité par “l’ensemble des lois, règles et pratiques qui régissent la façon dont l’information sensible et les autres ressources sont gérées, protégées et distribuées à l’intérieur d’un système spécifique ”.

La politique de sécurité d’un système informatique établit donc, les actions autorisées dans ce système, l’ensemble des propriétés de sécurité qui doivent être assurées par le système ainsi que les dispositifs pour les assurer. La sécurité informatique recouvre trois types d’objectifs selon les ITSEC :

- La confidentialité des données (aucune accès illicite): garantir que l’information dans le système est uniquement accessible aux utilisateurs autorisés. Elle ne doit être, ni rendue accessible, ni divulguée à un utilisateur, une entité ou un processus non autorisé.

- L’intégrité des données (aucune falsification): éviter la corruption ou la destruction des données traitées par le système. Il faut d’abord empêcher les utilisateurs malveillants d’accéder aux données non autorisées (assurer la confidentialité et l’isolation des informations) et également assurer que les données sont accédées de façon cohérente.

- La disponibilité des données (aucun retard): garantir que les ressources et les services du système sont disponibles aux utilisateurs autorisés.

1.2 Problématique de la recherche

1.2.1 Évaluation du niveau de sécurité

Les lignes directrices du (ITSEC, 1991) sont souvent utilisées pour évaluer le niveau de confiance dans la sécurité d'un SI. Le ITSEC identifie six classes d'évaluation des produits de sécurité, qui sont conçues pour regrouper des besoins typiques de sécurité. Le tableau 1.1 montre les niveaux et résume leurs caractéristiques.

Tableau 1.1 Les niveaux de sécurité et leurs caractéristiques

Classes d'évaluation	**Caractéristiques**
D	Protection minimale. On y regroupe ce qui ne satisfait pas les catégories les plus élevées.
C1	Protection discrétionnaire Contrôle d'accès discrétionnaire des objets en fonction de l'identification des usagers.
C2	Protection par accès contrôlé Tests plus rigoureux que C1
B1	Protection par étiquetage Contrôle d'accès mandataire basé sur des classes de sécurité.
B2	Protection structurée Support du principe du moindre privilège requis et analyse des canaux cachés possibles.
B3	Domaines de sécurité Haute résistance à la pénétration.
A1	Conception vérifiée - Distribution sécurisée, imposant un état sécuritaire de transport du système. - Analyse formelle et preuve mathématique que la conception du système est conforme à la politique de sécurité et aux spécifications de conception.

Tel qu'on peut le voir dans ce tableau, deux aspects importants sont à considérer lors de

la conception d'un système sécuritaire et fonctionnel : la formalisation de la politique de sécurité et la vérification de la conception du système. La conception du système devrait donc être compatible avec le modèle de sécurité, et doit être prouvée par des méthodes formelles. Ces méthodes constituent une approche basée sur un raisonnement rigoureux de logique mathématique, et comportent ainsi, par la preuve, la démonstration de leur propre correction.

1.2.2 Les méthodes formelles

Dans le domaine du génie logiciel, les méthodes formelles sont utilisées pour renforcer la robustesse des logiciels. " On peut comparer les fonctions du système logiciel au plan d'une maison. Dans la méthode formelle, toutes les fonctions sont décrites principalement par notations mathématiques. Or, la notation mathématique implique nécessairement une preuve mathématique. C'est là toute la différence avec la méthode traditionnelle, qui ne peut pas prouver ce qu'elle décrit; elle ne peut que tester après coup[2]."

Au cours des dernières années, les démarches de vérification formelle ont connu un essor croissant dans le domaine de la vérification des politiques de sécurité. Cependant, il s'agit d'un sujet d'étude encore émergent.

La vérification d'une politique de sécurité se fait généralement en deux étapes qui sont complémentaires : d'une part, la formalisation de la politique sous la forme d'un langage ou d'un modèle formel qui exprime les exigences de sécurité propres au système auquel la politique est destinée, et d'autre part, la vérification au moyen de techniques et outils appropriés permettant de vérifier que ces exigences seront effectivement satisfaites après une implémentation dans le système cible.

[2]http://www.usherbrooke.ca/courrier_sciences/2005/09/frappier.html.

Ce cadre bien particulier a connu le foisonnement d'importantes recherches pour intégrer les méthodes formelles dans la vérification des systèmes de sécurité. Les méthodes proposées diffèrent entre elles selon le type de systèmes et leur contexte d'utilisation (protocoles de sécurité, détection d'intrusion, analyse de programmes, système d'information), les mécanismes de sécurité mis en oeuvre (contrôle d'accès ou contrôle du flot d'information), les propriétés de sécurité visées (confidentialité, intégrité, disponibilité, anonymat), le formalisme de description utilisé (matrice d'accès, réseaux de Petri, algèbres de processus ...), ou encore selon les approches de vérification (les méthodes de preuve, le "*model checking*").

De plus, les systèmes réels ont souvent des comportements qui dépendent du temps. La capacité de modéliser et de manipuler la dimension temporelle des évènements qui se déroulent dans le monde réel est fondamentale dans un grand nombre d'applications. Ces applications peuvent concerner le secteur bancaire, le secteur médical, ou encore les applications multimédia. La variété des applications motive nombreux travaux récents qui visent à intégrer toutes les fonctionnalités nécessaires à la prise en compte du temps lors de la vérification.

1.3 Méthodologie de la recherche

Dans la présente thèse, nous considérons les réseaux de Petri (RdP) (Petri, 1962) pour modéliser les politiques de contrôle d'accès. Les RdP sont un formalisme utilisé pour différents types d'applications réactives, telles que les protocoles de communication, les téléphones portables, les ateliers flexibles, etc. Ils ont le double avantage de fournir un support graphique naturel qui est d'une aide précieuse dans l'analyse, et de posséder des propriétés analytiques qui permettent une évaluation simple du comportement du système étudié. En outre, les RdP se prêtent à la simulation lorsque le système modélisé est trop complexe pour autoriser une étude analytique. C'est donc un outil complet et

efficace.

Nous nous intéressons également au model-checking comme technique de vérification. Le model-checking, introduit en 1980 par Queille et Sifakis, et par Clarke et Emmerson, est une technique de vérification automatique qui présente l'avantage indéniable de permettre une détection rapide et économique des erreurs et ce, dès les premières phases du processus de conception. Le model-checking se base sur la construction de l'espace des états accessibles et permet par la suite, de vérifier le respect des propriétés attendues du système en parcourant l'ensemble de ces états.

De plus, nous traitons la problématique de la modélisation et la vérification du contrôle d'accès et des contraintes temporelles dans le contexte des systèmes workflows (processus métiers). En effet, le terme workflow (ou la gestion automatique du flux de travail) désigne un travail coopératif impliquant un nombre limité de personnes devant accomplir, en un temps limité, des tâches articulées autour d'une procédure définie et ayant un objectif global.

Pour concevoir un bon système workflow, il est impératif d'effectuer en amont un travail de modélisation de processus afin de décrire formellement les activités, les tâches, les règles de transition et les données du système. Par la suite, il faut formaliser les exigences du système, et décrire comment seront vérifiées ces exigences et simuler le fonctionnement du futur système. Tout particulièrement, une bonne approche de modélisation pour les systèmes workflows doit assurer les éléments suivants :
- la mécanisation des flux d'informations (assurer le flux d'information entre les tâches),
- l'automatisation de l'ordonnancement des tâches (selon les règles de transition exprimées dans le modèle),
- la distribution des rôles (le travail à faire est affecté au bon acteur et au bon moment),
- la traçabilité du processus (pour savoir à tout moment ce qui a été fait, par qui, quand et où et ce qu'il reste à faire).

1.4 Contributions

L'état de l'art a montré que de nombreux modèles de contrôle d'accès ont été proposés pour organiser les droits des utilisateurs selon les besoins des organisations, ainsi que plusieurs formalismes pour modéliser leurs structures, leurs principes et leurs propriétés. Toutefois, la plupart des propositions se limitent à la modélisation et n'offrent pas des techniques pour vérifier les objectifs de sécurité visés, certains travaux se sont penchés à la fois sur les problèmes de modélisation et de vérification, sans toutefois, répondre à tous les objectifs de sécurité, mais peu de modèles prennent en compte le facteur temps lors de la modélisation sachant la complexité de la vérification qui en découle.

La problématique porte ainsi, sur la modélisation d'une part et la vérification d'autre part. Comme il est cependant difficile de dissocier l'une de l'autre, l'intégration d'une méthode formelle pour la vérification de la sécurité doit être le résultat d'un bon compromis entre la puissance de modélisation, la complexité et la décidabilité de la vérification.

La thèse ne vise pas à définir un nouveau modèle de contrôle d'accès mais à proposer une formalisation des politiques de contrôle d'accès qui prend en compte le facteur temps ainsi qu'une approche de vérification appropriée aux modèles temporels pour vérifier les propriétés de sécurité souhaitées. L'intérêt est de pouvoir mener une validation théorique de certains aspects de la politique préalablement à son intégration dans des SI. Cela a pour avantage de permettre au concepteur d'effectuer un premier niveau de vérification à partir d'une modélisation représentative de la politique qui sera mise sur le système. Eventuellement, après le déploiement du système, cette approche peut être d'une aide aux administrateurs de sécurité pour la mise à jour et la maintenance de la politique de sécurité.

Afin de réaliser ces objectifs, nous avons défini un cadre formel qui permet d'exprimer et d'inférer sur le plus grand nombre d'aspects présentés dans l'état de l'art. Ainsi, nos

principales contributions sont :

1- Proposition d'un cadre formel pour modéliser et vérifier des systèmes soumis à des politiques de contrôle d'accès (politiques mandataires multi-niveaux). Dans un premier temps, ma thèse de doctorat est le prolongement de mes travaux de recherche à la maîtrise où on a exploré l'utilisation des réseaux de Petri colorés (CPN) (Jensen, 1997) et les extensions temporelles pour modéliser et vérifier des systèmes soumis à des politiques de type mandataire (Bell-LaPadula). Ce projet de maîtrise a conduit à l'aboutissement du modèle TSCPN (*Timed Secured Colored Petri net*) ainsi qu'une approche de vérification par model-checking (Rakkay and Boucheneb, 2006). Dans la présente thèse, nous proposons une extension du modèle TSCPN. Nous présentons aussi une nouvelle approche d'analyse plus concise et adéquate à l'utilisation du model-checking. Finalement, en plus des propriétés de sécurité de base, nous vérifions aussi la propriété de l'opacité suivant une approche basée sur le contrôle du flot d'information.

2- Modélisation des modèles RBAC au moyen des réseaux de Petri Colorés et l'outil CPNtools. Un second travail s'est achevé par une formalisation des modèles RBAC (Rakkay and Boucheneb, 2009). Notre objectif est de fournir des guides précis pour aider à la spécification d'une politique RBAC cohérente et complète, appuyée par les réseaux de Petri colorés et l'outil CPNtools[3]. Nous utilisons l'outil CPNtools qui offre un environnement permettant de réaliser la modélisation, de la simuler et de générer automatiquement le graphe des marquages accessibles (i.e. l'espace des états) du modèle. Nous montrons que les réseaux de Petri représentent un formalisme intuitif pour aider à la détection des incohérences et incomplétudes dans les règles d'une politique de sécurité RBAC. Les anomalies auxquelles on s'intéresse peuvent être exprimées par des propriétés d'atteignabilité pour déceler la redondance et l'incomplétude des règles, des propriétés de sûreté pour vérifier l'inconsistance, ou des propriétés de vivacité pour la

[3]http://wiki.daimi.au.dk/cpntools/cpntools.wiki.

présence d'états de blocage. Ces anomalies peuvent être détectées en procédant à une analyse du graphe de marquages produit par l'outil CPN à partir du réseau de Petri modélisant la politique RBAC.

3- Définition d'une extension des réseaux de Petri Colorés à arcs temporels généralisés à plusieurs sémantiques. Dans une perspective d'intégrer la dimension temporelle dans la vérification de la sécurité, nous avons remarqué que les contraintes temporels peuvent être de différentes natures : des contraintes de durée, de périodicité, des contraintes douces, fortes, etc. D'autre part, il n'existe pas des réseaux de Petri temporels qui permettent d'exprimer, dans un même modèle, plusieurs types de contraintes. Pourtant, un système réel est très souvent sujet à une variété de contraintes temporelles dont il importe d'en tenir compte pour une modélisation plus réaliste. Pour cela, nous proposons une nouvelle extension des réseaux de Petri à arcs temporels, nommée $TA_{WS}PN$ (*Timed Arc Petri net* / *Weak and Strong semantics*) (Rakkay et al., 2007). Cette extension permet d'inclure, dans un même modèle, différentes sémantiques de franchissement et offre ainsi plus de flexibilité dans la modélisation des systèmes temporisés complexes. Au delà de cette flexibilité, l'intérêt de ce modèle est d'offrir également des possibilités de vérification comparables à celles des autres extensions temporelles des réseaux de Petri.

4- Application aux systèmes Workflows. Nous proposons de modéliser les systèmes Workflows au moyen du modèle généralisé, $TA_{WS}PN$, pour prendre en compte diverses contraintes temporelles (Yu et al., 2004). Il y a des contraintes associées aux sujets pour exprimer le moment et la durée d'assignation d'un rôle à un usager, des contraintes associées aux données pour spécifier leur disponibilité et validité, des contraintes sur la durée d'exécution d'une tâche et éventuellement du processus en entier, etc. D'autre part, nous proposons d'intégrer le contrôle d'accès basé sur les rôles (RBAC) dans la modélisation des Workflows. Une approche globale, basée sur les réseaux de Petri, permettrait de tracer à la fois le flux des données, de contrôle et d'autorisation entre les

différents intervenants du sytème.

1.5 Organisation du document

Après cette introduction générale de la thèse, le chapitre 2 rappelle quelques concepts de base relatifs à la sécurité et présente certaines politiques et modèles de sécurité existants. Le chapitre 3 présente les différentes approches de modélisation des politiques de sécurité ainsi qu'une recherche non-exhaustive des études qui utilisent ces approches. De la même façon, nous présentons dans le chapitre 4 les approches de vérification des politiques de sécurité et la revue de littérature associée. Dans le chapitre 5, nous abordons les aspects importants dans la modélisation des systèmes workflows. Nous y discutons aussi de l'utilisation des réseaux de Petri pour la modélisation des workflows aussi bien pour ajouter le temps, que pour inclure le contrôle d'accès. Dans le chapitre 6, nous présentons une extension du modèle TSCPN. Le chapitre 7 présente notre approche de modélisation des modèles RBAC au moyen des réseaux de Petri colorés. Le chapitre 8 est consacré aux réseaux de Petri temporels. Par la suite, nous y présentons la définition du modèle $TA_{WS}PN$ et sa sémantique. Pour finir, nous y proposons une application du modèle $TA_{WS}PN$ et du modèle RBAC au domaine des Workflows. Finalement, nous terminons ce document par une conclusion et quelques perspectives de recherche.

CHAPITRE 2

POLITIQUES ET MODÈLES DE SÉCURITÉ

Ce chapitre est consacré à la présentation de différentes notions permettant de préciser le cadre et le contexte de nos travaux. Par la suite, nous ferons un état de l'art des modèles de sécurité, leurs caractéristiques et limites.

2.1 Notions préliminaires

2.1.1 Relation d'ordre

Soit E un ensemble. On nomme relation d'ordre sur E toute relation binaire, à la fois :

- Réflexive
- Antisymétrique
- Transitive

La relation $\leq$ dans $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ et $\mathbb{R}$ est une relation d'ordre ssi:

- quelque soit x: $x \leq x$ (réflexivité)
- si $x \leq y$ et $y \leq x$, alors : $x = y$ (antisymétrie)
- si $x \leq y$ et $y \leq z$ alors $x \leq z$ (transitivité)

Une relation d'ordre $\leq$ sur E est dite *relation d'ordre total* si chaque paire d'éléments de E sont comparables, c'est-à-dire on a $x \leq y$ ou bien $y \leq x$. Si par contre il existe au moins un couple (x, y) où x et y ne sont pas comparables la relation $\leq$ est dite *relation d'ordre partiel*. L'ensemble E est dit un *ensemble ordonné* s'il est muni d'une relation d'ordre.

2.1.2 Majorant/Minorant et Borne supérieure/inférieure

Soit $(E, \leq)$ un ensemble ordonné. Soit $A \subseteq E$.

- Un majorant de A est un $x \in E$ tel que pour tout $y \in A$, $y \leq x$.
- Un minorant de A est un $x \in E$ tel que pour tout $y \in A$, $x \leq y$.
- La borne supérieure de A, notée $lub(A)$[1], est le plus petit des majorants de A (si z est un majorant de A alors $lub(A) \leq z$).
- La borne inférieure de A, notée $glb(A)$[2], est le plus grand des minorants de A (si z est un minorant de A alors $z \leq glb(A)$).

2.1.3 Treillis

Si, dans l'ensemble ordonné $(E, \leq)$, toute paire $\{x, y\}$ d'éléments de E admet une borne supérieure et une borne inférieure, on parle d'ensemble *réticulé* ou encore de *treillis (lattice)*.

[1] Least upper bound.
[2] Greatest lower bound.

2.2 Les politiques et modèles de contrôle d'accès

Tout d'abord, une politique de sécurité est exprimée en langage naturelle, mais son application nécessite une formalisation au moyen d'un modèle de sécurité pour lever les ambiguïtés sur la spécification et implémenter les mécanismes de sécurité nécessaires. L'un des mécanismes les plus utilisés est le contrôle d'accès (Lampson, 1974).

Le contrôle d'accès définit l'ensemble des opérations qu'une entité du système peut réaliser ou non sur un ensemble de ressources. Les informations décrivant les droits d'accès peuvent être intégrées au système d'information ou stockées à part.

Il existe une panoplie de politiques et de modèles de contrôle d'accès en vue de répondre aux différents besoins en sécurité. Nous présentons, dans ce qui suit, les plus répandues.

2.2.1 Les modèles de contrôle d'accès mandataires (MAC)

Dans les politiques mandataires (MAC), le contrôle d'accès est effectué sous l'égide du système plutôt que des utilisateurs. Le sujet n'a accès à une information que si le système l'y autorise. Les règles d'une politique mandataire diffèrent selon qu'il s'agisse de maintenir des propriétés de confidentialité ou d'intégrité. Les politiques mandataires les plus souvent utilisées sont des politiques multi-niveaux (MLS), qui reposent sur la notion de classes de sécurité. Le modèle de Bell-LaPadula (Bell and LaPadula, 1973), qui a été utilisé pendant longtemps dans les systèmes militaires, est l'un des plus influents pour la confidentialité. Pour l'intégrité, nous citons le modèle de Biba (Biba, 1977), le modèle de Clark et Wilson (Clark and Wilson, 1987) ainsi que le modèle de la Muraille de Chine de Brewer et Nash (Brewer and Nash, 1989).

2.2.1.1 Modèle de Bell-LaPadula

Le modèle de (Bell and LaPadula, 1973) met en oeuvre une politique mandataire multi-niveaux, développée pour le DoD (*Department of Defense*) des Etats-Unis. Le modèle de Bell-LaPadula a pour objectif de contrôler la confidentialité des données et d'éviter la propagation d'information d'un domaine de sécurité supérieur vers un domaine inférieur.

Le modèle se base sur la matrice de contrôle d'accès pour représenter les permissions d'accès du système, et définit en plus des sujets S et des objets O, un treillis de sécurité, noté $(SC, \leq)$ où SC est un ensemble de classes de sécurité et $\leq$ une relation de dominance. Chaque objet a ainsi un niveau de sécurité qui représente le danger que peut constituer la divulgation de l'information contenue dans cet objet, et chaque sujet a une habilitation qui désigne la confiance qui lui est accordée. La notion de dominance entre deux classes sc et sc' est définie pour que sc domine sc', noté $sc' \leq sc$, si le flot d'information de sc' vers sc est autorisé.

Les objectifs de sécurité de cette politique sont les suivants :

- interdire toute fuite d'information d'un objet possédant une certaine classification vers un objet possédant un niveau de classification inférieur ;
- interdire à tout sujet possédant une certaine habilitation d'obtenir des informations d'un objet d'un niveau de classification supérieur à cette habilitation.

Le modèle peut être représenté par une machine à états où chaque état est un triplet $(b, M_{so}, f) \in B \times M \times F$ où :

- $B = S \times O \times A$ est l'ensemble des opérations d'accès en cours où $A = \{$exécuter, lire, ajouter, écrire$\}$;

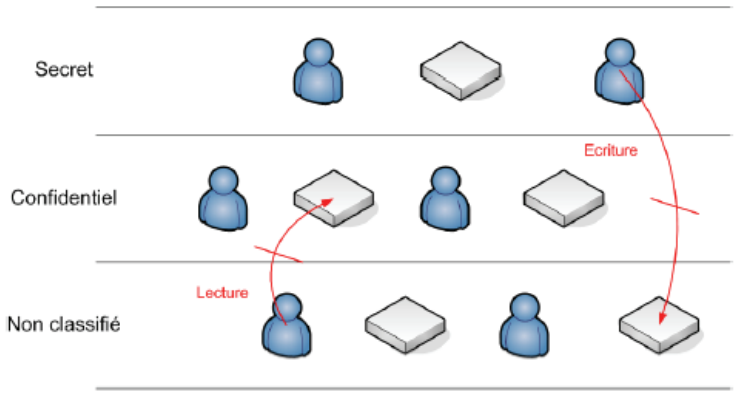


Figure 2.1 No Read Up et No Write Down

- M est l'ensemble des matrices de permissions d'accès $M = (M_{so})_{(s \in S, o \in O)}$. M_{so} associe les objets aux sujets en indiquant les droits qui s'y appliquent.
- $F \subset L_S \times L_C \times L_O$ est l'ensemble des niveaux de sécurité. Dans un état donné, f_S est le plus haut niveau de sécurité qu'un sujet peut avoir ; f_C est le niveau courant de chaque sujet ; f_O est la classification de l'objet.

Les objectifs de sécurité sont obtenus en imposant les deux règles suivantes:

- La propriété simple "***No Read up***": un état est sûr si et seulement si, pour chaque sujet s qui observe un objet o, le niveau de sécurité maximal du sujet domine le niveau de sécurité de l'objet. Formellement, un état satisfait la propriété simple si: $\forall (s, o, a) \in b, a \in \{lire, ecrire\} \Rightarrow f_O(o) \leq f_S(s)$.
- La propriété étoile "***No Write down***": un état est sûr si et seulement si, pour chaque sujet s qui modifie un objet o, le niveau de sécurité de o domine le niveau courant de sécurité de s. Formellement, un état satisfait cette propriété si: $\forall (s, o, a) \in b, a \in \{ajouter, ecrire\} \Rightarrow f_C(s) \leq f_O(o)$.

Toutefois, la politique de Bell-LaPabula présente plusieurs inconvénients. Le plus im-

portant est la dégradation du service provoquée par la surclassification des informations. En effet, le niveau de sécurité d'un objet est augmenté lorsqu'il est accédé en écriture par un sujet de niveau supérieur (car normalement il est interdit de modifier un fichier de niveau inférieur). Ainsi, petit à petit, les niveaux de classification des informations croissent de façon automatique, et il faut les "déclassifier", manuellement par un officier de sécurité ou par un processus dit "de confiance" (*trusted process*) n'obéissant pas aux règles du modèle.

Un article ultérieur de (Mclean, 1987) restitue le modèle Bell-LaPadula de façon plus claire, et accompagnée de critiques concernant des failles qui y apparaissent lorsqu'il est appliqué à un système réel. Il a été démontré qu'il est possible de construire un système, appelé Système Z, qui vérifie bien les deux propriétés, et qui n'est pourtant pas sûr. Le système Z (Mclean, 1987) est un système où un utilisateur de niveau minimal met les niveaux de tous les sujets et de tous les objets au niveau minimal, et autorise l'accès de tous les utilisateurs à tous les objets. Ceci est possible car le niveau d'un objet peut, lui-même, être mémorisé dans un objet ; la valeur de ce dernier peut donc être modifiée par un utilisateur de niveau minimal (puisque l'écriture dans un niveau dominant est autorisée).

2.2.1.2 Modèle de Biba

Le modèle de Bell-LaPadula ne répond qu'à des besoins de confidentialité. C'est pour cette raison que d'autres politiques obligatoires ont été développées pour le maintien de l'intégrité. C'est le cas de la politique proposée par (Biba, 1977). Celle-ci applique à l'intégrité un modèle analogue à celui de Bell-LaPadula pour la confidentialité. À chaque sujet s, on affecte un niveau d'intégrité $is(s)$, correspondant à la confiance qu'on a dans ce sujet et chaque objet o possède un niveau d'intégrité $io(o)$, correspondant au niveau d'intégrité du sujet qui l'a créé.

Les objectifs de sécurité de cette politique visent à :

- interdire toute propagation d'information d'un objet situé à un certain niveau d'intégrité vers un objet situé à un niveau d'intégrité supérieur;
- interdire à tout sujet situé à un certain niveau d'intégrité de modifier un objet possédant un niveau d'intégrité supérieur.

Les objectifs de sécurité sont obtenus en imposant les règles suivantes:

- Un sujet ne peut modifier un objet que si le niveau d'intégrité du sujet domine le niveau d'intégrité de l'objet: $(s, o, modifier) \Rightarrow io(o) \leq is(s)$.
- Un sujet ne peut observer un objet que si le niveau d'intégrité de l'objet domine le niveau d'intégrité du sujet: $(s, o, observer) \Rightarrow is(s) \leq io(o)$
- Un sujet s_i ne peut invoquer un sujet s_j que si le niveau d'intégrité de s_i domine le niveau d'intégrité de s_j: $(s_i, s_j, invoquer) \Rightarrow$ is$(s_j) \leq is(s_i)$

Cela dit, le modèle de Biba présente un inconvénient analogue à celui de Bell-LaPadula: la dégradation des niveaux d'intégrité. Ici l'idée est de migrer un sujet vers un niveau d'intégrité inférieur lorsqu'il accède à un objet de niveau inférieur. L'effet néfaste est que l'ensemble des sujets va rapidement se trouver en bas de l'échelle des niveaux d'intégrité. Dés lors, l'intérêt du modèle Biba est fortement remis en cause, puisqu'il n'y a plus de différence entre les sujets.

2.2.1.3 Modèle de La muraille de Chine

Le modèle de la muraille de chine se compose de trois ensembles :

- C, l'ensemble des compagnies;
- S, l'ensemble des sujets (analystes financiers) ;
- O, l'ensemble des objets (fichiers)

Le système classe l'information en trois niveaux hiérarchiques :

- le niveau le plus bas contient les données de toutes les compagnies ;
- le niveau intermédiaire, regroupe les objets qui concernent la même compagnie;
- le niveau supérieur regroupe les données des compagnies en compétition.

À chaque objet, sont associés le nom de la compagnie à laquelle il appartient ainsi que la classe de conflit d'intérêt qui contient ses données. Ce modèle considère deux fonctions:

- la fonction X, tel que $X(o_i)$ désigne la classe de conflit d'intérêt de o_i. Autrement dit, pour un objet donné, X fournit l'ensemble de toutes les compagnies en compétition autour de cet objet ;
- la fonction Y, tel que $Y(o_j)$ désigne l'ensemble des données de la compagnie de o_j.

Une information est aseptisée si elle a été purgée des détails sensibles et elle n'est pas sujette à des restrictions d'accès. Pour ce type d'information, on a : $X(o) = \emptyset$. Notons qu'un conflit d'intérêt peut surgir, non seulement à cause des objets consultés à un moment donné, mais aussi en raison des accès antérieurs. Pour enregistrer l'historique des actions, le modèle utilise une matrice N_{so} définie par: $N(s, o) = 1$ si s a déjà accédé à o; $N(s, o) = 0$ sinon.

L'objectif de cette politique de sécurité est de garantir qu'aucun utilisateur n'accède simultanément à des données appartenant à des ensembles en conflit d'intérêt. Cet objectif est obtenu en imposant les deux règles suivantes:

- Propriété simple: l'accès est accordé à un sujet seulement si l'objet demandé appartient au même ensemble de données de la compagnie "à l'intérieur de la muraille", ou à une classe de conflit d'intérêt complètement différente.
- Propriété étoile: un sujet s est autorisé à écrire dans un objet o, si l'accès est autorisé par la propriété simple ; et si s ne peut lire aucun objet o' appartenant à un ensemble de données de compagnies différent de celui de o, et contenant des données non aseptisées.

Voici un exemple pour illustrer la deuxième propriété. Supposons que le système gère une banque $Banque_A$, deux compagnies en compétition C_A et C_B, deux sujets (analystes) A_1 et A_2 et trois fichiers qu'on note $Y(C_A)$, $Y(C_B)$ et $Y(Banque_A)$). La propriété étoile n'interdit pas que A_1 ait accès à $Y(C_A)$, $Y(Banque_A)$ et que A_2 ait accès à $Y(C_B)$ et $Y(Banque_A)$. Mais si A_1 lit $Y(C_A)$ et l'écrit dans $Y(Banque_A)$, A_2 peut lire cette information (puisqu'il a accès à $Y(Banque_A)$ alors que cette information (initialement concernant C_A) est dans la même classe de conflit d'intérêt de C_B (à laquelle il a déjà accédé). La propriété simple, seule, ne permet pas d'empêcher de telles attaques, tandis que l'attaque ne peut pas réussir si le système implémente la propriété étoile.

2.2.2 Les modèles de contrôle d'accès basé sur les rôles (RBAC)

De façon générale, un système de politique de contrôle mandataire peut être utilisé aisément dans une administration centrale. Cependant, il est statique et inapproprié pour sécuriser les interactions de diverses organisations indépendantes.

Pour combler ces lacunes, les politiques de contrôle d'accès à base de rôles (RBAC) (Sandhu et al., 1996) ont été introduites. Un rôle est une description des fonctions qu'un sujet a le droit d'accomplir au sein d'une organisation. Ainsi, à chaque rôle est associé un ensemble de permissions. Les sujets ayant reçu l'autorisation de jouer un rôle, héritent alors des permissions associées à ce rôle. Les politiques RBAC s'appliquent bien dans les grandes organisations ou entreprises qui comptent un nombre important d'utilisateurs. Chaque organisation peut avoir sa propre politique interne basée sur les rôles des divers sujets qui lui appartiennent. Initialement, les règles de la politique sont définies en fonction de tous les rôles existants au sein de l'organisation. Par la suite, lorsqu'un nouvel employé est recruté, il suffira de lui assigner les rôles qu'il est censé avoir au sein de l'organisation pour que le système puisse automatiquement cantonner ce nouveau sujet dans les limites autorisées par la politique. Ce type de politique est utilisé au sein des systèmes informatiques de grandes banques européennes (Schaad et al., 2001).

2.2.2.1 Modèle RBAC

Le modèle RBAC, illustré à la figure 2.2, définit différentes composantes nécessaires à l'implantation d'un système de contrôle d'accès. Tel qu'il a été présenté par (Sandhu et al., 1996), RBAC se subdivise en plusieurs niveaux: Core RBAC, RBAC hiérarchique, RBAC avec contraintes de séparation des tâches statique SSD (*Static Separation of Duty*) et dynamique DSD (*Dynamic Separation of Duty*). Core RBAC est le système de base. Il contient l'architecture principale. Les trois autres niveaux affinent le premier niveau en autorisant la hiérarchie des rôles, le contrôle d'accès statique et dynamique.

Core RBAC est défini dans (Sandhu et al., 1996) de la manière suivante:

- *Users*, *Roles*, *Permissions*, *Sessions*, représentent respectivement des ensembles

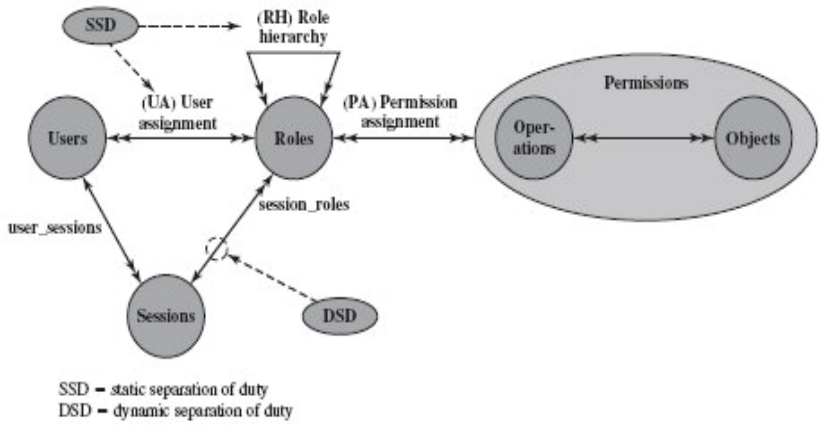


Figure 2.2 Composantes du modèle RBAC

d'utilisateurs, de rôles, de permissions et de sessions ;

- $PA \subseteq Roles \times Permissions$, une relation associant une permission à un rôle ;
- $UA \subseteq Users \times Roles$, une relation associant un ou plusieurs rôles à un utilisateur;
- $user\_sessions : Sessions \rightarrow Users$, une fonction associant chaque session à un seul utilisateur pour toute la durée de vie de la session ;
- $role : Sessions \rightarrow 2^{Roles}$, une fonction associant chaque session s à un ensemble de rôles $role(s)$.

La variante hiérarchique de RBAC permet d'organiser les rôles suivant une hiérarchie partiellement ordonnée, i.e: $RH \subseteq Roles \times Roles$; $r' \leq r$ signifie que r' est un rôle junior de r (r est un rôle sénior); On peut voir RBAC hiérarchique comme un mécanisme d'héritage. Les rôles héritent les autorisations des autres rôles qui leur sont hiérarchiquement inférieurs.

Le modèle RBAC introduit aussi la notion de contrainte pour spécifier des politiques d'autorisation incluant des situations plus restrictives. Ainsi, une contrainte de séparation

statique spécifie que certains rôles, par exemple médecin et infirmier, ne peuvent pas être simultanément affectés à un utilisateur. Une contrainte de séparation dynamique spécifie que, même si certains rôles peuvent être affectés à un même utilisateur, par exemple médecin libéral et chirurgien, ces rôles ne peuvent pas être activés simultanément dans une même session.

Un modèle de contrôle d'accès est généralement constitué de deux parties distinctes : un modèle pour exprimer une politique d'autorisation et un modèle d'administration de cette politique. Dans les modèles RBAC, l'administration des droits est définie dans le modèle ARBAC (*Administration of Role Based Access Control*) (Sandhu and Munawer, 1999). Ce dernier permet de spécifier les rôles administrateurs qui ont la permission d'attribuer de nouveaux rôles aux sujets ou de mettre à jour les permissions associées aux rôles. L'administration des droits dans RBAC est ainsi moins rigide que dans MAC. Toutefois, le modèle reste statique.

D'autres modèles plus récents ont également été proposés tels que les modèles TBAC (*Task Based Authorization Controls*) (Thomas and Sandhu, 1998), TMAC (*Team-based Access Control*) (Thomas, 1997), ORBAC (*Organization Role Based Access Control*) (Kalam et al., 2003), TRBAC (*Temporal Role Based Access Control*) (Bertino et al., 2000) et GTRBAC (*Generalized Temporal Role Based Access Control*) (Joshi, 2003). Ces modèles raffinent le modèle RBAC en introduisant respectivement les notions de tâches, d'équipes, d'organisation et de temps. Nous présenterons, ci-après, certaines de ses extensions temporelles les plus connues, TRBAC (Bertino et al., 2000) et GTRBAC (Joshi, 2003).

2.2.2.2 Modèle TRBAC

Dans les systèmes opérationnels, il est très fréquent que les droits d'accès d'un utilisateur varient selon une contrainte de temps. Par exemple, un employé peut réaliser une opération seulement pendant ses heures de travail. À l'origine, le modèle RBAC ne permet pas ce type de contrainte. Le modèle TRBAC (Bertino et al., 2000) est ainsi, à la base un modèle RBAC auquel on a ajouté des contraintes temporelles liées à l'activation et la désactivation des rôles. Ces contraintes temporelles s'expriment en termes d'intervalle de temps durant lequel la contrainte doit être considérée et en termes de période à l'intérieur de l'intervalle. TRBAC introduit la notion de triggers, qui correspond à des actions exécutées automatiquement en fonction du temps. Ces actions peuvent être la mise à disposition d'un rôle, c'est-à-dire la possibilité d'activer un rôle (*role enabling*), ou, au contraire, l'interdiction d'activer un rôle (*role disabling*) lors d'un intervalle de temps. Lorsque de telles règles sont spécifiées, il est possible d'être confronté à des conflits portant sur l'application de règles contradictoires. C'est pourquoi un système de priorité a été proposé dans (Bertino et al., 2000) pour résoudre ces conflits : à chaque contrainte est donnée une priorité.

Cependant, le modèle TRBAC ne fait aucune distinction entre un rôle rendu disponible et un rôle rendu actif. Un rôle est disponible si les conditions temporelles associées à sa mise en disponibilité sont satisfaites. Alors qu'un rôle est actif s'il est utilisé par un usager. Évidemment seuls les rôles disponibles peuvent être rendus actifs. Cependant, dans le modèle TRBAC, on suppose que le rôle est toujours disponible. Dès qu'un usager autorisé en fait la demande, le rôle lui sera accordé et son état passe à actif. Ainsi, étant donné ces limitations, certaines contraintes ne sont pas exprimables par le modèle TRBAC, notamment le nombre d'activations maximal d'un rôle par un usager dans un intervalle de temps donné.

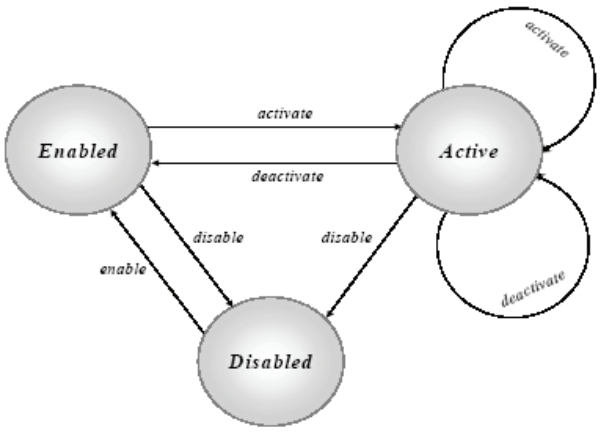


Figure 2.3 Les états d'un rôle dans le modèle GTRBAC

2.2.2.3 Modèle GTRBAC

Le modèle GTRBAC proposé par (Joshi, 2003) est une généralisation de TRBAC en spécifiant plus largement les contraintes temporelles. Le modèle GTRBAC distingue entre différents états d'un rôle. Un rôle peut avoir trois états:

- *Disabled* : le rôle n'est disponible pour aucune session d'utilisation, c'est-à-dire que l'usager ne peut avoir les permissions associées à ce rôle,
- *Enabled* : le rôle est disponible et il peut être activé par tout usager autorisé à utiliser ce rôle au moment de la demande,
- *Active* : cet état indique qu'il y a un au moins un usager qui utilise le rôle durant une session.

Un rôle peut changer d'un état à un autre lorsque des évènements surviennent suite à des actions de l'usager ou de l'administrateur. Les points suivants donnent la spécification des contraintes et événements :

- Contraintes temporelles sur la disponibilité ou non d'un rôle. Une telle contrainte

spécifie un intervalle global de validité et un ensemble de sous-périodes dans cet intervalle pour lesquelles le rôle est mis à disposition,

- Contraintes temporelles sur les assignations usager/rôle et permission/rôle. C'est le même type de contrainte que le précédent, sauf que l'événement généré est une assignation ou déassignation de rôle,
- Contraintes d'activation portent sur l'activation de rôle, par exemple la durée totale pendant laquelle un rôle peut être activé,
- Evénements à l'exécution permettent de prendre des mesures dynamiquement comme l'activation de rôles après un événement,
- Triggers. Ce sont des actions prises en réponse à un événement. Par exemple, $[enable\ r \rightarrow enable\ r']$ signifie qu'un événement, ici, une demande de mise à disposition d'un rôle r, génère un autre événement qui demande la mise à disposition d'un second rôle r' automatiquement.

2.3 Conclusion

Tel que nous pouvons le constater, de nombreux concepts ont été introduits pour structurer les droits d'accès selon l'activité et le rôle des utilisateurs du système. Ces notions sont toutes des intermédiaires entre les utilisateurs et les permissions du système. Toutefois, il est difficile de considérer qu'un modèle de contrôle d'accès est meilleur qu'un autre. Cela dépend essentiellement du domaine d'application et du type de l'organisation qui le met en œuvre. Par exemple, les modèles à niveaux sont bien adaptés aux structures très organisées, où la confidentialité est une priorité. D'autre part, les propositions basées sur les rôles forment la plus grande famille des modèles de contrôle d'accès et sont les plus étudiées du domaine. Leurs extensions aussi s'orientent souvent vers la prise en compte du temps et des contraintes.

Cependant, la définition de nouveaux concepts et relations dans un modèle implique la définition de contraintes s'assurant que la politique est valide. Par exemple, l'interaction entre la notion de hiérarchies et l'exclusion mutuelle induit des propriétés que les politiques doivent respecter. Ainsi, dans tous les cas d'utilisation des modèles de contrôle d'accès, il est nécessaire de s'assurer de la cohérence (ou l'intégrité) de la politique, c,est-à-dire s'assurer que la politique de contrôle d'accès respecte bien le modèle dont elle est une instance.

La vérification d'une politique de sécurité se fait au moyen de techniques et outils appropriés permettant de vérifier que ces exigences seront effectivement satisfaites après une implémentation dans le système cible. Plusieurs modèles et langages ont été proposés pour formaliser les structures, les principes et les propriétés des modèles de contrôle d'accès. Dans le chapitre suivant, nous présentons les cadres existants et les comparerons selon leur expressivité et les applications aux problématiques du contrôle d'accès proposées.

CHAPITRE 3

MODÉLISATION DES POLITIQUES DE SÉCURITÉ

Le problème de formalisation des modèles de contrôle d'accès n'est pas nouveau, il remonte aux années soixante-dix environ où une large partie de la recherche a été faite dans le contexte des systèmes militaires. Dans ce cadre bien particulier, une grande attention a été accordée aux différents aspects formels de la sécurité. Il en est résulté un nombre de modèles formels, de politiques, de mécanismes de même que de méthodes et critères d'évaluation de sécurité (ITSEC, 1991; JCSEC, 1992; CTCPEC, 1993). Depuis les recherches se poursuivent, dans toutes les directions, pour étendre ces aspects à des cadres plus généraux. Un problème principal qui a suscité l'intégration des méthodes formelles est la modélisation et la vérification des politiques de sécurité.

La modélisation ou formalisation des politiques de sécurité offre un cadre logique qui permet une meilleure caractérisation des propriétés qu'elles doivent satisfaire et donne un langage d'expression commun sans ambiguïtés. La formalisation permet également d'utiliser des outils mathématiques pour raisonner sur les modèles et vérifier les propriétés souhaitées. Cette vérification doit être effectuée quelle que soit la politique déterminée par les administrateurs. C'est donc une preuve qui fait abstraction des politiques et qui ne concerne que le modèle de contrôle d'accès lui-même.

Dans ce chapitre, nous allons présenter différents concepts reliés à la modélisation formelle des systèmes. Par la suite, nous allons découvrir davantage les différents travaux effectués dans le domaine de la modélisation des politiques de sécurité, moyennant une recherche bibliographique plus détaillée.

3.1 Notions préliminaires

3.1.1 Ensemble

Soient X et Y deux ensembles. Nous noterons $X \subseteq Y$ si X est un sous ensemble de Y. $|X|$ est le cardinal de X.

3.1.2 Multi-ensembles

Étant donné un ensemble X, un multi-ensemble sur X est une fonction $M : X \rightarrow \mathbb{N}$. Pour tout $x \in X$ nous appellerons $M(x)$ la multiplicité de x dans M. Nous noterons X_{MS} l'ensemble des multi-ensembles sur X. Ainsi, un multi-ensemble est un ensemble qui peut contenir plusieurs occurrences d'un même élément. Un multi-ensemble peut être représenté par la somme formelle : $\sum_{x \in X} M(x) \bullet x$ où l'entier $M(x)$ désigne le nombre d'occurrences de l'élément x dans le multi-ensemble M. Par exemple, la somme $m = 2\bullet < x_1 > +3\bullet < x_2 >$ désigne le mutlti-ensemble qui contient deux éléments de x_1 et trois éléments de x_2.

Soient M_1 et M_2 deux multi-ensembles sur X. Nous définissons les opérations usuelles (+,-, $\leq$, =) sur les multi-ensembles comme suit :

- $M_1 + M_2 = \sum_{x \in X}(M_1(x) + M_2(x)) \bullet x$
- $M_1 \leq M_2$ ssi, $(\forall x \in X, (M_1(x) \leq M_2(x)))$
- $M_1 = M_2$ ssi, $(\forall x \in X, (M_1(x) = M_2(x)))$
- Si $M_1 \leq M_2$ alors $M_2 - M_1 = \sum_{x \in X}(M_2(x) - M_1(x)) \bullet x$
- Le cardinal de M_1, noté $|M_1|$ est $\sum_{x \in X}(M_1(x))$

L'ensemble de tous les multi-ensembles sur X est noté X_{MS} ou encore $Bag(X)$. Nous notons le multi-ensemble vide de la même manière que l'ensemble vide $\emptyset$ ou 0.

3.1.3 Matrices de bornes et graphe de contraintes

3.1.3.1 Matrices de bornes

Une borne est une paire $(c, \prec)$ où $c \in Q \cup \{\infty, -\infty\}$ et $\prec \in \{<, \leq\}$. Nous définissons un ordre totale sur les opérations $<$ et $\leq$, tel que '$<$' $<$ '$\leq$', et sur les bornes comme suit:

- $(c, \prec) = (c', \prec')$ ssi $c = c'$ et $\prec = \prec'$,
- $(c, \prec) < (c', \prec')$ ssi $c < c'$ ou ($c = c'$ et $\prec < \prec'$),
- $(c, \prec) \leq (c', \prec')$ ssi $(c, \prec) = (c', \prec')$ ou $(c, \prec) < (c', \prec')$.

La somme de deux bornes $(c, \prec)$ et $(c', \prec')$, notée $(c, \prec) + (c', \prec')$ est la borne $(c + c', min(\prec, \prec'))$. Le complément d'une borne $(c, \prec)$ est définie comme étant la borne $(c', \prec')$ tel que $(c, \prec) + (c', \prec') = (0, <)$.

Soit $X = \{x_1, \ldots, x_n\}$ un ensemble de variables sur $\mathbb{R}$. La formule F associée à une zone Z sur X est une conjonction de contraintes atomiques sur X. Si nous ajoutons une variable x_0 à l'ensemble X tel que $x_0 = 0$ (x_0 est parfois notée o aussi), chaque contrainte atomique qui apparaît dans F peut être écrite sous forme $x_i - x_j \prec_{i,j} c_{i,j}$ où $c_{i,j} \in Q \cup \{\infty, -\infty\}$, $\prec_{i,j} \in \{<, \leq\}$ et $x_i, x_j \in X \cup \{x_0\}$. La formule F peut être alors représentée par une matrice B d'ordre $n + 1$ telle que $b_{i,j}$ est la borne $(c_{i,j}, \prec_{i,j})$. La matrice B est appelée matrice de bornes (DBM) (Dill, 1990). En général, une zone sur X peut être représentée par une infinité de formules. Cependant, il existe une formule

unique en forme canonique pour chaque zone. L'existence de cette forme, calculable en $O(n^3)$, où $n = |X|$, simplifie considérablement le test d'égalité et d'inclusion entre zones, ainsi que le calcul de l'intersection. La complexité de ces opérations est $O(n^2)$.

3.1.3.2 Graphe de contraintes

Soient Z une zone sur $X = \{x_1, \ldots, x_n\}$ et F sa formule. Le calcul de la forme canonique de Z consiste à mettre chaque contrainte atomique de F dans sa forme la plus restreinte. Ceci revient à remplacer, dans chaque contrainte atomique $x_i - x_j \prec_{i,j} c_{i,j}$ de F, la constante $c_{i,j}$ par la constante la plus petite en valeur absolue, sans affecter le domaine de F. Pour résoudre ce problème, on associe à la formule F une interprétation sous forme de graphe, appelé graphe de contraintes. Le graphe de contraintes G associé à F est un graphe orienté et valué, où les variables de $X \cup \{x_0\}$ sont des sommets (la variable x_0 est représentée par le sommet o) et les contraintes atomiques de F sont des arcs. À chaque contrainte atomique $x_i - x_j \prec_{i,j} c_{i,j}$, est associé un arc du sommet x_j au sommet x_i ayant comme poids la borne $(c_{i,j}, \prec_{i,j})$. Cette borne est interprétée comme la borne supérieure de la distance du sommet x_j au sommet x_i.

Soient F une formule et G le graphe de contraintes qui lui est associé :

- F est consistante ssi, G n'admet aucun cycle négatif,
- Si F est consistante, le poids du plus court chemin du noeud x_j au noeud x_i dans G est égal à $Sup(x_i - x_j, F)$. $Sup(x_i - x_j, F)$ étant le supremum de $(x_i - x_j$ dans le domaine de F.

On peut donc déduire que le calcul de la forme canonique de F revient à calculer le graphe des plus courts chemins sur G. L'algorithme de Floyd-Warshall (Cormen et al., 1990) est une référence dans ce contexte, avec une complexité de $O(n^3)$, où $n = |X|$.

3.1.4 Système de transitions

Un système de transitions est une structure définie par un triplet $(Q, q_0, \rightarrow)$ où Q est un ensemble d'états, $q_0 \in Q$ est l'état initial, et $\rightarrow$ est une relation de transitions entre les états de Q. La notation $q \rightarrow q'$ est utilisée pour spécifier que $(q, q') \in \rightarrow$. Un système de transitions est dit fini si et seulement si $\rightarrow$ est finie. Dans le cas contraire, le système est dit infini. Un système de transitions est dit à branchements infinis si et seulement s'il existe $q \in Q$ tel que $|\{q' \in Q | q \rightarrow q'\}| = \infty$, c'est-à-dire que q a une infinité, éventuellement non dénombrable, de sucesseurs.

3.1.5 Système de transitions temporisé

Les systèmes de transitions temporisées (*Timed Transition System* en anglais, ou TTS) sont des systèmes de transitions particuliers pour lesquels deux types de transitions sont possibles : des transitions d'action et des transitions de temps modélisant respectivement des évolutions discrètes et des évolutions continues du système

Un système de transitions temporisé (TTS) sur un ensemble d'action Σ (ou alphabet) est un quadruplet $(Q, Q_0, \Sigma, \rightarrow)$ où Q est un ensemble d'état, $Q_0 \subseteq Q$ est un ensemble d'états initiaux, Σ est un ensemble fini d'actions (disjoint de $\mathbb{R}_{\geq 0}$), $\rightarrow \subseteq Q \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times Q$ est une relation de transition (ensemble d'arcs). Si $(q, e, q') \in \rightarrow$, nous notons aussi $q \xrightarrow{e} q'$. La relation de transition se décompose en une relation de transition continue $\xrightarrow{dv \in \mathbb{R}_{\geq 0}}$ et une relation de transition discrète $\xrightarrow{a \in A}$.

3.1.5.1 Évolution d'un système de transitions temporisé

Une *évolution* ρ d'un TTS S est une séquence maximale de la forme :
$\rho = q_0 \xrightarrow{dv_0} q'_0 \xrightarrow{a_0} q_1 \xrightarrow{dv_1} q'_1 \xrightarrow{a_1} \cdots q_n \xrightarrow{dv_n} q'_n \cdots$.

Pour une évolution de taille n, nous notons $Untimed(\rho) = a_0 a_1 \ldots$ et $Duration(\rho) = \Sigma_{i=0}^{n} dv_i$.

3.1.5.2 Trace d'une évolution

La trace temporisée de ρ est le mot temporisé $w = (dv_i, a_i)_{0 \leq i \leq n}$. La trace de ρ est le mot $\mu = (e_i)_{0 \leq i \leq n}$.

Un mot temporisé est une trace acceptée par le TTS S s'il existe une évolution de trace w. Le langage temporisé de S désigné par $\mathcal{L}(S)$ est l'ensemble de tous les mots temporisés acceptés par S.

3.1.5.3 Système de transitions temporisé ε-abstrait

Soit $S = (Q, Q_0, \Sigma_\varepsilon, \rightarrow)$ un TTS. Nous définissons le TTS $S^\varepsilon = (Q, Q_0^\varepsilon, \Sigma, \rightarrow_\varepsilon)$ dans lequel les actions ε ont été abstraites de S par :

- $q \xrightarrow{dv}_\varepsilon q'$ avec $dv \in \mathbb{R}_{\geq 0}$ si et seulement si il existe une excution $\rho = q \rightarrow q'$ avec $Untimed(\rho) = \varepsilon^*$ et $Duration(\rho) = dv$.
- $q \xrightarrow{a}_\varepsilon q'$ avec $a \in \Sigma$ si et seulement si il existe une excution $\rho = q \rightarrow q'$ avec $Untimed(\rho) = \varepsilon^* a \varepsilon^*$ et $Duration(\rho) = 0$
- $Q_0^\varepsilon = \{q \mid \exists q' \in Q_0 \mid q' \rightarrow q \ \ et \ \ Duration(\rho) = 0 \wedge Untimed(\rho) = \varepsilon\}$.

3.1.6 Simulation temporelle forte / faible

Soient $S_1 = (Q_1, Q_0^1, \Sigma, \rightarrow_1)$ et $S_2 = (Q_2, Q_0^2, \Sigma, \rightarrow_2)$ deux systèmes de transitions temporisés sur Σ et $\sqsubseteq$ une relation binaire sur $Q_1 \times Q_2$. Nous écrivons $s \sqsubseteq s'$ pour

$(s, s') \in \sqsubseteq$. La relation $\sqsubseteq$ est une relation de simulation temporelle forte de S_1 par S_2 si les trois assertions suivantes sont vérifiées :

1. si $s_1 \in Q_0^1$, alors il existe $s_2 \in Q_0^2$ tel que $s_1 \sqsubseteq s_2$;
2. si $s_1 \xrightarrow{v} s_1'$ avec $v \in \mathbb{R}_{\geq 0}$ et $s_1 \sqsubseteq s_2$ alors il existe $s_2 \xrightarrow{v} s_2'$ tel que $s_1' \xrightarrow{v} s_2'$;
3. si $s_1 \xrightarrow{a} s_1'$ avec $a \in \Sigma$ et $s_1 \sqsubseteq s_2$ alors il existe $s_2 \xrightarrow{a} s_2'$ tel que $s_1' \xrightarrow{a} s_2'$;

Soient $S_1 = (Q_1, Q_0^1, \Sigma_\varepsilon, \rightarrow_1)$ et $S_2 = (Q_2, Q_0^2, \Sigma_\varepsilon, \rightarrow_2)$ deux systèmes de transitions temporisés sur Σ_ε et $\sqsubseteq$ une relation binaire sur $Q_1 \times Q_2$. Nous écrivons $s \sqsubseteq s'$ pour $(s, s') \in \sqsubseteq$. La relation $\sqsubseteq$ est une relation de simulation temporelle faible de S_1 par S_2 si c'est une relation temporelle forte entre ces deux TTS ε-abstraits. Un TTS S_2 simule faiblement S_1 si il existe une relation de simulation faible de S_1 par S_2. Nous notons alors $S_1 \sqsubseteq_W S_2$.

3.1.7 Bisimulation temporelle

Deux TTS S_1 et S_2 sont en relation de bisimulation temporelle forte (respectivement faible) si il existe une relation de simulation forte (respectivement faible)$\sqsubseteq$ de S_1 par S_2 et si $\sqsubseteq^{-1}$ est aussi une relation de simulation forte[1](respectivement faible) de S_2 par S_1. Nous notons alors $S_1 \approx_S S_2$ (respectivement $S_1 \approx_W S_2$).

3.2 Les formalismes de modélisation

La littérature propose de nombreuses techniques de modélisation et de spécification qui montrent une grande diversité tant au niveau du langage utilisé (syntaxe et sémantique)

[1] $S_2 \sqsubseteq^{-1} S_1$ si et seulement si $S_1 \sqsubseteq S_2$

que des domaines d'application, on peut donc citer:
- les approches basées sur la logique à: théorie des ensembles et substitutions généralisées (la méthode B (Abrial, 1996)), logiques temporelles (LTL, CTL, μ-calcul), logiques d'ordre supérieur (λ-calcul).
- les approches algébriques: algèbres de processus (CCS (Milner, 1989), CSP (Brookes et al., 1984), π-calcul, Sπ-calcul).
- les approches par modèles à états tels que les systèmes de transitions comme les automates (Arnold, 1990) et les réseaux de Petri (Reisig, 1985).
- les approches synchrones, les langages à flots de données synchrones (LUSTRE).

Nous présenterons ci-après brièvement quelques approches.

3.2.1 Les approches logiques et la théorie des types

La logique des propositions est l'étude des raisonnements dont la forme est constituée par des variables propositionnelles $(p, q, r, \ldots)$ et des connecteurs logiques tels que : *et* ($\wedge$), *ou* ($\vee$), *non* ($\neg$), *si ... alors ...* ($\Rightarrow$), *si et seulement si* ($\Leftrightarrow$). Toutefois, la logique propositionnelle ne peut pas rendre compte de la totalité des raisonnements. Par exemple, elle ne permet pas de faire allusion à une propriété d'une variable, ni de décrire des relations entre plusieurs variables ... etc. Pour cela, la logique de premier ordre a été introduite (Kleene, 1967). Elle reprend l'ensemble des éléments de la logique propositionnelle et y ajoute des constantes $(a, b, c, \ldots)$, des variables $(x, y, z, \ldots)$, des prédicats (relations), des fonctions $f(x, y, ..), g(y, z, ..), \ldots)$, des quantificateurs universel $\forall$ et existentiel $\exists$, etc. Les termes du langage sont constitués des variables, des constantes et des fonctions appliquées à ces variables ou constantes, i.e.: $t_1 := x$; $t_2 := f(x, y, b)$; $t_3 := a$, etc. Les prédicats $(P(t_1, .., t_n), Q(t_1, .., t_n), \ldots)$ sont des relations qui ont pour arguments les termes du langage. Le langage prédicatif comporte toutes les formules bien formées à partir des formules atomiques $(A := P(t_1, .., t_n), B := (t_1 = t_n), \ldots)$,

des connecteurs binaires de la logique propositionnelle $\wedge, \vee, \neg, \supset, \equiv$ ainsi que des quantificateurs ($\forall$ et $\exists$). Si A et B sont des formules bien formées du langage L, alors $A \wedge B$, $A \vee B$, $\neg A$, $\neg B$, $A \supset B$, $\forall x A$, $\exists x B$, etc., sont des formules bien formées.

La logique modale est une extension de la logique de premier ordre qui a été enrichi par de nouveaux opérateurs. Selon le type de la modalité, diverses logiques modales peuvent être distinguées (Chellas, 1980; Catach, 1989), dont les logiques déontiques et temporelles.

3.2.1.1 La logique déontique

La logique déontique comporte une unique modalité O qui dénote l'obligation. Comme dans toutes les logiques modales, la modalité s'applique à des formules bien formées de la logique propositionnelle. À partir de la modalité d'obligation, on définit communément trois modalités abrégées Per, F et Op, qui désignent respectivement le caractère permis, interdit ou optionnel d'une proposition logique. Per est défini comme le dual de O, F comme la négation de l'obligation, et Op comme le dual de F. Si f est une formule bien formée de la logique déontique, le langage de la logique déontique noté L_O est l'ensemble des formules (ou expressions) construit par les règles suivantes : $a|\neg\mathbf{f}|\mathbf{f} \wedge \mathbf{f}|\mathbf{f} \vee \mathbf{f}|\mathbf{f} \Rightarrow \mathbf{f}|O\mathbf{f}|Per\mathbf{f}|F\mathbf{f}|Op\mathbf{f}$ où a est une proposition atomique de L_O.

3.2.1.2 La logique temporelle

Les logiques temporelles sont parmi les formalismes mathématiques les plus utilisés pour spécifier des propriétés sur les comportements des systèmes. Ces propriétés peuvent être classées de la manière suivante :

- Propriété d'accessibilité (*Reachability property*): indique qu'un état du système

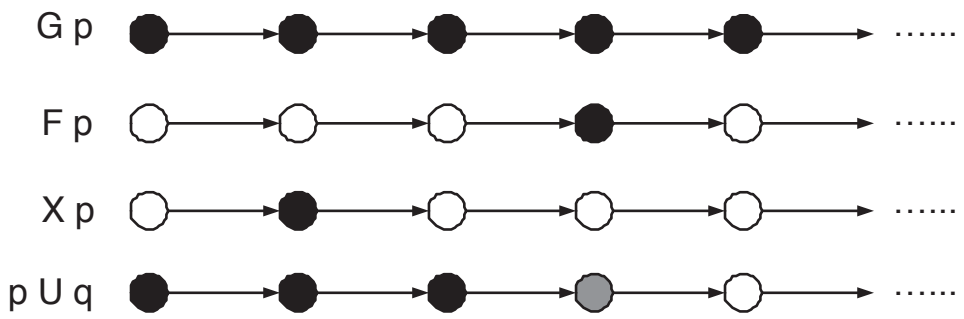


Figure 3.1 Exemple de propriétés temporelles LTL

peut être atteint.

- Propriété de sûreté (*Safety property*): exprime que sous certaines conditions, un événement ne peut jamais se produire. Notons que la négation d'une propriété d'accessibilité est une propriété de sûreté.

- Propriété de vivacité (*Liveness property*): exprime que sous certaines conditions, un événement finira par se produire.

- Propriété d'absence de blocage (*Deadlock free property*): exprime que le système ne se retrouvera jamais dans une situation où il ne pourra plus progresser.

- Propriété d'équité (*Fairness property*): exprime que sous certaines conditions, un événement se produira (ou ne se produira pas) infiniment souvent.

Il existe plusieurs extensions des logiques temporelles qui se distinguent selon deux axes:
- La logique temporelle linéaire LTL (*Linear Temporal Logic*) qui considère l'ensemble des exécutions d'un système comme des séquences distinctes (Figure 3.1).
- La logique temporelle arborescente CTL (*Computation-Tree Logic*) qui représente l'ensemble des exécutions comme un arbre où les différents successeurs d'un état sont obtenus par les instants d'évènements possibles en cet état (Figure 3.2).

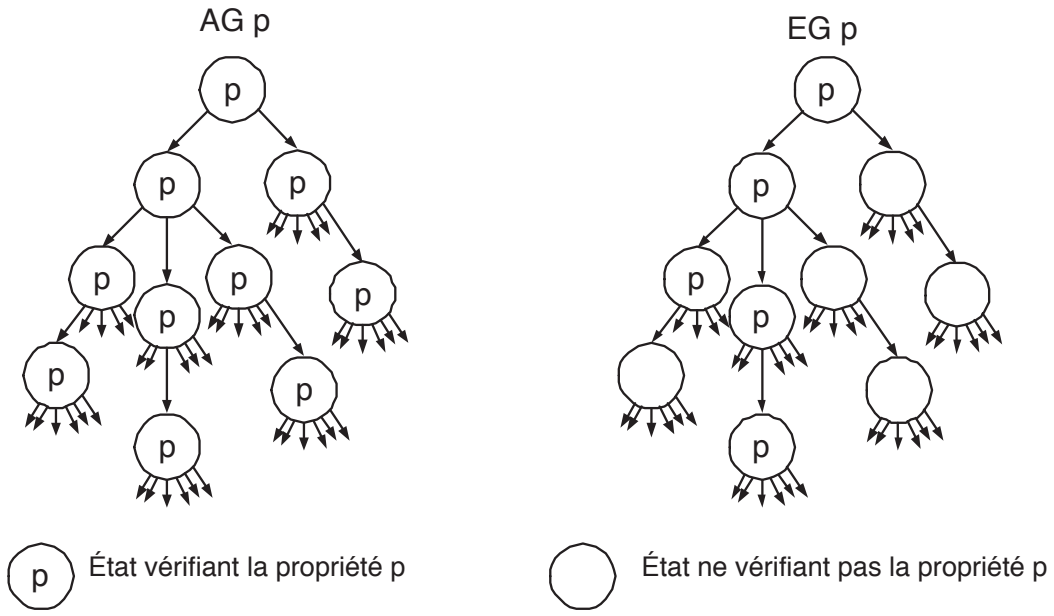


Figure 3.2 Les quantificateurs de chemins

La différence entre ces logiques temporelles provient de l'ensemble d'opérateurs temporels qui peut être utilisé et des objets sur lesquels ils sont interprétés (séquences ou arbres d'états). Ces logiques ont un ensemble d'opérateurs temporels communs qui sont interprétés sur des séquences d'états, tels que G (Toujours "*Always*"), F (Fatalement "*Eventually*"), X (Suivant "*Next*"), U (Jusqu'à "*Until*") et l'opérateur W (À moins que "*Weakness*"). Tandis que les logiques temporelles arborescentes utilisent en plus, les quantificateurs de chemins : le quantificateur A (Quelque soit) et le quantificateur E (Il existe) pour exprimer l'aspect arborescent des propriétés. Les logiques temporelles utilisent des propositions P pour qualifier les états et des opérateurs temporels qui permettent d'exprimer des propriétés sur les enchaînements d'états (exécutions). Une formule propositionnelle combine des propositions sur les états et des connecteurs logiques classiques comme la conjonction et la négation.

3.2.1.3 La théorie des types

Les logiques d'ordre supérieur sont des logiques formelles qui étendent le calcul des prédicats du premier ordre en permettant d'utiliser les variables dans les termes en tant que fonctions, et dans les expressions en tant que prédicats. La théorie des types simples est une formalisation de la logique d'ordre supérieur dont le langage est construit sur le λ-calcul simplement typé plutôt que sur le langage des théories du premier ordre.

Le langage λ a été inventé par le logicien américain Alonzo Church dans les années 1930 (Church, 1936). La syntaxe du λ-calcul simplement typé est définie par l'ensemble des expressions formées de la façon suivante : $e ::= \lambda x.e|e_1e_2|x$. En supposant l'existence d'un ensemble infini de variables notées x, y, z, etc. Les trois constructions principales du λ-calcul sont:

- Si x est une variable et e est un λ-terme déjà construit, alors $\lambda x.e$ est un λ-terme, appelé **abstraction**. Intuitivement, $\lambda x.e$ est la fonction qui envoie x vers e.
- Si e_1 et e_2 sont des λ-termes déjà construits alors e_1e_2 (la juxtaposition des termes) est un λ-terme, appelé une **application**.
- Si x est une variable, alors c'est également un λ-terme.

Dans le terme $\lambda x.e$, la variable x peut bien sûr apparaître dans e. On peut remplacer x par une autre variable y, à condition de remplacer toutes les occurrences de x par y, et à condition que y n'apparaisse pas déjà dans e. La nouvelle expression représente alors la même fonction. On dit que les deux expressions sont α**-équivalentes**. Deux termes sont donc égaux s'ils sont α-équivalents.

Une opération de **substitution** consiste à remplacer toutes les occurrences d'une variable libre par un λ-terme. Intuitivement, une variable x est libre dans un terme si et

seulement si cette variable n'apparaît sous aucune expression de la forme $\lambda x.e$. Dans la substitution, on note $e[e'/x]$ le terme obtenu en remplaçant toutes les occurrences libres de x par e' dans e.

La sémantique opérationnelle dans le λ-calcul repose sur la notion de β**-réduction** dans les λ-termes, basée sur la substitution. L'intuition de cette réduction est la suivante : toute fonction appliquée à un argument peut être déroulée. Ce comportement se décrit en définissant une relation binaire notée $\rightsquigarrow$ de la façon suivante: $(\lambda x.e)e' \rightsquigarrow e[e'/x]$.

3.2.2 Les algèbres de processus

C'est une méthode qui permet de spécifier et de réfléchir à propos des systèmes communicants. Elle est constituée d'un ensemble: de termes, d'opérateurs et d'axiomes, qui sont utilisés pour écrire et manipuler des expressions algébriques. La force principale des algèbres de processus réside dans le fait qu'elles peuvent modéliser des systèmes d'une certaine complexité à l'aide d'un nombre très restreint d'opérateurs, qui peuvent décrire, par exemple, le parallélisme ou la communication entre sous-processus. Nous en introduirons ci-après quelques unes, notamment les plus connues.

3.2.2.1 CCS (*Calculus of Communicating Systems*)

CCS est une algèbre de processus proposée par Milner (Milner, 1980) pour formaliser la notion de programmation concurrente. Un système (ou programme) est décrit comme un ensemble d'agents concurrents (c'est-à-dire s'exécutant indépendamment les uns des autres) et interagissant via un mécanisme de communications synchrones (les émissions et réceptions de messages sont bloquantes). Les agents communiquent les uns avec les autres en utilisant des ***ports de communication***, qui sont identifiés par un ***nom***. Dans la syntaxe d'un processus, le ***nom a*** désigne l'action de recevoir un message sur le port

a tandis que le ***co-nom*** $\bar{a}$ désignera l'action d'émettre un message sur le port **a**. Son mécanisme d'interaction de base est la synchronisation par rendez-vous (mais de nombreuses extensions existent).

La construction élémentaire d'un agent est la mise en séquence d'actions: pour une action α (α étant une émission $\bar{a}$ ou une réception a), αP désigne un agent qui effectuera l'action α et poursuivra son exécution comme P. Ainsi, si un agent $\bar{a}.P$ propose une émission via le port a et l'agent $a.Q$ une réception pour ce même port, alors une interaction peut se produire entre ces deux agents: la synchronisation a lieu et les deux agents poursuivent leur exécution comme P et Q respectivement. Hormis la composition séquentielle d'actions, CCS dispose de bien d'autres caractéristiques (Talbot, 2005).

Formellement, la sémantique opérationnelle de CCS peut être définie comme un système de transitions Étiquetées (LTS - *Labeled Transition System*) qui décrit les interactions d'un agent avec son environnement (d'autres agents). CCS est utilisée pour l'analyse de sûreté et de viabilité de protocoles ou de programmes distribués. Plus récemment, elle a été utilisée pour l'analyse de propriétés de sécurité. Il existe une multitude d'algèbres de processus en plus de CCS, et elles permettent toutes de modéliser mathématiquement des systèmes parallèles ou communicants. Les plus connues sont CSP (*Communication Sequencial Processes*) par (Hoare, 1978) où la communication peut se faire entre plusieurs processus (ne se limite pas à deux processus) et ASP (*Algebra of Communicating Processes*) par Bergstra et Klop (Bodei et al., 1999). Plusieurs autres algèbres de processus ont été développées dans les années 90 et dans les dernières années, notamment le TCCS (*Timed* CCS) (Chen et al., 1990) qui ajoute la notion du temps et le π-calcul.

3.2.2.2 Le π-calcul

Introduit par Milner et Parrow (Milner et al., 1993; Parrow, 2001). Ce calcul se base principalement sur CCS, sans toutefois en être une extension. Le π-calcul améliore considérablement l'aspect communication, déjà présent dans CCS, en permettant le passage non seulement de valeur, mais également de canal de communication, voire même de processus entier. Les noms de ports, appelés canaux dans le π-calcul sont désormais, des valeurs possibles de la communication. Cette nouveauté amène une notion de mobilité qui n'était pas présente dans le CCS.

Initialement, la sémantique du π-calcul était définie comme celle de CCS par un système de transitions étiquetées. Par la suite, (Milner, 1980) a proposé une sémantique opérationnelle alternative pour le π-calcul. Cette sémantique se base sur deux relations définies sur l'ensemble des processus : *la congruence structurelle* et *la relation de réduction*, notées respectivement $\equiv$ et $\rightarrow$. Brièvement, on dit que le processus $(vn)(P \mid Q)$ est congru à $((vn)P) \mid Q$ si n n'apparaît pas (libre) dans Q, c'est-à-dire si Q ne connaît pas le secret n. $(vn)P$ désigne un processus dont le nom n est restreint et lié au processus P par le lien (v). La relation de réduction traduit notamment l'interaction par communication entre processus, correspondant ainsi aux τ-transitions. Ainsi, la règle de communication pour deux processus voulant interagir via le canal x est donnée par: $\bar{x}(y).P \mid x(z).Q \rightarrow P \mid Q\{z \leftarrow y\}$. Le processus $Qz \leftarrow y$ désigne le processus Q dans lequel toutes les occurrences libres du nom z ont été remplacées par y en prenant garde que toutes les occurrences remplacées demeurent des occurrences libres de y.

Bien qu'offrant de grandes possibilités pour la modélisation de systèmes concurrents, le π-calcul a néanmoins quelques défauts : la notion de concurrence est souvent associée à celle de distribution, c'est-à-dire à des agents ou processus s'exécutant sur un ensemble de sites distincts, sites pouvant être des machines ou simplement des processeurs. Le π-calcul n'offre cependant pas la possibilité de modéliser la notion de distribution.

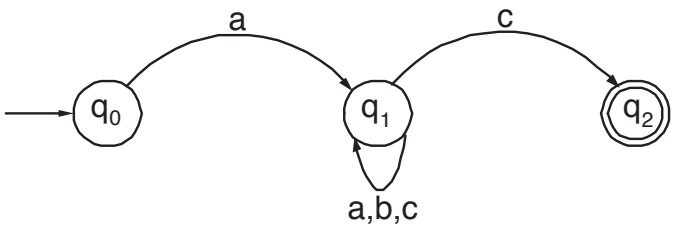


Figure 3.3 Automate fini

Ainsi, des extensions du π-calcul ont été proposées (Amadio and Prasad, 1994; Riely and Hennessy, 1998).

3.2.3 Les systèmes de transitions

Intuitivement, un système de transitions (Arnold, 1990), modélisant une application, est constitué de l'ensemble des états ou situations dans lequel peut se trouver l'application et de l'ensemble des transitions entre états modélisant l'évolution de l'application. Il en existe un très grand nombre, presque tous des extensions des automates à états finis ou des réseaux de Petri respectivement introduits en 1955 et 1962.

3.2.3.1 Les automates à états finis

Les automates à états finis sont représentés graphiquement par des diagrammes de transition d'états, graphes orientés dont les nœuds sont des *états* et les arcs des *transitions*. La figure 3.3 représente un automate qui reconnaît les mots définis sur l'alphabet $\{a, b, c\}$ qui commencent par a et qui finissent par c.

Un état est un ensemble de valeurs qui caractérise le système à un moment donné dans le temps. Une transition d'état est une relation entre deux états indiquant un changement d'état possible, et qui peut être annotée pour indiquer les conditions et les sources de

déclenchement (événements) et les opérations qui en résultent (sorties).

3.2.3.2 Les réseaux de Petri

Les réseaux de Petri (Petri, 1962) sont une généralisation des automates à états. Ils offrent un contexte général pour modéliser la concurrence et la synchronisation dans les systèmes distribués. Un réseau de Petri est un graphe biparti alterné qui possède deux types de nœuds : les places (cercles) et les transitions (rectangles). Des arcs (flèches) relient les places aux transitions (figure 3.4). L'état du système, nommé marquage, est défini par la répartition des jetons dans les places. Une transition est franchissable sous certaines conditions, notamment lorsqu'il y a suffisamment de jetons dans ses places d'entrée. Le franchissement d'une transition se traduit par une modification du marquage consistant en la consommation des jetons indispensables au franchissement de la transition et la création éventuelle de nouveaux jetons dans les places en sortie de la transition.

Définition 3.2.1 ***(Réseau de Petri simple)***
Un réseau de Petri simple est un tuple $(P, T, Pre, Post, M_0)$*:*

- P *est un ensemble fini de places,*
- T *est un ensemble fini de transitions (*$P \cap T = \emptyset$*),*
- Pre*:* $P \times T \rightarrow N$ *est la fonction d'incidence avant,*
- $Post$*:* $P \times T \rightarrow N$ *est la fonction d'incidence arrière,*
- M_0 *est le marquage initial* $M_0 : P \rightarrow N$.

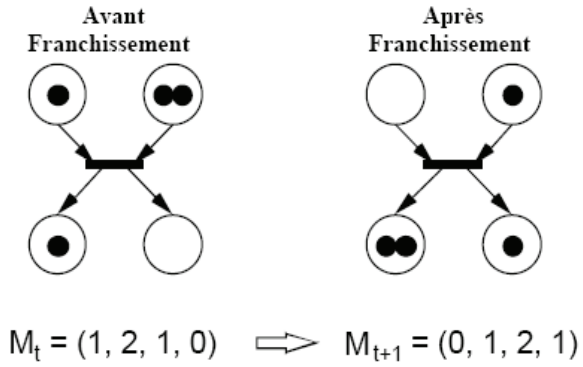


Figure 3.4 Opération de franchissement

Un réseau de Petri peut être vu comme un système de transitions dont les états sont les marquages du réseau et les transitions entre états correspondent au franchissement des transitions du réseau.

3.2.3.2.1 Propriétés des réseaux de Petri Il existe un certain nombre de propriétés qui ont été définies pour les réseaux de Petri, à savoir, le caractère borné, la réinitilisation, la vivacité, la conservation, la terminaison (Diaz, 2001). Certaines de ces propriétés sont dites propriétés dynamiques car elles dépendent du marquage initial et sont liées à l'évolution du réseau, alors que d'autres sont dites propriétés statiques du fait qu'elles soient liées à la typologie du réseau et indépendantes du marquage initial.

Définition 3.2.2 ***(Réseau de Petri borné)***
Une place P_i est bornée pour un marquage initial M_0 si pour tout marquage accessible à partir de M_0, le nombre de jetons dans P_i reste borné. Elle est dite k-bornée si le nombre de jetons dans P_i est toujours inférieur ou égal à k. Un RdP est (k) borné si toutes ses places sont (k) bornées.

Un RdP peut ne pas être borné. Sur l'exemple représenté à la figure 3.5, la transition T_1 admet la place P_1 comme unique place d'entrée. La place P_1 a un jeton : la transition T_1

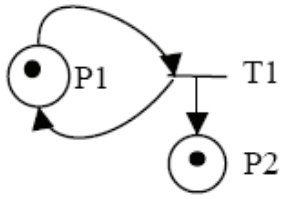


Figure 3.5 Réseau de Petri non borné

est franchissable. Comme P_1 est aussi place de sortie de T_1, le franchissement de T_1 ne change pas le marquage de P_1. La transition T_1 est donc franchissable en permanence et peut donc être franchie un nombre de fois infini. Chaque franchissement de T_1 ajoute un jeton dans P_2 dont le marquage va donc tendre vers l'infini.

Définition 3.2.3 ***(Réseau de Petri sauf)***

Un RdP est sauf ou binaire pour un marquage initial M_0 s'il est 1-borné.

Définition 3.2.4 ***(La vivacité)***

Une transition T_j est vivante pour un marquage initial M_0 si pour tout marquage accessible M_k, il existe une séquence de franchissement S à partir de M_k contenant T_j: $M_k \in^* M_0, \exists S, M_k \mid S >$ *et* $S = \ldots T_j \ldots$

Si une transition T_j est vivante alors, à tout instant, on sait que T_j peut être franchie dans le futur. Dans le cas d'un réseau de Petri modélisant un système fonctionnant en permanence, si une transition n'est pas vivante et si une fonction du système est associée au franchissement de cette transition, cela veut dire qu'à partir d'un certain instant, cette fonction ne sera plus disponible dans le futur, ce qui peut traduire une erreur ou une panne.

Définition 3.2.5 ***(Blocage)***

Un blocage (ou état puits) est un marquage pour lequel aucune transition n'est validée.

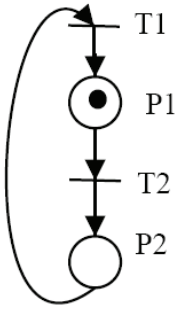


Figure 3.6 Réseau de Petri vivant

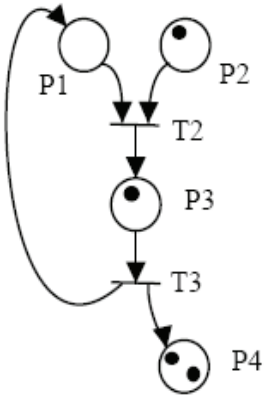


Figure 3.7 Exemple de réseau de Petri bloquant

Un réseau de Petri est dit sans blocage pour un marquage initial M_0 si aucun marquage accessible n'est un blocage.

Le réseau de Petri de la Figure 3.7, par exemple, a pour blocage le marquage : $M = (1, 0, 0, 1)$.

Définition 3.2.6 ***(États d'accueil et Réseau de Petri réinitialisable)***
Un RdP a un état d'accueil M_a pour un marquage initial M_0 si pour tout marquage accessible M_k à partir de M_0, il existe une séquence de franchissement permettant d'atteindre le marquage M_a: $\forall M_k \in^* M_0, \exists S_j, M_k | S_j > M_a$

Un RdP est réinitialisable pour un marquage initial M_0 si M_0 est un état d'accueil

Si un réseau de Petri présente un état d'accueil, il est facile de vérifier s'il est sans blocage et d'étudier sa vivacité.

3.2.3.2.2 Les réseaux de Petri colorés Les réseaux colorés (Jensen, 1997) ont été introduits afin de modéliser des systèmes complexes tout en gardant les possibilités de vérification. Lorsque le nombre d'entités du système à modéliser est important, la taille du réseau de Petri devient rapidement énorme, et si les entités présentent des comportements similaires, l'usage des réseaux colorés permet de condenser le modèle. En effet, une couleur est une information attachée à un jeton. Cette information permet de distinguer des jetons entre eux et peut être de type quelconque. Par conséquent, une place peut contenir des jetons de différentes couleurs et une transition peut être franchie de différentes manières, selon la couleur. Ceci est réalisé en attachant un domaine de couleur à chaque place et à chaque transition. Ainsi, les arcs ne sont pas seulement étiquetés par le nombre de jetons mais aussi par leurs couleurs. Le franchissement d'une transition est alors conditionné par la présence dans les places en entrée du nombre de jetons nécessaires, qui en plus satisfont les couleurs qui étiquettent les arcs. Après le franchissement d'une transition, les jetons qui étiquettent les arcs d'entrée sont retirés des places en entrée tandis que ceux qui étiquettent les arcs de sortie sont ajoutés aux places en sortie de cette transition.

Ainsi, pour un même système, le nombre de comportements qui peuvent être exprimés par un réseau coloré est nettement plus élevé qu'avec un réseau simple. Ce sont des réseaux très adaptés aux architectures distribuées. D'autant plus qu'à tout réseau coloré correspond un réseau de Petri simple qui lui est isomorphe. Ceci permet donc d'exploiter les mêmes techniques d'analyse que celles développées pour les réseaux simples en plus d'autres qui ont été complétées et adaptées aux réseaux colorés telle que le support de la

hiérarchisation.

Définition 3.2.7 ***(Réseau de Petri colorés)***

Un réseau de Petri coloré (CPN) est un tuple $(\Delta, P, T, Arc, N\text{œ}ud, Couleur, Garde, E, M_0)$ *où:*

- Δ *est un ensemble de domaines de couleurs (chaque domaine est un ensemble fini et non vide).*
- Arc *est un ensemble fini d'arcs tel que* $P \cap Arc = T \cap Arc = \emptyset$
- $N\text{œ}ud$ *est la fonction nœud,* $N\text{œ}ud : Arc \rightarrow P \times T \cup T \times P$.
- $Couleur : P \rightarrow PowerSet(\Delta)$. $Couleur(p)$ *est la fonction couleur qui associe à chaque place un domaine de couleur.*
- $Garde$ *est une garde, qui fait correspondre à chaque transition une expression booléenne. Les variables de la garde appartiennent à* Δ.
- E *est l'application qui associe à chaque arc, un élément de* $Couleur(p)_{MS}$ *où* p *est une place appartenant à l'arc.* E *indique le nombre de jetons colorés à recevoir de la place qui se trouve en entrée de la transition, et le nombre de ceux à produire dans la place qui se trouve en sortie.*
- M_0 *est l'application qui associe à chaque place* p*, un élément de* $Couleur(p)_{MS}$. $M_0(p)$ *indique la distribution initiale des jetons colorés dans la place* p.

De manière générale, un marquage M d'un réseau coloré est une application qui associe à chaque place p, un élément de $Couleur(p)_{MS}$. $M(p)$ est un multi-ensemble sur $Couleur(p)$ qui indique les marques colorées présentes dans la place p au marquage M. L'état du modèle est défini par un marquage coloré.

Tableau 3.1 Application des réseaux de Petri

Réseau de Pétri ordinaire	Modélisation de systèmes logiciels Modélisation de processus d'affaires Gestion des flux Programmation concurrente Génie de la qualité Diagnostic
Réseau de Pétri généralisé	Gestion des flux complexes Modélisation de chaînes logistiques Utilisation pour les techniques quantitatives
Réseau de Pétri temporisé	Gestion du temps Modélisation d'attentes
Réseau de Pétri coloré	Modélisation des systèmes de collaboration
Réseau de Pétri continu	Modélisation de réactions chimiques

3.2.3.2.3 Quelques outils de simulation Il existe de nombreux outils de simulation, souvent libres, et développés dans le cadre de thèses ou de recherches scientifiques.

- **TINA** : un des logiciels les plus utilisés, simulation de réseaux de Pétri temporelles.

- **CPN Tools** : un des logiciels les plus utilisés, simulation de réseaux de Pétri colorés.

- **MISS-RdP**© (Interactive Modelling and System Simulation) : un outil permettant de simuler et de modéliser des RdP de haut niveau.

- **Petri-Parc**: un simulateur graphique de RdP qui permet la modélisation et la simulation de réseaux simples ou colorés.

- **CoopnBuilder** : un simulateur graphique utilisé pour la modélisation des réseaux de Pétri orientés-objets.

3.2.3.2.4 Utilisation actuelle [2]

L'utilisation des réseaux de Pétri est plus tournée vers la recherche que vers l'industrie. Le tableau 3.1 montre les domaines d'application de différents types de réseaux de Petri.

[2] Source: http://www.benjamin-monteil.com/ecole/petri.htm.

3.3 Modélisation des politiques de sécurité

3.3.1 Modélisation en langage logique

Les travaux sur la modélisation de politiques de sécurité en langage logique sont très nombreux, notamment dans le contexte des politiques de type RBAC.

Dans (Kalam et al., 2003), les auteurs définissent un langage basé sur la logique du premier ordre pour représenter la politique de sécurité Or-BAC. Ensuite, dans (Kalam and Deswarte, 2003), les auteurs proposent un autre formalisme pour raisonner sur Or-BAC fondé sur la logique déontique. Leur approche est appliquée aux Systèmes d'Information et de Communication en Santé et social (SICSS). Afin de faciliter la conception et la manipulation de politiques fondées sur Or-BAC, ils soulignent qu'il serait souhaitable d'utiliser une représentation graphique dont l'intérêt pratique peut s'avérer significatif si cette représentation est supportée par un outil d'édition.

Dans (Barker and Stuckey, 2003), les auteurs proposent d'utiliser la programmation logique par contraintes (la logique des prédicats à laquelle est ajoutée la notion de contrainte linéaire) pour la modélisation des règlements de contrôle d'accès. Leur proposition permet la modélisation de politiques fermée (tout ce qui n'est pas explicitement permis est interdit), de type ouverte (tout ce qui n'est pas explicitement interdit est permis) ou hybride. Ils étudient en particulier le modèle RBAC en considérant la gestion de la hiérarchie des rôles et le temps dans les autorisations pour se rapprocher du modèle TRBAC. Cependant, la vérification repose sur des algorithmes qui ne permettent pas toujours une traçabilité efficace de l'inférence.

(Jagadeesan et al., 2006) présente une formalisation du modèle RBAC au moyen du λ-calcul. Le modèle résultant est noté λ-RBAC. La syntaxe et la sémantique opérationnelle du λ-calcul typé ont été adaptées au contexte RBAC.

Les modèles de contrôle d'accès ont également été décrits par d'autres méthodes de spécification formelle telles que Z (PRG, 1980; Spivey, 1992) et ALLOY (Jackson, 2004). Le langage Z appliqué surtout dans le cadre de projets d'informatique industrielle, est un langage mathématique formel, orienté modèle qui se base sur la théorie des ensembles de Zermelo-Fränkel et sur la logique de premier ordre. (Yuan et al., 2006) a développé un modèle formel avec Z pour décrire le modèle RBAC. Une spécification Z type est un ensemble de schémas d'état et d'opérations. Le schéma d'état contient une partie permettant la déclaration de variables (*Users* : P*USERS*, *Roles* : P*ROLES*, *Ops* : P*OPS*. . .) et une autre, prédicative, définissant les contraintes sur ces variables (dom *assigned roles* = *Users* $\cup$ (ran *assigned roles*) $\subseteq$ *Roles*). Les relations entre un état avant et un état après sont définies par les schémas d'opérations. Ainsi, (Yuan et al., 2006) définit l'espace d'état du modèle RBAC et les opérations possibles qui transforment un état à un autre sous certaines contraintes.

(Zao et al., 2002) propose une spécification du modèle RBAC en logique d'ALLOY. ALLOY (Jackson et al., 2000) est un outil qui offre à la fois un langage et un outil de vérification et de validation de modèles formels. Il a été développé par le groupe de recherche *Software Design Group* dirigé par Daniel Jackson du MIT (*Massachusetts Institute of Technology*). Il permet de modéliser les systèmes afin de les simuler, les vérifier et valider certaines propriétés. Un modèle ALLOY est un modèle formel et abstrait qui est aussi, à la fois un modèle analysable et testable. Le langage possède une syntaxe simple basée sur le langage Z (Spivey, 1992). Les propriétés d'un modèle ALLOY peuvent être vérifiées, et le modèle peut être simulé avec l'analyseur ALLOY (ALLOY Analyser) (Jackson et al., 2000).

Le modèle résultant de (Zao et al., 2002) à base de ALLOY sert éventuellement comme prototype pour vérifier l'exactitude de différentes implémentations de RBAC d'une part, et la consistence des relations et contraintes d'autre part, dans le but d'obtenir une instance plausible. Dans un contexte sensiblement similaire, (Mankai, 2005) présente dans

son mémoire de maîtrise, une méthode basée sur la modélisation en logique de premier ordre pour analyser et détecter les interactions ainsi que les conflits présents dans un ensemble de politiques de contrôle d'accès exprimées en XACML. Le modèle logique obtenu est traduit vers le langage ALLOY et ensuite soumis à l'analyseur ALLOY pour détecter les différentes conflits et incohérences potentiels.

3.3.2 Modélisation en algèbre de processus

D'autres approches s'orientent vers l'intégration des algèbres de processus dans l'expression des modèles de contrôle d'accès à base de rôles. L'intérêt de ces approches est motivé par la capture des comportements dynamiques des modèles RBAC.

(Braghin et al., 2004) propose une extension du π-calcul pour décrire les modèles RBAC. L'extension consiste à intégrer la notion d'utilisateurs représentés par des processus, deux nouvelles réductions pour exprimer l'activation et la désactivation des rôles et de nouvelles façons pour exprimer l'assignation des permissions aux rôles. Le terme $r|P|_{\rho}$ représente une session de l'utilisateur appelé r, s'exécutant via un processus P et ayant l'ensemble de rôles actifs ρ. Les nouvelles réductions sont exprimées comme suit: $r|roleR.P|_{\rho} \mapsto r|P|_{\rho \cup R}$ et $r|yieldR.P|_{\rho} \mapsto r|P|_{\rho - R}$

Intuitivement, quand un utilisateur r active un rôle R pendant une session, R doit être ajouté à l'ensemble de rôles ρ activé, et le restant de la session P sera exécuté avec l'ensemble ρ mis à jour. Vice-versa pour la désactivation du R.

(Lu et al., 2006) propose de formaliser l'administration des rôles dans les modèles RBAC au moyen du π-calcul. Les auteurs présentent d'abord un nouveau modèle d'administration DARBAC "*Domain Administration of RBAC*" qui introduit la notion de domaine d'administration. Chaque rôle administratif est affecté à un domaine d'administration et ne peut exécuter que les opérations administratives spécifiques au domaine auquel

il est rattaché. Une approche basée sur le π-calcul est par la suite proposée pour formaliser le modèle DARBAC. Chaque élément du modèle DARBAC est représenté par son équivalent dans le π-calcul. Certains éléments du modèle DARBAC sont considérés des processus (les usagers, les rôles, les objets et les opérations administratives), tandis que d'autres sont représentés par des canaux (les opérations sur les objets, la hiérarchie des rôles, les objets administratives, les opérations d'affectation des rôles aux usagers et des permissions aux rôles). Les interactions et la communication entre les différents éléments se fait via des ports. (Lu et al., 2006) reconnaît au π-calcul sa grande puissance d'expression qu'ils ont tenté de transférer au domaine des RBACs. Toutefois, (Lu et al., 2006) ne présente aucune approche d'analyse, mais souligne que le π-calcul se base sur un nombre restreint d'opérations et offre des techniques de preuve fondées sur la bisimulation et la congruence qui pourraient être intéressantes pour vérifier le modèle DARBAC.

3.3.3 Modélisation à l'aide des automates

Dans (Schneider, 2000), l'auteur fournit un formalisme appelé les automates de sécurité. Celui-ci se base sur les automates et permet de spécifier des politiques de sécurité pouvant être mises en oeuvre par des mécanismes de surveillance d'exécution d'une application. C'est une approche dynamique à la sécurité des applications. Une politique de sécurité selon Schneider définit les exécutions (d'une application) qui pour une raison ou une autre ont été jugées inacceptables. Cette définition englobe aussi bien les politiques de sécurité dites d'ordre général (le contrôle d'accès aux informations sensibles, le contrôle de flux d'information, la disponibilité de service), que les politiques plus spécifiques, ayant pour but de décrire un comportement jugé correct d'une application donnée (par exemple une politique qui interdit à une application d'envoyer des données sur le réseau après avoir lu un fichier). Schneider considère que les politiques de sécurité correspondent à des propriétés de sûreté définies pour des ensembles

de traces. Il propose d'utiliser les automates de sécurité comme formalisme pour définir des politiques de sécurité. On obtient alors une classe d'automates qui définissent les séquences d'actions légales qu'un système peut exécuter. L'exécution d'un système est ainsi surveillée de sorte qu'une action prête à être exécutée puisse être empêchée si elle est considérée illégale par l'automate de sécurité.

Une proposition récente (Mondal and Sural, 2008) rejoint notre travail (Rakkay and Boucheneb, 2009) au sujet de la modélisation des modèles RBAC. (Mondal and Sural, 2008) proposent une représentation modulaire d'un modèle RBAC temporel au moyen des automates temporisés. La construction se base sur l'outil Uppaal (Behrmann et al., 2004). Chaque rôle est représenté par un automate temporisé qui modélise les différents états d'un rôle (*Disabled*, *Enabled*, *Activated*) ainsi que les actions qui permettent de passer d'un état à un autre. Chaque sujet est représenté par un automate temporisé qui exprime la relation d'assignation d'un rôle à un sujet. Chaque permission est représenté par un automate temporisé pour exprimer l'assignation des permissions aux rôles. Finalement, un automate contrôleur est construit pour assurer une synchronisation de l'ensemble des actions. Le modèle utilise une horloge globale qui sert à spécifier des contraintes temporelles sur la disponibilité des rôles. Le modèle permet de représenter la hiérachie des rôles grâce à une synchronisation entre des automates temporisés associés à des rôles hiérarchiquement dépendant. Les propriétés de sécurité sont spécifiées en logique CTL.

3.3.4 Modélisation en Réseaux de Petri

Durant les dernières années, l'utilisation des réseaux de Petri s'est répandue grâce au nombre de travaux qui a été développé pour enrichir les réseaux de Petri ainsi que la disponibilité des outils. Selon de nombreux chercheurs (Osborn, 2002), les réseaux de Petri sont le seul formalisme qui permet de modéliser la structure du système et de

faire des analyses qualitative et quantitative. Les réseaux de Petri ont été utilisés pour la vérification de la sécurité (Ahmed and Tripathi, 2003), pour la spécification des autorisations dans les workflows (Atluri and Huang, 1996; Yi et al., 2004), l'analyse des politiques de sécurité y compris les politiques de contrôle d'accès discrétionnaires (Kumar et al., 2002; Knorr, 2000), mandataires (Knorr, 2001; Varadharajan, 1991; Juopperi, 1995; Juszczyszyn, 2003; Jiang et al., 2004; Rakkay and Boucheneb, 2006; Zhang et al., 2006b; Zhang et al., 2006a) et à base de rôles (Koch et al., 2002; Rakkay and Boucheneb, 2007; Shafiq et al., 2005).

3.3.4.1 Les réseaux de Petri pour les modèles mandataires MAC

Les réseaux de Petri ont connu une large diffusion dans le domaine de la vérification de la sécurité, notamment dans le contexte des modèles mandataires. (Varadharajan, 1991) propose un modèle de sécurité à base des réseaux de Petri ordinaires, alors que les modèles proposés dans (Juopperi, 1995; Juszczyszyn, 2003; Jiang et al., 2004; Zhang et al., 2006b; Zhang et al., 2006a) sont à base de leurs abréviations très connues, soient les réseaux Prédicat/Transition (Genrich, 1987) et les réseaux de Petri colorés (Jensen, 1997).

(Knorr, 2001) propose une modélisation au moyen des réseaux de Petri ordinaires pour analyser le flux d'informations dans un système workflow où les autorisations sont accordées en fonction du modèle de Bell-LaPadula. La vérification des propriétés de sécurité se base sur la construction d'un graphe des marquages multi-niveaux où des niveaux de sécurité sont associés à tous les objets en circulation dans le système. Le contrôle du flux d'information est réalisé par une analyse des chemins du graphe en vérifiant les propriétés de Bell et Lapadula. Cependant, il est difficile d'adopter ce modèle pour des systèmes complexes, surtout si les données sont affectées tous les niveaux de sécurité possibles en vue d'énumérer tous les états possibles du modèle.

(Varadharajan, 1991) propose une nouvelle extension des réseaux de Petri classiques, appelée IFS “Information Flow Secure net”, qui introduit un treillis de sécurité de plusieurs niveaux. La vérification de la sécurité repose sur une analyse de tous les marquages accessibles ainsi que les flots d'information afin de repérer toutes les exécutions qui pour une quelconque raison sont jugées inacceptables. Toutefois, le modèle présente des ambiguités qui ont été soulignées dans le travail de (Juopperi, 1995). Ce dernier a donc repris le modèle IFS en utilisant les réseaux Prédicat/Transition. Quelques années plus tard, (Juszczyszyn, 2003) propose un autre modèle de sécurité appelé Secure Colored Petri Net (SCPN) issu de l'IFS, pour une représentation plus compact et plus concise au moyen des réseaux de Petri Colorés. Dans chacun des modèles, des redéfinitions ont été apportées pour rendre le modèle IFS plus expressif, la sémantique plus rigoureuse et surtout pour permettre des possibilités d'analyse plus intéressantes.

(Jiang et al., 2004) propose une représentation du modèle de Bell-LaPadula à base des réseaux de Petri colorés. (Jiang et al., 2004) adopte ainsi une sémantique différente de celles présentées dans (Varadharajan, 1991; Juopperi, 1995; Juszczyszyn, 2003). En effet, le but est de proposer l'équivalent du modèle de Bell-LaPadula en un réseau de Petri coloré. L'analyse est basée sur l'exploration du graphe d'accessibilité pour vérifier les objectifs de sécurité suivants : (1) Le contrôle des accès aux objets, (2) les problèmes de transfert caché des objets (par la règle de No-Write down) et (3) les flots d'information implicites qui se produisent entre les objets. Les auteurs définissent des relations d'accès temporelles qui seront évaluées sur le graphe des marquages pour vérifier si une propriété est vérifiée ou non.

Par ailleurs, (Zhang et al., 2006b; Zhang et al., 2006a) présentent respectivement une approche pour modéliser la politique de la muraille de Chine (Brewer and Nash, 1989) et le modèle d'intégrité de Biba (Biba, 1977) en utilisant les réseaux de Petri Colorés.

On peut donc constater qu'il y a un nombre important de propositions pour modéliser

les modèles de sécurité mandataires, particulièrement au moyen des réseaux de Petri Colorés. Toutefois, ces modèles ne traitent que de la confidentialité, alors que les systèmes réels ont souvent des comportements qui dépendent du temps. La capacité de modéliser et de manipuler la dimension temporelle des évènements qui se déroulent dans le monde réel est fondamentale dans un grand nombre d'applications (bancaire, médical, multimédia, etc). La variété des applications motive nombreux travaux récents qui visent à intégrer toutes les fonctionnalités nécessaires à la prise en compte du temps dans les méthodes formelles de vérification. Dans (Rakkay and Boucheneb, 2006), nous avons proposé un nouveau modèle de sécurité à base d'une variante temporisée des réseaux de Petri colorés, nommée TSCPN (*Timed Secure Colored Petri Net*) qui intègre à la fois le temps et les notions de sécurité. À travers cette double extension, nous avons voulu offrir un modèle concis qui présente plusieurs caractéristiques et répond à différents besoins observés dans la littérature de la sécurité des informations. La notion de sécurité se base sur le principe des modèles mandataires, à savoir que nous attribuons des classes de sécurité aux informations et des niveaux d'autorisation aux utilisateurs. Cela est pour contrôler à la fois l'accès aux informations et les flux qui peuvent se produire dans tout le système. L'intégration du temps consiste à associer aux données des intervalles qui expriment leur disponibilité et validité, et ce dans le but de permettre des contrôles d'accès et de flux temporels et dynamiques. En effet, comme mesure de sécurité supplémentaire, les informations ne peuvent être utilisées qu'à l'intérieur de leurs intervalles de temps pour garantir leur validité. En dehors de cet intervalle, soit que l'information n'est pas encore disponible, soit qu'elle n'est plus valide. Nous présenterons le modèle plus en détails dans le chapitre 5.

3.3.4.2 Les réseaux de Petri pour les politiques à base de rôles RBAC

De tous les modèles de sécurité, les modèles RBAC sont les plus utilisés, essentiellement parce qu'ils présentent l'avantage d'offrir une politique neutre et garantissent plusieurs

objectives de sécurité tels que le moindre privilège "*least privilege*" et les contraintes de séparation de tâches.

Dans le contexte des modèles RBAC, il semblerait que l'utilisation des réseaux de Petri est en train d'offrir de nombreuses pistes prometteuses. Plusieurs travaux publiés récemment, proposent des approches de modélisation et de vérification pour les modèles RBAC à base des réseaux de Petri principalement pour deux raisons : (1) expliciter les flots d'information dans un modèle de contrôle d'accès, ou encore, (2) faciliter l'administration d'une politique RBAC dans les systèmes d'information. Nous avons aussi remarqué que les efforts engagés se classent en deux catégories. Certains travaux s'orientent vers la modélisation de la gestion des autorisations et des contraintes de sécurité dans les systèmes workflows (Van der Aalst, 1998). Alors que d'autres s'intéressent à élaborer une implémentation plus efficace des modèles de contrôle d'accès à base des rôles, particulièrement, pour les extensions temporelles TRBAC (Bertino et al., 2000) et GTRBAC (Joshi, 2003).

(Shafiq et al., 2005) propose une description du modèle RBAC au moyen des réseaux de Petri colorés en considérant certaines contraintes du modèle GTRBAC, sans toutefois intégrer le temps. Le modèle proposé prend en compte les contraintes de cardinalité, de séparation de tâches, de relation d'héritage entre les rôles, de précédence et dépendance. Le but d'une approche basée sur les réseaux de Petri est de prouver que les propriétés ou les règles de la politique de sécurité sont bien renforcées dans le système en bénéficiant des différentes techniques et outils offerts par les CPN. La cohérence du modèle RBAC est vérifiée à partir d'une analyse de chaque état accessible du réseau de Petri coloré. L'état est dit cohérent s'il satisfait à toutes les contraintes de cardinalité, de séparation de tâches, d'héritage et de précédence et dépendance. Il s'agit ainsi d'effectuer une analyse d'accessibilité qui repose sur le graphe des marquages correspondant au réseau de Petri. En s'assurant de la cohérence du réseau de Petri, on s'assure également de la garantie des différentes propriétés de sécurité liées à la politique de sécurité. Cependant, aucun

outil n'est utilisé. Bien que (Shafiq et al., 2005) a déjà proposé une correspondance des RBAC en réseaux de Petri colorés, l'étude est encore à compléter pour proposer une méthode d'analyse plus robuste que le graphe des marquages, procéder à la vérification des différentes propriétés de sécurité et éventuellement étendre le modèle en explicitant davantage la notion temps (des intervalles de temps pourraient être assigner aux jetons pour exprimer la durée ou la périodicité des différentes commandes d'assignation, de changement d'états des rôles, etc).

Dans un second article (Rakkay and Boucheneb, 2009) qui porte sur l'analyse de la sécurité dans les modèles RBAC, nous proposons une représentation qui exploite plus efficacement le pouvoir des réseaux de Petri colorés ainsi que l'outil CPNtools pour automatiser la vérification. Nous présentons une approche qui consiste en trois étapes : la modélisation du modèle RBAC, la spécification des contraintes de RBAC et finalement la vérification du modèle et de ses contraintes. Nous présenterons le modèle plus en détails dans le chapitre 6.

Dans sa thèse de doctorat, (Shin, 2005) présente une extension du modèle RBAC afin d'y incorporer de nouvelles contraintes plus appropriées aux systèmes d'exploitation de confiance. Cette nouvelle extension, notée E-RBAC (*Extended RBAC*), introduit deux nouvelles propriétés aux entités du contrôle d'accès, à savoir : l'ordre et la permission. Un ordre partiel est attribué à l'ensemble des opérations d'accès pour ainsi former une procédure. La permission négative est ajoutée à la procédure pour exprimer un ensemble d'opérations indésirables. Avec cette extension, l'idée est d'exprimer l'exécution des séquences de fonctionnement du système comme des ensembles ordonnés.
La spécification et la vérification formelle du modèle E-RBAC sont réalisées au moyen des réseaux de Petri Colorés en utilisant l'outil CPNtools. La vérification consiste à analyser l'exécution du modèle soit par simulation ou par la propriété de vivacité (vérifier si toutes les transitions sont franchissables).

3.3.5 D'autres approches graphiques

Dans leur article, les auteurs (Koch et al., 2002) indiquent que pour éviter les inconsistances, il est utile d'avoir une représentation précise des règles de contrôles des accès (les auteurs se basent sur des exemples de type RBAC), et si possible graphiquement afin de faciliter les tâches d'administration. L'article compare trois approches et leurs intérêts du point de vue de l'intuitivité et de la vérifiabilité. La première approche se base sur UML. Comme il n'existe pas d'outils permettant la vérification de contraintes sur les diagrammes UML, les auteurs proposent de transformer cette représentation en graphes (le troisième formalise étudié dans l'article). Les deux autres approches sont ALLOY et les Graph Transformation. Ceux-ci sont un modèle de graphe sur lesquels sont bâties des règles de transformations utilisées pour représenter le modèle de contrôle des accès. Des procédures d'inférences travaillant sur ces graphes permettent de vérifier graphiquement les politiques.

Ces trois formalismes permettent de vérifier que l'ajout d'une nouvelle contrainte (par exemple d'exclusion mutuelle entre deux rôles) ne rend pas la politique inconsistante. Cependant, ces approches sont statiques et ne permettent pas d'exprimer des contraintes linéaires comme les contraintes temporelles.

3.4 Conclusion

En résumé à ce que nous avons présenté dans cette section, il va sans dire que la formalisation du contrôle d'accès est incontournable pour une gestion sûre et efficace des droits d'accès. Plusieurs modèles et langages formels ont été proposés pour formaliser les structures, les principes et les propriétés des modèles de contrôle d'accès. Cependant aucune formalisation n'est meilleure qu'une autre dans le cas général.

Néanmoins, grâce à l'état de l'art nous pouvons identifier les critères à prendre en compte pour l'évaluation des modèles et langages proposés pour la formalisation du contrôle d'accès, que nous résumons dans les points suivants :

- le besoin d'expressivité nécessaire pour capturer la richesse des modèles de contrôle d'accès, en particulier le temps.
- le besoin d'exprimer et de concevoir des modèles de contrôle d'accès structurés,
- le besoin de définir formellement des propriétés que les modèles doivent respecter,
- le besoin d'utiliser des outils de preuve pour vérifier ces propriétés,
- le besoin de pouvoir contrôler la décidabilité des raisonnements,
- le besoin de proposer une représentation graphique des modèles,
- le besoin de proposer des outils pour assister la modélisation et la vérification des politiques.

Dans le chapitre suivant, nous présenterons les méthodes de vérification utilisées pour différentes politiques. Ces méthodes ont été proposées dans la littérature pour certains des formalismes que nous avons présenté dans ce chapitre car nous avons constaté que plusieurs propositions se sont limitées à la modélisation et n'offrent pas des techniques pour vérifier la cohérence (ou l'intégrité) de la politique de sécurité résultante.

CHAPITRE 4

VÉRIFICATION DES POLITIQUES DE SÉCURITÉ

Une politique de sécurité vise, à travers ses règlements, les différents objectifs de sécurité auxquels un système devra se conformer. Lorsqu'un système soumis à une politique présente des états incohérents ou erronés, cela peut être dû à une faille potentielle dans cette politique. Dans ce cas, c'est soit la politique de sécurité est incohérente, soit elle est incomplète :

- L'incohérence dans une politique de sécurité peut être due à des règles qui se contredisent,
- L'incomplétude se produit suite à une insuffisance de règles nécessaires pour un fonctionnement sécuritaire du système.

Il est donc essentiel de vérifier, avant d'aborder l'implantation, que la politique de sécurité considérée pour un système est cohérente (intègre). Dans ce qui suit, nous allons présenter différents concepts reliés à la vérification formelle des systèmes. Par la suite, nous allons découvrir davantage les différents travaux effectués dans le domaine de la vérification des politiques de sécurité, moyennant une recherche bibliographique plus détaillée.

4.1 Approches de vérification

La vérification nécessite trois éléments :

- un modèle, c'est-à-dire une représentation mathématique du système à vérifier,

- une formule traduisant dans une logique particulière la propriété à vérifier,
- un moteur de vérification, le moyen permettant de décider si la formule est satisfaite sur le modèle.

La vérification consiste alors à comparer le modèle du système avec sa spécification ou les propriétés qui décrivent le fonctionnement attendu du système. Il existe plusieurs approches de vérification permettant de s'assurer qu'un modèle formel d'un système informatique obéit à une spécification, typiquement écrite comme une formule d'une logique adaptée. Les recherches bibliographiques conduites au préalable nous ont permis de dénombrer deux grandes approches dans la vérification de la sécurité des systèmes d'information: les méthodes syntaxiques de preuve d'une part "*Theorem proving*" et les méthodes sémantiques d'évaluation de modèle "*Model-checking*" d'autre part.

4.1.1 Les méthodes de preuve "*Theorem proving*"

Les méthodes de preuve sont des preuves au sens mathématique du terme. Elles cherchent à déterminer si une propriété peut être obtenue à partir d'hypothèses sur les règles de déduction propres à la logique utilisée. Elles sont très puissantes au sens où elles traitent des systèmes à nombre d'états infinis, mais elles sont indécidables dans le cas général, et par conséquent difficiles à automatiser.

4.1.2 Les méthodes sémantiques "*Model-checking*"

Le Model-checking est une technique de vérification de propriétés temporelles par exploration des graphes d'accessibilité (systèmes de transitions). Il permet via un algorithme appelé model-checker, capable de dire à partir du modèle si oui ou non le système vérifie la propriété énoncée (Figure 4.1).

Figure 4.1 Vérification par Model-Checking

Les méthodes de model-checking sont donc entièrement automatiques. Cependant, elles se heurtent à deux problèmes:

- Leur utilisation est limitée aux systèmes ayant un nombre fini d'états.
- Elles posent des problèmes, en temps de traitement et en espace mémoire, liés à l'explosion combinatoire du nombre d'états.

Pour pallier ces problèmes, plusieurs solutions ont été étudiées visant notamment à :

- Réduire la taille du système de transitions (abstraction, agglomération). Plusieurs méthodes d'abstraction ont été proposées pour raisonner sur des systèmes avec données non bornées (suite au temps par exemple). L'article de (Hadjidj and Boucheneb, 2005) présente des techniques d'abstraction pour atténuer le problème de l'explosion combinatoire telle que l'abstraction par inclusion et l'abstraction par combinaison convexe. Il est démontré pour des réseaux de Petri temporels que ces abstractions ont un impact sur la performance. Ils peuvent réduire jusqu'à 100 fois la taille du graphe et conséquemment le temps de calcul.
- Explorer partiellement le système de transitions.
- Vérifier, sans construction au préalable, du système de transitions. La méthode de la vérification à la volée est reconnue comme une bonne alternative au problème

de l'explosion combinatoire et qui peut donc être employée sans avoir à produire tout le graphe des états.

L'avantage des model-checkers est qu'ils permettent de produire des contre-exemples, c'est-à-dire des exécutions du modèle qui ne satisfont pas la propriété. Plusieurs techniques de Model-checking adaptées à la vérification de classes de propriétés ont été proposées et réalisées. Les logiques utilisées pour d'écrire les propriétés sont, par exemple, les logiques temporelles (CTL, LTL,...). Nous présentons ici deux types de model-checking selon la logique temporelle utilisée.

4.1.2.1 Le Model-checking LTL pour des systèmes finis

Le model-checking ici se base sur le langage issu du produit synchrone entre un automate du système (système de transitions) et un automate de Büchi représentant la négation de la propriété à vérifier sur le système. Toute propriété exprimée en formule de LTL peut être transformée en un automate de Büchi. Le model-checking consiste à vérifier que l'intersection du langage reconnu par le système et de celui reconnu par la négation de la formule est vide. En d'autres mots, il suffit de tester que le produit des deux automates est vide. Si le système ne vérifie pas la formule, on peut extraire une erreur qui sera une trace d'exécution ou contre-exemple.

4.1.2.2 Le Model-checking CTL pour des systèmes finis

Pour les propriétés exprimables en CTL, le Model-checking se base sur le marquage des états du système de transitions. Un système de transitions satisfait une propriété p si et seulement si son état initial satisfait p. Un algorithme construit par induction l'ensemble de tous les états qui satisfont la propriété p. Chaque état est marqué par p s'il satisfait la

propriété ou sa négation si non.

4.2 Vérification des objectifs de sécurité

4.2.1 Les méthodes à base du *Model-checking*

Le model-checker SPIN "*Simple Promela Interpreter*" (Holzmann, 1997) a été proposé pour vérifier des politiques de sécurité dans de nombreux travaux (Ahmed and Tripathi, 2003; Hansen and Oleshchuk, 2005). Dans SPIN, on peut exprimer des propriétés (par exemple de vivacité et d'équité) comme des formules de LTL. À partir de ces propriétés, on transforme leurs négations en automates de Büchi. SPIN calcule ensuite le produit synchronisé de cet automate avec celui du système. Le résultat est encore un automate de Büchi dont on vérifie si le langage est vide ou non. S'il est vide, alors il n'existe aucune exécution violant la formule de départ, sinon il accepte au moins une exécution (ou mot du langage de l'automate, chaque lettre du mot étant représentée par une action du système ou de la propriété). Le langage Promela est un langage qui permet de décrire des systèmes pour le logiciel SPIN.

Les auteurs dans (Ahmed and Tripathi, 2003) présentent une méthodologie qui utilise SPIN pour la vérification des requis de la sécurité lors de la conception des systèmes de collaboration tels que les CSCW (*Computer Supported Cooperative Work*). Cette méthodologie requiert d'abord, de disposer d'un modèle formel qui décrit ces systèmes sous certaines politiques de sécurité. La coordination et les contraintes de sécurité d'un CSCW sont formellement exprimées par un modèle de collaboration basé sur les rôles. L'article propose dans un premier temps, de vérifier les propriétés de cohérence globales, à savoir : les opérations d'accessibilité, les flots de tâches, les contraintes basées sur les rôles, les flots d'information et la confidentialité, l'intégrité et la fuite d'accès. Pour cela, quatre modèles de vérification (le modèle de tâches "*Task Model*", le modèle de rôles

"*Role Model*", le modèle d'information "*Information Model*" et le modèle d'assignation "*Owner Assignment Model*") sont automatiquement construits à partir des spécifications converties en langage PROMELA. La spécification des propriétés est faite au moyen de la logique linéaire LTL. Les quatre modèles de vérification proposés sont utilisés pour vérifier les différentes propriétés de sécurité liées aux objectifs suivants :

- Assurer que les interactions entre les usagers sont conformes aux contraintes de coordination et aux flots des tâches.
- Assurer que les rôles n'ont pas de contraintes conflictuelles ou incohérentes.
- Assurer que les informations confidentielles ne sont pas acheminées à des sujets non-autorisés.
- Garantir que les informations autorisées sont accédées.
- S'assurer que toute contrainte temporelle ou conditionnelle concernant l'accès aux objets peut être satisfaite.
- Détecter les fuites d'informations à des sujets non autorisées.

Comme exemple, une propriété de séparation de tâches est exprimée en logique LTL de la façon suivante: $StaticSOC(Assistant, Student, C)$:=!$G$ $(member(C, Assistant)$ $\&\&$ $member(C, Student))$ où $member(C, Assistant)$ signifie que l'usager C est assigné le rôle d'*Assitant*. L'expression LTL signifie que le modèle ne doit jamais atteindre un état où l'usager C est assigné les deux rôles $Assistant$ et $Student$. D'autres propriétés qui répondent aux objectifs énoncés ci-dessus ont aussi été formulées en LTL et soumises à une vérification par SPIN. Le model-checker SPIN permet une représentation explicite des états, et par conséquent s'expose aux problèmes de l'explosion combinatoire. C'est la raison pour laquelle la vérification est fondée sur les quatre modèles qui

repose sur le principe d'une conception incrémentale avec abstraction de certaines informations jugées inutiles vis-à-vis de la vérification d'une propriété. Le model-checking est ainsi effectué sur les modèles réduits. Le plus intéressant dans cette approche est qu'elle offre un modèle de vérification pour les flots d'information à partir d'un modèle de contrôle d'accès. Toutefois, elle ne tient pas compte des flots d'information implicites.

Aussi, l'approche de (Hansen and Oleshchuk, 2005) utilise la logique LTL et le model-checker SPIN pour vérifier les contraintes d'autorisation dans les politiques RBAC (Figure 4.2). Le principe de vérification, est comme expliqué antérieurement, consiste à décrire le modèle RBAC en langage PROMELA et les propriétés attendues, c'est-à-dire les contraintes sur le modèle, sont exprimées en LTL. Comme exemple, une contrainte de précédence est exprimée au moyen de la logique LTL par: $\forall u \in U$: G $(account\_manager \in assign\_roles(u)) \rightarrow (account \in assign\_roles(u))$. La contrainte indique qu'un usager affecté au rôle $accountingmanager$ doit avant être affecté au rôle $accountant$. Par la suite, le vérificateur SPIN vérifie que les propriétés sont satisfaites en explorant tous les états possibles du système RBAC. Les auteurs présentent une étude de cas mais ne mentionnent aucune technique pour éventuellement pallier le problème de l'explosion combinatoire.

Les auteurs dans (Laborde et al., 2004) s'intéressent au contrôle de sécurité dans les réseaux au moyen de méthodes formelles. Ils soulèvent, en effet, le besoin de développer une technique d'évaluation formelle qui définit correctement la stratégie de sécurité dans les réseaux, un domaine qui a, jusque-là, peu retenu l'attention de la communauté des méthodes et outils formels. Certes, les méthodes formelles sont utilisées avec succès pour la vérification de propriétés fonctionnelles des protocoles réseaux (théorèmes de preuve dans (Melliar-Smith and Rushby, 1985; Owre et al., 1992) et le model-checking dans (Burch et al., 1992; Meadows, 1996), néanmoins elles restent plus marginales pour le contrôle de la sécurité. Le but de l'article est de partir d'une définition d'une poli-

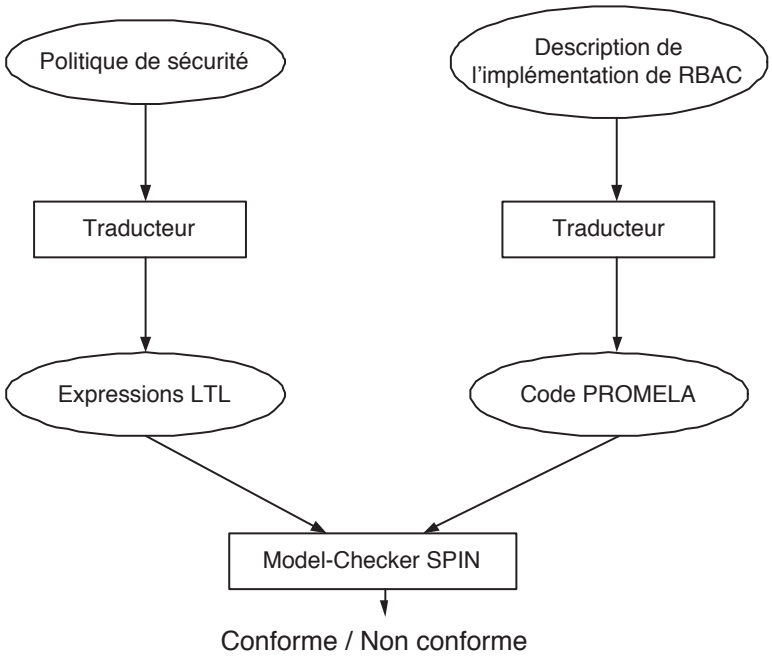


Figure 4.2 Vérification de conformité de RBAC par le model-checker SPIN

tique de sécurité, RBAC par exemple, et de vérifier si elle satisfait à des objectifs de protection précis dans les réseaux. Comme l'évaluation de la sécurité dans les réseaux nécessite de prendre en compte des détails concernant entre autres, la topologie du réseau et les différents équipements implémentés pour assurer la communication, le transfert des données et le filtrage des informations, les auteurs précisent qu'une approche basée sur les réseaux de Petri se prêtent mieux à la fois à modélisation de l'architecture du réseau et la validation des objectifs de sécurité. Les réseaux de Petri permettent de prendre en compte conjointement les aspects comportementaux (réactifs) et organisationnels (topologiques) des réseaux. Ils proposent ainsi un nouvel outil à base des réseaux de Petri colorés pour vérifier et valider qu'une politique RBAC implémentée au niveau d'un réseau garantie bien un nombre d'objectifs de sécurité (dont la confidentialité, l'intégrité et la disponibilité) exprimés dans des formalismes comme la logique temporelle. La vérification est faite au moyen du model-checking assisté par l'outil CPNtools (en appliquant une analyse d'accessibilité sur le graphe des marquages pour prouver si une

propriété est vérifiée ou non).

4.2.2 Les méthodes à base du "*Theorem Proving*"

(Drouineaud et al., 2004) propose d'utiliser Isabelle/HOL, un outil du "*Theorem Proving*", pour la vérification de la cohérence d'une politique RBAC. Quand on fait de la preuve de théorèmes, on exprime en logique les propriétés auxquelles on s'intéresse, et puis on utilise le système de déduction sous-jacent pour prouver que ces propriétés sont vérifiées. Les auteurs introduisent une extention de la logique LTL pour exprimer des contraintes d'autorisation de RBAC et dériver les propriétés souhaitées. La signature de la logique considérée contient un ensemble de symboles de prédicats et un ensemble de symboles de fonctions. Ces symboles peuvent être de deux types: rigides (intuitivement, leur interprétation est fixe à travers le temps), ou flexibles (leur interprétation peut varier en fonction du temps). La spécification des propriétés à vérifier est basée sur un ensemble d'axiomes dont ces trois qui sont les plus importantes:

spec1 : $H \vdash (AUTH\ usr\ operat\ obj) \rightarrow (EX1\ r.Role\ r\ \&\&\ ((UA\ usr\ r)\ \&\&\ ((PA\ r\ operat\ obj)\ \&\&\ (ACTIVE\_FOR\ usr\ r))))$

spec2 : $H \vdash (EXEC\ usr\ operat\ obj) \rightarrow (AUTH\ usr\ operat\ obj)$

spec3 : $H \vdash ((PDMD((ACTIVE\_FOR\ usr\ r)\ \&\&\ ((RP::'a \Rightarrow 'a\ wff)\ r)\ \&\&\ ((UP::'a \Rightarrow 'a\ wff) usr))) \mid\mid (DMD\ ((ACTIVE\_FOR\ usr\ r)\ \&\&\ ((RP::'a \Rightarrow 'a\ wff)\ r)\ \&\&\ ((UP::'a \Rightarrow 'a\ wff)\ usr)))) \rightarrow ((UA\ usr\ r)\ \&\&\ (RP\ r)\ \&\&\ (UP\ usr))$

Le premier axiome de la spécification indique que si $AUTH$ est vrai pour un triplet (usr, $operat$, obj) cela implique qu'il y a un rôle r ayant la permission d'appliquer l'opération $operat$ à l'objet obj et que l'utilisateur usr a activé le rôle r. Autrement dit,

un usager n'est autorisé à effectuer des opérations sur un objet que s'il a déjà reçu le rôle requis et que ce rôle est actif. Le deuxième axiome de la spécification indique que si $EXEC(usr, operat, obj)$ est vrai, cela implique que $AUTH(usr, operat, obj)$ est aussi vrai. Autrement dit, un usager qui exécute des opérations sur un objet a forcément les autorisations requises pour le faire. Le troisième axiome de la spécification impose, cela $ACTIF\_FOR(usr, r)$ à un moment donné implique, que $UA(usr, r)$ est vrai au moment actuel. Selon la notation familière de la logique temporelle, la spécification 3 pourrait être écrite comme suit :
$\forall usr, r$:(($F \quad ACTIVE\_FOR(usr,r)) \vee (F \quad ACTIVE\_FOR(usr,r))) \Rightarrow UA(usr,r)$

Les propriétés de sécurité à vérifier sont des propriétés de séparation des tâches qui se basent sur le principe du secret partagé (Shamir, 1979). Ainsi, une contrainte de séparation entre deux usagers ne leur permet pas de partager le même secret clé au même moment. Cette propriété s'exprime en formule logique de la façon suivante : $\forall usr, op, k : (GetShare(op) \vee EXEC(usr, op, k)) \Rightarrow (\forall u : (EXEC(u, op, k) \Rightarrow u = usr))$

Cette proposition est pour vérifier que deux utilisateurs ne pourront jamais partager la clé pour un même secret en même temps. Les auteurs expliquent que leur choix d'utiliser la preuve de théorèmes plutôt que le model-checking pour contourner le problème d'indécidabilité des propriétés exprimées en logique temporelle.

(Braghin et al., 2004) présente une approche qui intègre la théorie du π-calcul dans le système de preuve pour formaliser les règles du modèle RBAC. La sémantique opérationnelle du π-calcul est définie à l'aide d'un système de transitions étiquetées LTS. La preuve de théorème est fondée sur ce système de transitions au moyen de preuves par équivalence. Plus précisément, les différentes contraintes de sécurité sont exprimées sous forme de règles sur le système de transitions et la vérification est réalisée au moyen de la bisimulation et la congruence barbue.

Pour vérifier la validation des spécifications, (Yuan et al., 2006) ont recours au théorème prooving Z/EVES. Celui-ci comprend une interface graphique permettant d'éditer des spécifications Z et de les analyser via des démonstrations de théorèmes sur la spécification. Cependant, dans certains cas, il est très fastidieux d'arriver à démontrer tous les théorèmes.

4.3 Conclusion

À l'issue de cet état de l'art sur les méthodes de vérification, nous remarquons qu'aussi bien les méthodes syntaxiques de preuve (*Theorem prooving*) que les méthodes sémantiques (*Model-checking*) sont utilisées pour s'assurer qu'une politique est sûre, qu'elle instancie bien un modèle ou qu'elle satisfait à des propriétés de sécurité.

Toutefois, certaines propositions ne répondent pas à tous les objectifs de sécurité et peu de travaux prennent en compte le facteur temps sachant la complexité de la vérification qui en découle. Il est, en effet, délicat de trouver des critères objectifs pour comparer les différentes propositions. Néanmoins, nous pensons qu'une approche de vérification nécessite le choix d'un formalisme adapté à l'expression des aspects de sécurité et de sûreté, à l'expression des entités et relations relatives aux modèles de sécurité, à l'expression des enrichissements tels que le temps et les contraintes, et finalement validable automatiquement.

La problématique porte ainsi, sur la modélisation d'une part et la vérification d'autre part. De ce fait, il ne faut pas augmenter la puissance de modélisation au point de rendre l'analyse irréalisable. Il s'agit surtout de chercher un bon compromis entre la puissance d'expressivité, la complexité et la décidabilité de la vérification. Pour aboutir à un tel compromis, une étude très minutieuse du modèle est requise.

CHAPITRE 5

MODÈLISATION DU CONTRÔLE D'ACCÈS ET DU TEMPS DANS LES SYSTÈMES WORKFLOWS

De nos jours, les systèmes de Workflow se présentent comme une réponse technologique idéale pour répondre aux objectifs fixés par une activité de réingénierie. La qualité de la spécification, de la validation et de la mise en œuvre des systèmes workflows est donc un aspect essentiel de l'ingénierie des systèmes d'information pour garantir des systèmes fonctionnellement fiables, flexibles et maintenables, sûrs de fonctionnement même dans un environnement d'exécution ouvert. Le terme workflow (ou la gestion automatique du flux de travail) désigne un travail coopératif impliquant un nombre limité de personnes devant accomplir, en un temps limité, des tâches articulées autour d'une procédure définie et ayant un objectif global.

Dans le présent chapitre, nous présenterons, tout d'abord, les aspects de spécification et les critères de modélisation des systèmes Workflows. Par la suite, nous discuterons, appuyés par la littérature, de l'utilisation des réseaux de Petri pour la modélisation de workflow aussi bien pour ajouter le temps, que pour exprimer le contrôle d'accès.

5.1 Modélisation des systèmes *Workflows*

5.1.1 Aspects de spécification

L'article de (Jablonski and Bussler, 1996) identifie cinq aspects clés dans la spécification et la gestion des Workflows, qui sont:

- **L'aspect fonctionnement** décrit les fonctionnalités du système. Il décrit les processus

en termes de sous-processus, d'activités et de tâches indépendamment des règles, des événements et des contraintes auxquelles elles sont soumises.

- **L'aspect comportement** met en évidence les flux de contrôle intrinsèque à un processus et permet de définir les états des activités et des tâches, leurs conditions d'exécution et le flot d'événements les caractérisant.

- **L'aspect information** décrit les données utilisées dans le Workflow ainsi que les dépendances en termes de données entre les tâches. Dans un système Workflow, on peut distinguer entre les données internes, gérées par le Workflow, et les données externes, gérées par l'environnement et existent indépendamment du Workflow. L'exécution de certaines tâches peut dépendre des données produites par d'autres tâches, le flux de données décrit donc ce type de dépendance qui existe entre les tâches.

- **L'aspect organisation** décrit les structures organisationnelles, les ressources et acteurs du système, leurs rôles, la hiérarchie qui peut exister entre les rôles et les politiques de sécurité appliquée pour allouer les ressources (rôles ou autres) aux activités.

- **L'aspect opérationnel** décrit comment le Workflow doit interagir avec son environnement, d'invoquer des applications externes et la manière de communiquer avec des utilisateurs.

Nous retenons donc que pour spécifier un Workflow, il est bien important de décrire précisément les agents impliqués dans la réalisation d'une tâche coopérative, la structure des interactions qui unissent ces agents, la nature des informations qu'ils échangent et la dynamique des traitements qui doivent être effectués.

5.1.2 Critères de Modélisation

Un travail de modélisation nécessite d'abord de connaître les critères qu'une approche de spécification doit remplir pour être appropriée au domaine du Workflow.

Selon (Attali et al., 1998), une modélisation de type Workflow doit répondre à plusieurs critères. Certains critères portent sur le pouvoir d'expression du formalisme (sa capacité à exprimer de manière concise et complète les aspects pertinents d'un problème); d'autres portent sur l'outillage théorique propre au formalisme (techniques d'analyse et de validation associées, capacité à raisonner sur les structures de causalité ou les aspects temporels d'un système); d'autres enfin sont relatifs aux possibilités de mise en oeuvre du formalisme pour la réalisation de systèmes réels, et non pas seulement pour leur modélisation abstraite. Nous expliquons ci-après chacun de ces critères tel que présenté dans (Attali et al., 1998).

Critères liés au pouvoir d'expression

- Dimension structurelle : le formalisme offre-t-il des primitives permettant de structurer les modèles, par exemple en employant des techniques de décomposition hiérarchique ou de structuration par objets ?
- Dimension dynamique : quels sont les concepts utilisés par le formalisme pour décrire le comportement du système ? Permet-il de décrire aisément les notions de concurrence, de synchronisation ?

Critères liés à l'outillage théorique

- Niveau de formalité : le formalisme est-il suffisamment bien défini pour permettre l'expression et la preuve de propriétés sur les spécifications ?

- Aspects temporels : le formalisme se prête-t-il à une description du temps “quantitative” (dates et durées), ou qualitative (causalité, séquence d’événements) ? Le formalisme permet-il d’analyser ces aspects temporels, par exemple pour réaliser une prédiction de la performance globale du système ou du temps nécessaire à l’exécution d’une action complexe ?

Critères liés à la mise en oeuvre du formalisme

- Exécutabilité : les spécifications ont-elles un caractère exécutable, de manière à permettre d’envisager le prototypage ? Le formalisme est-il de nature à être implémenté ?
- Evolutivité du modèle : dans quelle mesure le formalisme facilite-t-il la modification incrémentale d’un modèle lorsque les spécifications du système changent ? Quel est l’impact d’un tel changement sur l’architecture globale du système ? Le modèle peut-il être modifié dynamiquement pendant l’exécution ?
- Protocole de communication : quel type de communication peut-on décrire entre deux composants actifs du système ? (par exemple synchrone/asynchrone, par diffusion ou dirigé, client/serveur, ...)
- Architecture : à quel type d’architecture logicielle le formalisme se prête-t-il ? (répliquée, centralisée, hybride)
- Interface utilisateur : le formalisme permet-il de décrire l’interaction du système avec ses utilisateurs, par exemple en termes de sa représentation externe ou de la structure du dialogue homme-machine ?

Il existe plusieurs modélisations répandues pour représenter les systèmes Workflows, tels que le modèle de tâches, les réseaux de Petri ou des patterns de ressource (Russell

et al., 2005). Les diagrammes d'activité d'UML étendus permettent aussi de décrire ces systèmes, cependant, ils n'offrent aucun moyen de vérifier la validité des modèles. Dans le cadre de cette thèse, notre choix s'est orienté vers les réseaux de Petri qui pourraient répondre pratiquement à l'ensemble des critères (Salimifard and Wright, 2001). Plusieurs solutions existantes (Attali et al., 1998; Mahdaoui et al., 2005), montrent déjà dans quelle mesure le formalisme des réseaux de Petri peut apporter des solutions au problème général de la spécification de la structure et de la dynamique des Workflows.

Dans ce qui suit, nous allons présenter les différentes approches proposées pour modéliser les Workflows à l'aide des réseaux de Petri.

5.2 Modélisation des Workflows à l'aide des réseaux de Petri

Dans plusieurs travaux de recherches (Oren and Haller, 2005), les réseaux de Petri se sont avérés adéquats et ont été utilisés avec succès pour la spécification des applications Workflow et la description de comportements dynamiques complexes. Ils décrivent aussi bien les aspects statiques des éléments du système que ses aspects dynamiques. De plus, les services workflows peuvent naturellement être spécifiés à l'aide de réseaux de Pétri, ce qui permet leur analyse et leur simulation. Les transitions expriment les tâches à exécuter et les places modélisent les états du système, les liens entre les deux entités expriment les relations de précédence, de parallélisme, de choix, etc.

Des études sur la représentation de processus Workflow à l'aide de réseaux de Petri ont mené à la définition de Workflow net (Van der Aalst, 1998). Dans un Workflow net, il existe une place source (qui ne comporte pas de transitions en amont) et une place finale (qui ne comporte pas de transitions en aval). De plus, il existe toujours au moins un arc reliant reliant un élément du réseau à un autre élément. Formellement, nous avons la définition suivante:

Définition 5.2.1 ***(Workflow net)***

Formellement, un réseau de Petri (P, T, F, M_0)*est un Workflow net (WF-net) ssi:* $\exists i, o \in P, {}^\bullet i = \emptyset \wedge o^\bullet = \emptyset$ *(*i *est la place source et* o *est la place finale) et le réseau résultant, en ajoutant une transition* t *tel que* ${}^\bullet t = \{o\}$ *et* $t^\bullet = \{i\}$*, est fortement connexe*[1].

Selon plusieurs références relatives à la modélisation des Workflows par les réseaux de Petri (Van der Aalst, 1996; Esparza and Silva, 1991; Ellis et al., 1995; De Michelis et al., 1994), ceux-ci constituent un puissant formalisme pour l'expression du contrôle de flux dans un processus. Aussi, plusieurs travaux (van der Aalst, 1993; Atluri and Huang, 1996; Ling and Schmidt, 2000; Gou et al., 2001) ont montré l'intérêt des raisonnements temporels dans la spécification des systèmes Workflows. Le paragraphe suivant présente certaines de ces propositions.

5.2.1 Intégration du temps dans les workflows

5.2.1.1 Time Workflow-net

Dans (Ling and Schmidt, 2000), les auteurs proposent une extension temporelle des Workflows, appelée Time WF-net et notée TWF-net. Ce modèle associe un intervalle de temps statique à chaque transition pour exprimer le temps d'exécution des tâches. Nous définissions tout d'abord, un TWF-net.

Définition 5.2.2 ***(TWF-net)***

Formellement, un TWF-net est un tuple (P, T, F, FI, M_0) *où* (P, T, F) *est un réseau workflow (WF-net) et* $FI : T \rightarrow INT$ *est la fonction qui associe à chaque transition* $t \in T$ *un intervalle de franchissement statique, i.e.* $FI(t) = [min_t, max_t]$ *où* min_t *et*

[1] *Il existe un chemin orienté de* $\{o\}$ *vers* $\{i\}$

max_t *sont des nombres réels qui représentent respectivement le temps de franchissement minimal et maximal*.

L'état du modèle est défini par un marquage temporel, noté $(M, \overline{FI})$ où M est un marquage et $\overline{FI} : T \rightarrow INT$ est la fonction qui associe un intervalle de franchissement dynamique à chaque transition t. $\overline{FI}(t) = [\overline{min_t}, \overline{max_t}]$ où $\overline{min_t}$ et $\overline{max_t}$ sont des nombres réels qui représentent respectivement le temps de franchissement au plutôt et au plus tard. L'état initial est représenté par $s_0 = (M_0, FI_0)$. Une transition t est sensibilisée si ${}^\bullet t \subseteq M$ et $t^\bullet \cap M = \emptyset$. Cette dernière condition est introduite pour éviter une situation de conflit, appelé *Contact situation* où les pré-conditions et les post-conditions associées à une transition sont satisfaites en même temps. Le temps de franchissement de la transition t est noté $\tau(t)$, i.e.: $\tau(t) \in \overline{FI}(t)$. La transition t est franchissable à partir d'un état s ssi: $\overline{min_t} \leq \tau(t) \leq min\{\overline{max_t} \mid t' \neq t \Rightarrow {}^\bullet t' \cap {}^\bullet t = \emptyset \wedge t'$ est sensibilisée à l'état $s\}$. Autrement dit, la transition t doit être franchie avant qu'elle ne soit désensibilisée par une autre transition.

Si la transition t est franchie après une durée $\tau(t)$, le modèle atteint un nouvel état $s' = (M', \overline{FI'})$ où $M' = M - {}^\bullet t + t^\bullet$, $\overline{FI'}$ est calculé tel que :

- Pour toute transition t' sensibilisée à l'état s', on a $\overline{min_{t'}} = max\{0, \overline{min_t} - \tau(t)\}$ et $\overline{max_{t'}} = max\{0, \overline{max_t} - \tau(t)\}$
- Pour toute transition t' nouvellement sensibilisée, on a $\overline{min_{t'}} = min_{t'}$ et $\overline{max_{t'}} = max_{t'}$.

Ling et Schmidt (Ling and Schmidt, 2000) utilisent des réseaux de Petri classiques et saufs (au plus un jeton par place), et par conséquent, le pouvoir de modélisation est limité.

5.2.1.2 Réseaux de Petri ITCPN (*Interval Timed Colored Petri net*)

(Gou et al., 2001) utilisent les réseaux de Petri de (van der Aalst, 1993), appelés *Interval Timed Colored Petri net* (ITCPN). Ces modèles associent des intervalles de temps aux arcs en sortie des transitions pour indiquer les délais de franchissement des jetons produits dans les places en sortie de la transition, autrement dit le temps où le jeton sera rendu disponible. Une estampille est associée à chaque jeton pour mémoriser ce temps.

Définition 5.2.3 ***(Réseau de Petri ITCPN)***
Un ITCPN est un réseau de Petri coloré temporel $(\Delta, P, T, C, F, M_0)$ *où* F *est la fonction associée à une transition* t *telle que* $FI \in CT_{MS} \rightarrow (CT \times INT)_{MS}$ *avec* $CT = \{(p, c) | p \in P \wedge c \in C(p)\}$ *et* $INT = \{[y, z] \in \mathbb{R}^+ \times \mathbb{R}^+ \mid y \leq z\}$. $F(t)$ *identifie le multi-ensemble de jetons à consommer et celui à produire suite au franchissement de la transition* t *ainsi que les intervalles des jetons produits*.

Un jeton est un tuple (p, v, x) où $p \in P$, $v \in C(p)$ et $x \in \mathbb{R}^+$. Un état s du modèle est un multi-ensemble de jetons, i.e. $s \in (CT \times \mathbb{R}^+)_{MS}$ Un événement e est un triplet (t, b_{in}, b_{out}) qui représente tous les franchissements possibles de la transition t. b_{in} et b_{out} représentent respectivement le multi-ensemble de jetons à consommer et celui à produire. Les estampilles de jetons produits sont relatives au temps de franchissement et se situent à l'intérieur de l'intervalle spécifié par $F(t)(b_{in})$. E est l'ensemble des évènements, i.e.: $E = T \times (CT \times \mathbb{R}^+)_{MS} \times (CT \times \mathbb{R}^+)_{MS}$.

Un événement (t, b_{in}, b_{out}) est sensibilisé à l'état $s \in S$ si et seulement si (1) il y a suffisamment de jetons dans les places en entrée de la transition t, i.e. $(b_{in} \leq s)$. (2) pour tout jeton de $F(t)M(b_{in})$, il existe un jeton de la même couleur dans b_{out} et tout jeton dans b_{out} a son estampille à l'intérieur de l'intervalle du jeton correspondant dans $F(t)M(b_{in})$.

Le temps de sensibilisation d'un événement est calculé en prenant le maximum des estampilles de tous les jetons à consommer, i.e. $max_{(p,v,x)\in b_{in}} x$. Un événement est sensibilisé si et seulement s'il a le plus petit temps de sensibilisation. Un événement (t, b_{in}, b_{out}) sensibilisé à l'état s peut être franchi et le modèle atteint un nouvel état $s' = (s - b_{in}) + SC(b_{out}, max_{(p,v,x)\in b_{in}} x)$ où SC est une fonction qui ajuste la valeur des estampilles des jetons dans b_{out} en ajoutant $max_{(p,v,x)\in b_{in}} x$.

(Gou et al., 2001) s'appuient sur le modèle $ITCPN$ pour représenter les Workflows des entreprises virtuelles qui fondent l'essentiel de leurs activités sur le net. Une entreprise virtuelle englobe un ensemble d'entreprises coopérants dont chacune à ses propres workflows locaux. L'approche de (Gou et al., 2001) est une approche modulaire qui consiste à représenter pour chaque entreprise son workflow interne et une interface externe. Le workflow interne encapsule tous les détails de l'entreprise (ses activités, ses ressources, le flot de contrôle et le flot d'autorisation) et conserve ainsi son autonomie et sa confidentialité. L'interface externe définit les services offerts par l'entreprise aux usagers externes ainsi que les communications échangées. Au travers des interfaces, on peut représenter les interactions entre les différents workflows. Nous présentons dans le paragraphe suivant un exemple d'application tiré de (Gou et al., 2001).

Exemple 5.2.1 *Exemple d'utilisation du modèle $ITCPN$*

Chaque activité est représentée par un réseau de Petri ITCPN (figure 5.1), i.e.: $Activity$ = (Δ_a, P_a, T_a, C_a, F_a). Une activité est exécutée lorsque le flot de contrôle du système arrive au niveau de la place P_{wait} et les ressources nécessaires à son exécution avec les rôles requis sont disponibles dans la place P_{free}.

La place P_{free} conserve toutes les ressources nécessaires à l'accomplissement de l'activité. Par conséquent, c'est une place commune à toutes les activités du workflow de l'entreprise. La séquence $P_{wait} \rightarrow t_{begin} \rightarrow P_{progress} \rightarrow t_{end} \rightarrow P_{wait}$ représente le flot d'exécution

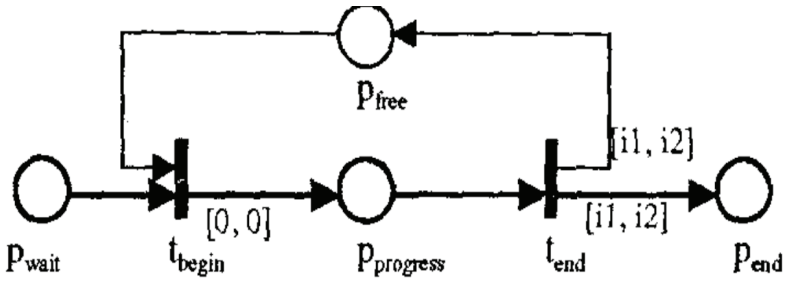


Figure 5.1 Réseau de Petri ITCPN "*Activity*"

de l'activité. Tandis que la séquence $P_{free} \rightarrow t_{begin} \rightarrow P_{progress} \rightarrow t_{end}$ représente la trace d'utilisation des ressources. Le domaine de couleur d'une activité, noté Δ_a comprend les couleurs suivantes:

- $CASEID = \{CD1, CD2, \ldots\}$ définit les IDs des différents cas d'utilisation[2].

- $PROCESSID = \{PD1, PD2, \ldots\}$ définit les IDs des processus.

- $RESOURCE = \{R1, R2, \ldots\}$ définit les resources.

La fonction couleur d'une activité est $C_a = \{C(P_{wait}), C(P_{end}), C(P_{free}), C(P_{progress})\}$ où $C(P_{wait}) = C(P_{progress}) = CASEID \times PROCESSID; C(P_{free}) = RESOURCE;$ $C(P_{progress}) = CASEID \times PROCESSID \times RESOURCE.$

Les ressources associées à une activité doivent pouvoir exécuter cette activité. L'assignation ressource-activité se fait selon les rôles des ressources.

- $ACTIVITY = \{A1, A2, \ldots\}$ définit l'ensemble des activités.

- $ROLE = \{O1, O2, \ldots\}$ définit l'ensemble des rôles.

$\forall A_i \in ACTIVITY : ROLE(A_i) \subset ROLE$ est l'ensemble des rôles requis pour exécuter l'activité A_i.

$\forall R_i \in RESOURCE : ROLE(R_i) \subset ROLE$ est l'ensemble des rôles détenus par la ressource R_i.

$\forall A_i \in ACTIVITY : CASEID(A_i) \subset CASEID$ est l'ensemble des cas d'utilisation

[2]Un cas d'utilisation correspond à un ensemble d'activités.

avec lesquels l'activité A_i peut collaborer.

Ainsi, la fonction transition $F_a = \{F(t_{begin}), F(t_{end})\}$ se définit comme suit:
- $dom(F(t_{begin}))$ = $\{<P_{wait}, <cid, pid>>, <P_{free}, rsc> \mid cid \in CASEID(A_i) \subset CASEID,\ pid \in PROCESSID,\ rsc \in RESOURCE \wedge \bigcup_{rsc \in RESOURCE} ROLE(rsc) = ROLE(A_i) \wedge ROLE(rsc) \cap ROLE(A_i) = \emptyset,$
$F(t_{begin})(<P_{wait}, <cid, pid>> + <P_{free}, rsc>) = <<P_{progress}, <cid, pid, rsc>>, [0, 0]>$
et
- $dom(F(t_{end}))$ = $\{<P_{progress}, <cid, pid, rsc>> \mid cid \in CASEID(A_i) \subset CASEID,$
$pid \in PROCESSID, rsc \in RESOURCE\},\ F(t_{end})(<P_{progress}, <cid, pid, rsc>>)$
$= <<P_{end}, <cid, pid>>, [i_1, i_2]> + <<P_{free}, rsc>, [i_1, i_2]>$

Si la transition t_{begin} est franchie, l'activité A_i va débuter son exécution immédiatement car les jetons produits ont un intervalle $[0, 0]$. Le franchissement de la transition t_{end} indique que l'activité A_i a une durée d'exécution fixée par l'intervalle $[i_1, i_2]$. Le réseau $Activity$ représente donc l'assignation des ressources à une activité où les ressources sont assignées pendant toute la durée d'exécution.

Un workflow est modélisé en connectant plusieurs réseaux $Activity$ suivant les opérateurs logiques de base, notamment AND-Split, OR-Split, AND-Join et OR-Join. Le Workflow global sera ainsi représenté par plusieurs réseaux structurés de façon hiérarchique en utilisant une transition hiérarchique $t_{activity}$ pour représenter chaque réseau $Activity$. Les activités sont reliées les unes aux autres par des transitions t_{link} (figure 5.2).

Les activités associées à un cas d'utilisation sont regroupées ensemble. Deux places sont ajoutées au réseau $P_s = \{P_{start}, P_{goal}\}$ ainsi que deux transitions $T'_s = \{t_{start}, t_{finish}\}$ pour modéliser respectivement le début et la fin d'exécution du workflow. La place P_{start} indique les cas en attente d'exécution, alors que la place P_{goal} indique les cas qui ont été exécutés. La transition t_{start} débute l'exécution d'un cas et t_{finish} termine son exécution.

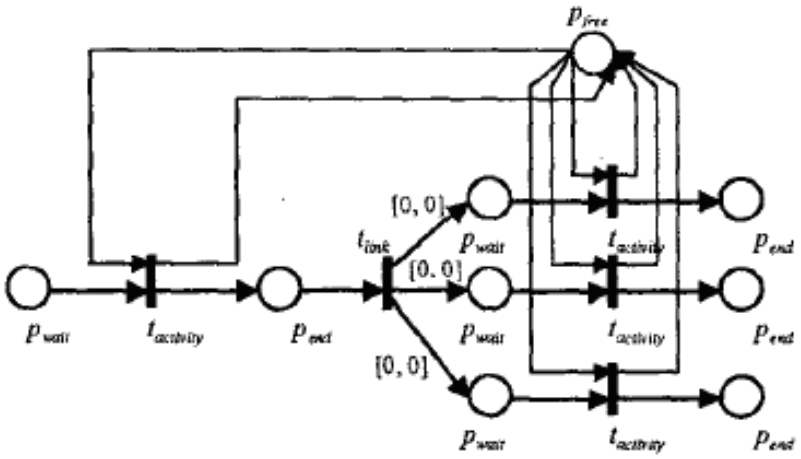


Figure 5.2 Réseau de Petri ITCPN d'une structure "AND-Split"

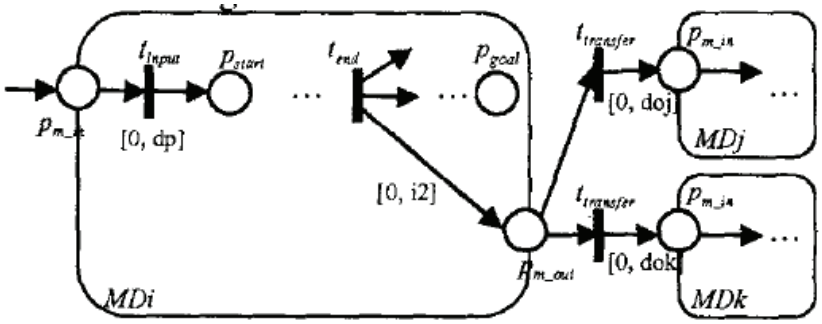


Figure 5.3 Modèle d'interaction entre les objets de l'entreprise et les interfaces

En définitif, le modèle ITCPN d'un workflow est un tuple $(ACTIVITY, \Delta_a, P_s, T_s, C_s, F_s)$ où $ACTIVITY$ est un ensemble de réseaux $Activity, T_s = T'_s \cup \{t_{link}\}, C_s = C(P_{start}), C(P_{goal})$ avec $C(P_{start}) = C(P_{goal}) = CASEID \times PROCESSID, F_s = F'_s \cup \{F(t_{link})\}$ avec $F'_s = \{F_{start}, F_{finish}\}$.

Une entreprise virtuelle comprend un ensemble d'entreprises coopérantes. Les entreprises membres sont représentées par un ensemble d'identificateurs $MEMBERID = \{MD1, MD2, ...\}$. De plus, chaque entreprise offre un certain nombre de services aux autres entreprises, ce qui est représenté par un ensemble de case IDs, i.e.: $SERVICE(MDi) = \{CDi1, CDi2, ...CDin\}$.

Les entreprises communiquent entre elles via des interfaces. Les échanges de messages se font au moyen de deux places $P_m = \{P_{m_{in}}, P_{m_{out}}\}$ ainsi que deux transitions $T_m = \{t_{transfer}, t_{input}\}$. La place $P_{m_{in}}$ contient les messages destinés à l'entreprise, alors que la place P_{m_out} contient les messages à envoyer à d'autres entreprises. La transition $t_{transfer}$ transfère les messages entre les différentes entreprises et t_{input} transforme le type des messages reçus.

Le modèle ITCPN d'une interface externe est un tuple $(Service(MDi), \Delta_m, P_m, T_m, C_m, F_m)$ où $\Delta_m = \Delta_a \cup MESSAGE$ avec $MESSAGE = \{M1, M2, ...\}$ qui est un ensemble de messages, $C_m = \{C(P_{m_{in}}), C(P_{m_{out}})\}$ avec $C(P_{m_{in}}) = C(P_{m_{out}}) = CASEID \times PROCESSID \times MESSAGE$, $F_m = \{F(t_{transfer}), F(t_{input})\}$

Dans cet exemple, le temps est utilisé pour exprimer différentes contraintes temporelles. Par exemple, l'intervalle $[0, dp]$ sur l'arc sortant de la transition t_{input} vers la place P_{start}, indique le délai d'attente pour traiter les messages entrant, soit immédiatement ou dans un délai égal à dp. L'intervalle de temps $[0, i_2]$ sur l'arc sortant de la transition t_{end} vers la place P_{m_out} signifie que les messages peuvent être produits à tout moment lors de l'exécution de l'activité. L'intervalle $[0, dok]$ sur l'arc reliant la transition transfert vers la place $P_{m_{in}}$ indique le délai de transfert des messages dans le module k.

5.2.2 Modélisation du contrôle d'accès dans les Workflows

Les auteurs de (Atluri and Huang, 1996; Yi et al., 2004) s'intéressent à la modélisation de la gestion des autorisations (les privilèges et les permissions) et des contraintes de sécurité telles que la séparation des tâches, le cloisonnement de l'information et la hiérarchie des rôles intégrée dans le modèle Workflow. Dans (Atluri and Huang, 1996), le modèle implémente le principe du cloisonnement de l'information (*Least Privilege*) où les permissions sont accordées aux sujets autorisés pour exécuter les tâches qui leur

sont prescrites pendant une période de temps. D'une part, les rôles sont attribués aux sujets selon les tâches qu'ils vont exécuter, et d'autre part l'utilisation des rôles associés aux sujets est restreinte à une période de temps en dehors de laquelle, les privilèges accordés à un sujet pour exécuter la tâche lui seront retirés pour éviter les abus. Une tâche est définie par un ensemble d'opérations, un ensemble d'objets reçus en entrée, un ensemble d'objets éventuellement produits en sortie et un intervalle de temps d'exécution. Chaque tâche est en plus, associée un ou plusieurs templates d'autorisation qui accordent aux sujets les privilèges requis au début de l'exécution de la tâche et qui les retire immédiatement à la fin de l'exécution. Des contraintes temporelles sont associées aux tâches pour vérifier que les autorisations sont spécifiées de manière à permettre aux sujets d'obtenir les accès aux objets seulement durant l'exécution de la tâche. Par exemple, si l'intervalle de temps d'une tâche est fixé à $[t_b, t_e]$ et que le temps d'exécution d'une instance de la tâche est $[t_s, t_f]$, les autorisations associées à la tâche ne sont valides que durant l'intersection des deux intervalles.

Selon (Atluri and Huang, 1996; Yi et al., 2004), représenter l'autorisation en termes de tâches offre l'avantage de raisonner à différents niveaux d'abstraction. La spécification du système peut être faite selon les compétences des décideurs puisque chacun va opérer au niveau d'abstraction auquel il est habileté. Par exemple un chef d'entreprise qui comprend l'organisation de la structure de l'organisation et les règles d'affaires, définirait les rôles légitimes pour chaque tâche. Un développeur, quant à lui, il déterminera les privilèges associés à chaque application utilisée pour effectuer une tâche. Tandis qu'un employé des ressources humaines se charge d'assigner les rôles aux usagers. Ainsi, le modèle de (Atluri and Huang, 1996) a été revu dans (Yi et al., 2004) pour prendre en compte la hiérarchie des rôles, raffiner le principe de la séparation des tâches et modéliser les situations de conflits entre tâches. Dans le modèle de (Yi et al., 2004), une tâche est représentée par deux transitions pour indiquer le début et la fin d'exécution. Il existe deux types de places : les places de rôles et les places de contrôle. Les places de rôles

en entrée d'une transition indiquent les rôles légitimes (en fonction de leurs couleurs) autorisés à exécuter la tâche. Il y a une place par rôle. Les usagers sont représentés par les jetons colorés. Une assignation rôle-usager est indiquée par la présence d'un jeton coloré dans la place de rôle. Lorsqu'une transition est tirée, les autorisations sont alors accordées à l'utilisateur pour exécuter la (les) tâche (s) associée (s) à la transition. Ces autorisations lui seront révoquées lorsque la (les) tâche (s) se termine (ent). Le modèle permet aussi de représenter la hiérarchie des rôles et ce, en ajoutant une transition qui relie un rôle senior à tous ses rôles juniors. Lorsque cette transition est franchie, un jeton dans la place du rôle senior est transféré dans une place de rôle junior. Cependant, ce modèle ne tire pas profit de la puissance de modélisation des réseaux de Petri colorés pour représenter les différentes relations d'assignation, ni les relations d'hiérarchie. Chaque rôle est représenté par une place. Ainsi, il y aura autant de places qu'il y a de rôles dans l'organisation. Par ailleurs, si un usager a plusieurs rôles, il y aura le même nombre de jetons distribués dans différentes places. Ceci pourrait produire des modèles d'autorisation dont la taille croît rapidement avec la complexité du système, et qui deviennent rapidement impossible à comprendre et à gérer. De plus, le modèle de (Yi et al., 2004) fait l'hypothèse qu'un usager ne peut être attribué plus d'un rôle dans une même instance du Workflow, ce qui n'est d'ailleurs pas réaliste.

5.3 Conclusion

Il est intéressant de remarquer qu'une approche de modélisation des Workflows, basée sur les réseaux de Petri, permet de tracer à la fois le flux des données, de contrôle et d'autorisation entre les différents intervenants du sytème.

Dans le chapitre suivant, nous présentons une extension du modèle TSCPN (*Timed Secure Colored Petri Net*), initialement présenté dans mon mémoire de maîtrise. Le modèle TSCPN est une extension des réseaux de Petri colorés temporels qui implémente une

politique mandataire multi-niveaux et permet d'exprimer des conditions temporelles sur la disponibilité et la validité des données. L'analyse du modèle se base sur le contrôle du flot d'information par le biais de la construction du graphe des classes. Or, la particularité des modèles incluant le temps, est d'être généralement infinis et à branchement infini. Pour y pallier, nous proposons une méthode de contraction qui présente des conditions suffisantes à satisfaire pour garantir la finitude (Diaz, 2001) du graphe contracté ainsi que la décidabilité des propriétés à vérifier.

CHAPITRE 6

MODÉLISATION DES POLITIQUES DE CONTRÔLE D'ACCÈS MANDATAIRE À BASE DES RÉSEAUX DE PETRI COLORÉS TEMPORELS

Un premier travail effectué à la maîtrise (Rakkay, 2005) s'est achevé par l'élaboration d'un modèle de sécurité à base d'une variante temporisée des réseaux de Petri colorés, nommé TSCPN *Timed Secure Colored Petri Net* pour des modèles de contrôle d'accès mandataires multi-niveaux.

Dans le présent chapitre, nous allons présenter une extension du modèle TSCPN telle que donnée dans (Rakkay and Boucheneb, 2006). En effet, notre objectif s'est poursuivi dans le cadre de mon projet de doctorat afin d'améliorer l'expressivité du modèle (la définition a été revue), l'approche d'analyse (en introduisant la notion du "dead token", en simplifiant le calcul et la relaxation des classes d'états au moyen des DBMs) et intégrer de nouvelles propriétés (telle que l'opacité). La nouvelle proposition a fait l'objet d'un premier article (Rakkay and Boucheneb, 2006) publié dans un numéro spécial de la revue "*Annals of Telecommunications*" sous le thème des "*New Technologies in Distributed Systems*". Aussi, une implémentation du modèle a également été réalisée dans un logiciel développé au sein de notre laboratoire de recherche.

Nous nous limiterons, dans ce chapitre, à expliciter les modifications et les ajouts qui ont été apportés. La proposition complète figure dans l'article (Rakkay and Boucheneb, 2006).

6.1 Introduction au modèle TSCPN

Le modèle TSCPN est principalement, un réseau coloré où une place peut contenir des jetons de différentes couleurs et où une transition peut être franchie de différentes manières, selon la couleur. Le choix des réseaux colorés est principalement pour la puissance de modélisation. C'est un formalisme de haut niveau, plus compact et mieux structuré que les réseaux simples, doté d'une sémantique de fonctionnement bien définie et pour lequel il existe déjà des techniques de vérification éprouvées.

La première extension consiste au facteur temps où nous avons associé des intervalles de temps aux jetons des arcs en sortie des transitions. Ainsi, lorsqu'un jeton est créé, un intervalle de temps (ou une estampille), lui sera associé. Il s'agit d'un intervalle de disponibilité où la borne inférieure spécifie le moment au plutôt où le jeton pourra être consommé et la borne supérieure indique le moment au plus tard.

La deuxième extension vise l'intégration des notions de sécurité en vue d'exprimer des politiques de sécurité mandataire multi-niveaux. La mise en oeuvre d'une politique est primordiale si l'on cherche à évaluer la sécurité d'un système. Autrement en son absence, toutes les actions peuvent être considérées comme autorisées ce qui impliquerait que n'importe quel système serait sûr. Le modèle que nous avons proposé est très général, du fait qu'il peut être employé pour mettre en œuvre un éventail de politiques de sécurité.

L'objectif est de s'assurer de la bonne conception du système, en accord avec les politiques de sécurité et par rapport aux propriétés énoncées.

6.1.1 Sémantique du modèle TSCPN étendu

Notre approche est semblable à celle proposée dans (Varadharajan, 1991). C'est-à-dire que nous attribuons à chaque jeton coloré un niveau de sécurité qui indique son degré de sensibilité, et nous rattachons à chaque place un ensemble de niveaux de sécurité afin de pouvoir spécifier les jetons colorés dont le niveau est en conformité avec ceux de la place. Nous adoptons une sémantique où les places sont des canaux de messages pour transférer des informations (les jetons) entre des processus atomiques (les transitions). De plus, la vérification est basée sur l'analyse du flot d'information.

Le modèle TSCPN étendu est un tuple $(\Delta, P, T, psec, C, FS, TM_0)$ où:

- Δ est un ensemble de couleurs de domaines. Chaque domaine est un ensemble fini et non vide. $\Delta = Data \times SC \times SC \times Flow$ où:
  - $Data$ est l'ensemble des données,
  - SC est un ensemble des classes de sécurité qui forme un treillis.
  - $Flow = \{s, i\}$ a deux valeurs possibles s (flot sûr) et i (flot non sûr).
- P et T sont des ensembles finis de places et transitions tel que $(P \cap T = \emptyset)$,
- $psec : P \rightarrow SC$ est la fonction qui associe une classe de sécurité à chaque place.
- $C : P \rightarrow PowerSet(\Delta)$. $C(p)$ est la fonction couleur qui associe à chaque place un domaine de couleurs. Soient CT l'ensemble de toutes les marques possibles, i.e.: $CT = \{(p, c)|p \in P \wedge c \in C(p)\}$ et INT l'ensemble des intervalles où les bornes sont des nombres rationnels, i.e. : $INT \subseteq \{[y, z] \in Q^+ \times (Q^+ \cup \infty)|y \leq z\}$.
- FS est la fonction associée à une transition t. $FS(t) : Dom(FS(t)) \rightarrow (CT \times INT)_{MS}$ où $Dom(FS(t)) \subseteq (CT)_{MS}$. $FS(t)$ indique, pour chaque transition, le

multi-ensemble de jetons à consommer et celui à produire après franchissement. La fonction indique aussi les intervalles de disponibilité et les classes de sécurité des jetons produits. la fonction $FS(t)$ rattache, à chacun des jetons produits, un symbole qui indique si le jeton résulte d'un flux autorisé (contrôle du flot), et dans ce cas le jeton portera le symbole s. Dans le cas contraire où le flux n'est pas en conformité avec la politique de sécurité, c'est-à-dire un flot indésirable, le jeton aura le symbole i.

- TM_0 est le marquage temporel initial qui indique les jetons se trouvant initialement dans le réseau, leurs données, leurs classes de sécurité, leurs intervalles de disponibilité ainsi que le symbole s en supposant que le modèle se trouve, initialement, dans un état sûr, i.e. : $TM_0 \in (CT \times INT)_{MS}$ où $Flow = s$ et les classes de sécurité de chaque jeton sont en conformité avec les niveaux de sécurité des places.

Définition 6.1.1 ***(jeton, jeton sûr/non sûr)***
Un jeton est un tuple de la forme $(p, cdat, sctok, scp, v, h, [a, b])$ *où* ***p*** *est la place où il réside,* ***cdat*** *est la couleur de la donnée,* ***sctok*** *est la classe de sécurité du jeton,* ***scp*** *est la classe de sécurité de la place,* ***v*** *symbole pour indiquer si le jeton résulte d'un flot sûr "*s*" ou non sûr "*i*",* ***h*** *est l'horloge du jeton (variable qui mesure le temps qui s'est écoulé depuis sa création jusqu'au moment où il sera consommé),* ***[a,b]*** *son intervalle de disponibilité.*

Pour les besoins de sécurité, on distingue entre deux types de jeton : jeton sûr ou non sûr. Un jeton sûr est un jeton dont le niveau de sécurité est en conformité à celui de la place où il réside $sctok \leq psec(p)$ où $psec(p)$ est le niveau de sécurité de la place p. Cependant, si cette condition n'est pas vérifiée, le jeton est dit non sûr.

Définition 6.1.2 ***(état)***

Un état σ du modèle TSCPN est un multi-ensemble de jetons, i.e.: $\sigma \in (CT \times Q^{+} \times INT)_{MS}$. Un état est sûr si tous les jetons présents sont sûrs.

Définition 6.1.3 ***(marquage)***
Le marquage d'un état σ, noté $M(\sigma)$ est obtenu en éliminant de σ tous les paramètres temporels.

6.1.2 Caractéristiques du modèle TSCPN étendu

Par rapport à la définition que nous avions donnée dans notre première proposition du modèle TSCPN, la définition du modèle étendu est différente dans le sens où les classes de sécurité et les symboles du flot d'informations sont des domaines de couleurs. Cette extension est en effet, plus concise et exploite mieux le pouvoir expressif des réseaux de Petri colorés. Il est ainsi plus simple de manipuler des couleurs (simples ou complexes) que ce soit pour associer des niveaux de sécurité aux places ou encore aux jetons. La nouvelle extension a induit d'autres modifications qui se résument aux points suivants :

- nous associons à un jeton le niveau de sécurité de la place où il est présent ($sctok = psec(p)$) afin de vérifier le contrôle d'accès. Nous considérons donc que l'accès est autorisé si le jeton, en l'occurrence l'information, possède un niveau de sécurité plus petit ou égale au plus grand niveau de sécurité admis dans le canal, i.e. $sctok \leq psec(p)$.

- dans la définition d'un état sûr, nous ne faisons plus mention du temps. Un état est dit sûr, si on a la garantie que l'information sera mise à la disposition de processus autorisés à accéder au canal, sans aucune autre considération temporelle car nous avons introduit la notion du jeton mort "***dead token***". Un jeton est dit mort s'il a dépassé son intervalle de disponibilité. Nous expliciterons l'intérêt de cette notion lorsque nous aborderons l'analyse du modèle.

6.2 Analyse du modèle TSCPN

En ce qui a trait à l'analyse du modèle, nous nous contenterons de présenter les modifications qui ont été apportées, notamment ce qui concerne la relaxation car c'est ce qui a nécessité le plus d'investigation. Précisons tout d'abord, que l'analyse repose sur la construction du graphe des classes où une classe regroupe tous les états accessibles suite au franchissement d'une même séquence d'évènements, indépendamment des dates de franchissement.

Définition 6.2.1 ***(Classe d'états)***
Une classe d'états est caractérisée par un marquage et une formule logique, i.e. : $C = (SM, FT)$ *où :*

- SM *(marquage symbolique) est un multi-ensemble de jetons où les horloges sont nommées, à préciser qu'il y aura une horloge par jeton.*
- FT *est une formule logique caractérisant les valuations des horloges de tous les états regroupés dans la classe* C*. Chaque valuation de l'horloge correspond à un état de la classe* C*.*

6.2.1 Évolution du modèle TSCPN

À partir de la classe initiale, le modèle évolue par franchissement d'événements valides. Les classes d'états successeurs sont calculées en appliquant la règle de franchissement présentée ci-après.

Proposition 6.2.1 ***(Règle de franchissement)***
Un événement e de la classe C est valide si et seulement si la formule suivante est

consistante : $FT \wedge (FD_{min}(e) \leq FD_{max}(e))$. *Soit* $E_v(C)$ *l'ensemble des évènements de* C *qui sont valides*.

- *Soit* e_f *un événement valide de* C. e_f *peut se produire à partir de* C *(i.e.* $C \Rightarrow_{e_f}$*) si et seulement si la formule suivante est consistante :* $FT \wedge (FD_{min}(e_f) \leq min_{e \in E_v(C)} FD_{max}(e))$. *Cette condition exprime que l'événement* e_f *est franchi dans son intervalle de franchissement et qu'il est tiré le premier parmi tous les autres évènements sensibilisés, en respectant leurs dates de franchissement au plus tard*.

- *Si l'événement* e_f *est franchissable à partir de* C*, la classe* $C' = (SM', FT')$ *accessible suite au franchissement de* e_f *est calculée comme suit :*

  1. *Le nouveau marquage symbolique* SM' *est:* $SM' = SM - {}^\bullet e_f + e_f^\bullet$
  2. *La formule logique* FT' *est calculée en quatre étapes :*
     (a) *Initialiser* FT *à:* $FT \wedge (FD_{min}(e_f) \leq dh \leq min_{e \in E_v(C)} FD_{max}(e))$.
     (b) *Remplacer chaque horloge* h *par* $h - dh$*;*
     (c) *Ajouter pour chaque jeton* $(p, cdat, sctok, scp, v, h, [a, b])$ *créé par* e_f*, i.e. :* $(p, cdat, sctok, scp, v, h, [a, b]) \in e_f^\bullet$*, la contrainte* $h = 0$*;*
     (d) *Éliminer par substitution toutes la variable* dh *et les horloges de tous les jetons consommés par l'événement* e_f *(i.e. les jetons de* ${}^\bullet e_f$*)*

Cependant, en basant la construction du graphe des classes uniquement sur la règle de franchissement, nous pouvons avoir une infinité de classes accessibles car les horloges de certains jetons peuvent accroître indéfiniment. Afin de remédier à cette limitation, nous avons proposé une relaxation de chaque nouvelle classe calculée, et ce avant de la comparer avec celles déjà produites pour tester l'égalité.

6.2.2 Relaxation d'une classe d'états

Nous appliquons la relaxation d'une classe d'états qui consiste à subdiviser et étendre le domaine de certaines horloges en ajoutant des valeurs qui n'affecteront pas l'évolution de la classe d'états (des valeurs non significatives).

Définition 6.2.2 ***(Relaxation)***
Soient la classe d'états $C = (SM, FT)$*, un jeton* $(p, cdat, sctok, scp, v, h, [a, b])$ *de la classe* C *et* lim_h *la plus grande valeur significative de* h*. La valeur de* lim_h *est définie par :* $lim_h = b$ *si* $b \neq \infty$ *autrement* $lim_h = a$*.*

La relaxation de C comporte deux étapes. La première étape consiste à subdiviser la classe C en des sous ensembles afin d'isoler les horloges qui ont dépassé leurs limites, de celles qui ne l'ont pas encore dépassé. Lors de la deuxième étape, le domaine de chaque horloge h qui a dépassé sa limite lim_h sera remplacé par $]lim_h, \infty[$ puisque ces valeurs ont le même effet sur l'évolution du système. La relaxation de C produit ainsi l'ensemble des sous classes suivantes : $\{(SM, FT_c) | (c = \emptyset \vee c \subseteq V_{SM}\}$ où c est l'ensemble des horloges qui n'ont pas encore dépassé leurs limites, pendant que celles de $V_{SM} - c$ ont dépassé leurs limites. La formule FT_c est une formule consistante pour caractériser les différentes sous classes de C. Cette formule est calculée en trois étapes :

1. Initialiser FT_c par : $FT \wedge (\bigwedge_{h \in c} h \leq lim_h) \wedge (\bigwedge_{h' \in V_{SM}-c} h' \leq lim_{h'})$
2. Éliminer par substitution toutes les variables $V_{SM} - c$
3. Ajouter les contraintes : $\bigwedge_{h' \in V_{SM}-c} lim_{h'} < h' \leq \infty$

La dernière opération risque d'ajouter des états qui ne sont pas accessibles. Toutefois, tous ces états ont la même évolution que d'autres états de la sous-classe. Ainsi, cette opération n'affecte pas le comportement de la sous-classe.

6.2.3 Implémentation

Comme le calcul des classes d'états accessibles requiert la résolution de systèmes d'inéquations, nous proposons une implémentation au moyen des matrices de bornes (DBMs) pour représenter la formule logique FT. Nous représentons donc la forme canonique de la formule FT par la matrice H définie comme suit:
$H : (V_{SM} \cup \{o\})^2 \rightarrow (Q \cup \infty) \times \{<, \leq\}, H(x, y) = (Sup(x - y, FT), \prec)$ où $Sup(x - y, FT)$ représente le suprêmum de $x - y$ dans le domaine de FT et $\prec \in \{\leq, <\}$.

Ainsi, toute classe $C = (SM, FT)$ peut être aussi caractérisée par $C = (SM, H)$ et la comparaison de deux classes d'états $C_1 = (SM_1, FT_1)$ et $C_2 = (SM_2, FT_2)$ est simplifiée à : $SM_1 = SM_2$ et $H_1 = H_2$

De plus, la règle de franchissement et les opérations de relaxation seront également simplifiées au moyen de cette nouvelle caractérisation qui permet d'éviter la résolution des systèmes d'inéquations. Nous avons en effet, poursuivi le travail fait à la maîtrise en proposant une implémentation de la règle de franchissement sous une forme simplifiée qui permet de calculer directement la forme canonique des classes d'états accessibles. Nous présentons la proposition 6.2.2 qui est une simplification de la règle de franchissement.

Proposition 6.2.2 *Simplification de la règle de franchissement*
Soit e_f un évènement valide de $C = (SM, FT)$ et H la forme canonique de FT i.e.: $e_f \in E_v(C)$. L'événement e_f peut se produire à partir de la classe C si et seulement si :
$\forall e_i \in E_v(C),\ \forall j_f \in^\bullet e_f, j_i \in^\bullet e_i,\ (a_f - b_i, \leq) \leq H(h_f, h_i)$

Soit $C' = (SM', FT')$ la classe successeur de C suite au franchissement de l'événement e_f. Nous calculons la forme canonique H' de FT' à partir de la forme canonique H correspondant à FT de la façon suivante :

- $\forall x \in V_{SM} \cup \{o\}, H'(x,x) = (0, \leq)$
- $\forall x \in V_{SM}$,
  - *Si les jetons sont créés suite à* e_f*:* $H'(o,x) = (0, \leq)$ *et* $H'(x,o) = (0, \leq)$
  - *Sinon* $H'(x,o) = min_{e \in E_v(C)}(min_{j_i \in {}^\bullet e_i}((b_i, \leq) + H(x, h_i)))$ *et* $H'(o,x) = min(H(o,x), min_{j_i \in {}^\bullet e_i}(H(h_f, x) + (-a_f, \leq)))$
- $\forall x, y \in V_{SM}, x \neq y$
  - *Si les jetons ne sont pas créés suite à* e_f*:* $H'(x,y) = min(H(x,y),\ H'(x,o) + H'(o,y))$
  - *Sinon* $H'(x,y) = H'(x,o) + H'(o,y)$

Preuve 6.2.1 *:*

(i) Si on développe FD *en éliminant* dh *par substitution, nous pourrons réécrire la condition de franchissement, présentée dans la proposition 6.2.1, de la façon suivante :* $\forall e_i \in E_v(C),\ \forall j_f \in {}^\bullet e_f, j_i \in {}^\bullet e_i,\ FT \wedge h_i \leq b_i \wedge a_f - b_i - h_f + h_i \leq 0$. *Puisque* $H(h_f, h_i) = (Sup(h_f - h_i, FT)$ *et* $h_i \leq b_i$ *alors cette formule est consistante ssi :* $\forall e_i \in E_v(C),\ \forall j_f \in {}^\bullet e_f, j_i \in {}^\bullet e_i,\ (a_f - b_i, \leq) \leq H(h_f, h_i)$

(ii) Calcul de $H'(o,x)$ *et* $H'(x,o)$ *pour* $\forall x \in V_{SM}$ *: les jetons nouvellement créés ont des horloges égales à 0 alors que les horloges des autres jetons vont être incrémentées de* dh*. Dans ce dernier cas, le domaine de* $x + dh$ *dans la condition de franchissement est égal au domaine de* x *dans* FT'*. Rappelons que la condition de franchissement est :* $FT \wedge (FD_{min}(e_f) \leq dh \leq min_{e_i \in E_v(C)} FD_{max}(e_i))$*. En développant* FD*, nous obtenons :* $FT \wedge (max(0, max_{j_f \in {}^\bullet e_f}(a_f - h_f)) \leq dh \leq min_{e_i \in E_v(C)} min_{j_i \in {}^\bullet e_i}(b_i - h_i))$.

Le domaine de $x + dh$ *peut être obtenu en ajoutant* x *à chaque membre du second*

terme de la formule. La valeur de l'élément $H'(x, o)$ est obtenue en remplaçant, dans la borne supérieure de $x + dh$, tous les termes $x - h_i$ par la plus grande valeur de $x - h_i$ satisfaisant FT (i.e.:$H(x, h_i)$). La valeur de l'élément $-H(o, x)$ est obtenue est remplaçant, dans la borne inférieure de $x + dh$, tous les termes $x - h_f$ par la plus petite valeur de $x - h_i$ qui satisfait FT (i.e.:$-H(h_f, x)$ et x est remplacé par $-H(o, x)$.

(iii) Calcul de $H'(x, y)$ pour $\forall x, y \in V_{SM}$: Si x et y sont associés à des jetons créés par e_f, on a $x = 0$ ou $y = 0$, et ainsi $H'(x, y) = H'(x, o) + H'(o, y)$
Si x et y sont associés à des jetons non créés suite à e_f, le domaine de $x - y$ est l'intersection entre le domaine $x-y$ dans FT' et $[-H'(o, x)-H'(y, o), H'(x, o)+ H'(o, y)]$ qui résulte de la relation $x - y = (x + dh) - (y + dh)$. Par conséquent, on obtient : $H'(x, y) = min(H(x, y),\ H'(x, o) + H'(o, y))$

Le graphe des classes du modèle TSCPN sera ainsi construit en appliquant la règle de franchissement simplifiée à chaque nouvelle classe d'états accessible, et ce partant de la classe initiale. Chaque nouvelle classe calculée sera d'abord relaxée avant d'être ajoutée au graphe. Nous avons donc aussi formulé la proposition 6.2.3 pour relaxer une classe exprimée sous la forme canonique. La relaxation d'une classe d'états $C = (SM, FT)$ produit l'ensemble des sous classes d'états suivantes: $\{(SM, FT_c) \mid (c = \emptyset \vee c \subseteq V_{SM}\}$. On peut calculer la forme canonique $C_c = (SM, H_c)$ de chacune des sous classes $C_c = (SM, H_c)$ (où $FT_c = FT \wedge (\bigwedge_{h \in c} h \leq lim_h) \wedge (\bigwedge_{h' \in V_{SM}-c} h' > lim_{h'})$) selon la proposition 6.2.3:

Proposition 6.2.3 *Relaxation d'une classe d'états sous sa forme canonique*
Tout d'abord, on vérifie la consistance de la formule FT_c. Celle-ci est consistante ssi:

- $\forall h \in c, (lim_h, \leq) + H(o, h) \geq (0, \leq)$,

- $\forall h' \in V_{SM} - c, (-lim_{h'}, <) + H(h', o) \geq (0, \leq)$, *et*
- $\forall h \in c \wedge h' \in V_{SM} - c, (lim_h - lim_{h'}, <) + H(h', h) \geq (0, \leq)$.

Si FT_c est consistante, on peut calculer sa forme canonique en se basant sur H :

- $\forall x \in V_{SM}$, $H_c(x, o) = min(H(x, o),\ min_{h \in c}(H(x, h) + (lim_h, \leq)))$
  $H_c(o, x) = min(H(o, x),\ min_{h' \in V_{SM} - c}(H(h', x) + (-lim_{h'}, <)))$
  $H_c(x, x) = 0$
- $\forall (x, y) \in V_{SM},\ x \neq y$, $H_c(x, y) = min(H(x, y),\ H_c(x, o) + H_c(o, y))$
- $\forall x \in V_{SM-c},\ H_c(x, o) = (\infty, <)$ *et* $H_c(o, x) = (-lim_x, <)$
- $\forall x \in V_{SM-c},\ \forall y \in V_{SM} - x,\ H_c(x, y) = (\infty, <)$ *et*
  $H_c(y, x) = H_c(y, o) + (lim_x, <)$

Preuve 6.2.2 *: Pour la preuve formelle de cette proposition, nous utilisons la notion du graphe des contraintes. Nous construisons le graphe des contraintes de la formule FT_c qui est obtenue à partir de celui de FT en ajoutant deux arcs : $\{(o, h, (lim_h, \leq)) \mid h \in c\}$ et $\{(h', o, (-lim_{h'}, <)) \mid h' \in V_{SM} - c\}$. Ces arcs correspondent respectivement aux contraintes : $\bigwedge_{h \in c} h \leq lim_h$ et $\bigwedge_{h' \in V_{SM-c}} h' < -lim_{h'}$.*

La formule FT_c est consistante si et seulement si le graphe résultant n'admet aucun cycle négatif. Dans ce cas, le poids du plus court chemin du nœud x au nœud y est égal à la borne supérieure de la distance $(y - x)$, i.e.: $H_c(y, x)$. L'opération de relaxation remplace le domaine de chaque horloge h', appartenant à $V_{SM} - c$, par l'intervalle $]lim_{h'}, \infty[$.

Lorsqu'une classe d'états est relaxée, elle est subdivisée en plusieurs sous-classes, séparant ainsi les états où les horloges des jetons n'ont pas encore atteint leurs limites, de

ceux où les horloges ont dépassé leurs limites. Ainsi, tous les états dont les jetons sont morts (*dead tokens*) sont regroupés ensemble dans une même classe. Par ailleurs, la relaxation garantit que les états d'une même classe partagent les mêmes évènements valides qui, de toute évidence, n'utilisent pas les jetons morts. La relaxation est ainsi plus efficace.

Nous avons aussi proposé d'utiliser l'abstraction par inclusion proposées dans (Hadjidj and Boucheneb, 2005), comme autre moyen pour atténuer davantage le problème de l'explosion combinatoire. L'abstraction par inclusion consiste à inclure une classe dans une autre de manière à réduire le nombre de classes à explorer. Plus précisément, lorsqu'une classe d'états est relaxée, nous utiliserons le critère d'inclusion au lieu de l'égalité pour regrouper ensemble les classes qui sont déjà incluses dans d'autres. Il existe aussi d'autres méthodes comme l'abstraction par combinaison convexe dans (Hadjidj and Boucheneb, 2005). Celle-ci suggère de regrouper ensemble des classes dont l'union est convexe[1]. De plus, ces deux types d'abstractions préservent les chemins d'exécution, ce qui fait qu'elles conviennent à notre approche puisque la vérification se ramène à un problème d'accessibilité.

6.3 Analyse de la sécurité

Nous avons utilisé une approche logique au moyen de CTL pour exprimer les propriétés de base de la sécurité, à savoir la confidentialité, l'intégrité et la disponibilité. Nous présentons ici les différentes propositions atomiques reformulées suivant les modifications apportées.

[1]Les zones d'états regroupés doivent avoir le même marquage

6.3.1 Confidentialité des données

La propriété de confidentialité peut être exprimée par la notion du flot sûr qui garantit que les informations sont utilisées uniquement pour le but dans lequel elles sont données. On formalise ainsi la confidentialité par la proposition atomique : **flot_sur** qui est évaluée à vrai si tous les jetons de l'état s'écrivent sous la forme $(p, cdat, sctok, scp, s, h, [a, b])$ avec $sctok \leq scp$, et à faux autrement. On écrira la formule CTL : $AG(flot\_sur)$. Il est à remarquer que les conditions présentées concordent avec celles de Bell et LaPadula, soient :

- La règle "*No Read-up*" qui stipule qu'un sujet s n'est autorisé à accéder en lecture aux informations contenues dans l'objet o que si le niveau de sécurité de s est supérieur ou égal au niveau de sécurité de o.
- la règle "*No Write-down*", qui considère qu'un sujet s n'est autorisé à accéder en écriture aux informations contenues dans l'objet o que si le niveau de sécurité de s est inférieur ou égal au niveau de sécurité de o.

6.3.2 Intégrité des données

La propriété de l'intégrité peut être formalisée par la proposition atomique : **etat_sur** qui est évaluée à vrai si tous les jetons de l'état s'écrivent sous la forme (p, $cdat$, $sctok$, scp, $v, h, [a, b]$) où $sctok \leq scp$ et $v \in \{s, i\}$, et à faux autrement. On écrira la formule CTL: $AG(etat\_sur)$.

6.3.3 Disponibilité des données

La propriété de disponibilité peut être formalisée par la proposition atomique : **violation_temporelle** qui est évaluée à vrai si au moins un jeton (p, $cdat$, $sctok$, scp, v, h, $[a, b]$) où $v \in \{s, i\}$ de l'état a une horloge qui se trouve au delà de la borne maximale de son intervalle de disponibilité $(h > b)$, et à faux autrement. On écrira la formule CTL: $AG(violation\_temporelle)$.

6.3.4 Fuites d'information : Non-interférence et opacité

Autres propriétés intéressantes à définir sont issues de la notion de flot d'information. Le contrôle du flot d'information a été formulé de diverses manières et différentes approches ont été proposées qui visent généralement à décrire la connaissance qu'un intrus pourrait gagner en termes de propriétés. Ici, nous citons la non-interférence et l'opacité.

6.3.4.1 Non-interférence

On dit qu'un processus P interfère avec un processus Q si les actions de P influencent ce que Q peut observer ou faire. En effet, les interférences sont souvent à la source de fuites d'informations confidentielles lorsqu'un processus réussit à déduire de l'information, à laquelle il n'a pas normalement accès, à partir d'une observation des actions d'un autre. Vérifier la non-interférence d'un programme consiste à s'assurer que l'information contenue dans une variable x n'interfère pas avec l'information contenue dans une variable y, si le niveau de sécurité de y est plus petit que celui de x. Un aperçu intéressant sur la non-interférence exprimé avec des réseaux de Pétri peut être trouvé dans (Busi and Gorrieri, 2003).

6.3.4.2 Opacité

La non-interférence est la plupart du temps considérée comme trop rigide pour servir de description utile parce qu'elle cherche à capturer l'absence complète de flots d'information. À cet égard, (Bryans et al., 2004; Bryans et al., 2005) proposent d'employer la notion de l'opacité à la place, une variante de la non-interférence qui se limite à capturer certaines informations en particulier. L'opacité a été utilisée dans le contexte de l'analyse des protocoles de sécurité (Abadi and Gordon, 1997). Par exemple, la valeur d'une variable v, est dite opaque pour une exécution du protocole si un attaquant ne peut pas déduire sa valeur à partir des observations et des déductions qui lui sont disponibles pendant l'exécution. Un système satisfait cette propriété si pour n'importe quelle autre valeur de v, il y a une autre exécution possible du protocole qui provoque des observations pour l'adversaire qui sont non distinguables des observations originales. Ainsi, l'opacité repose sur la notion d'équivalence observationnelle qui garantit que deux exécutions sont non distinguables de n'importe quel attaquant qui peut observer à un niveau de sécurité.

Dans (Rakkay and Boucheneb, 2006), nous avons montré, au moyen d'un exemple tiré de (Bryans et al., 2004), comment exprimer l'opacité via le modèle TSCPN.

La figure 6.1 est le réseau de Petri d'un scénario dans l'industrie chimique où une compagnie de produits chimiques A demande à une autre compagnie B (transition a_1) d'effectuer une étude de faisabilité concernant le développement d'un nouveau produit chimique. Lorsque l'étude est terminée (transition b_1). La compagnie A est informée des conclusions (transition a_2). À partir de ces conclusions, la compagnie A fera appel à une troisième compagnie C (transition a_3) ou la compagnie C' (transition a'_3) pour produire un rapport sur la sécurité du nouveau produit chimique. La loi permet à la compagnie choisie d'interroger la compagnie B sur tous les aspects de l'étude de faisabilité. Toutefois, la compagnie choisie n'est pas autorisée à révéler son identité à la compagnie B,

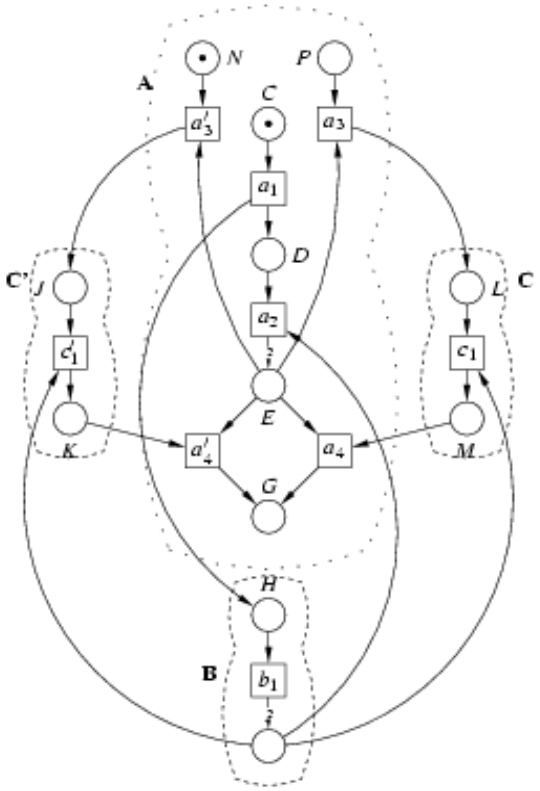


Figure 6.1 Réseau de Petri d'un scénario de l'industrie chimique

afin de protéger l'intégrité de B. L'objectif donc est de faire en sorte que les interactions visibles, par la compagnie B, ne révèlent par l'identité de la compagnie choisie.

L'essentiel de l'idée proposée dans (Bryans et al., 2004) consiste à définir une fonction *obs* qui traduit l'information qu'un observateur est permis de voir. De plus, étant donnée une fonction *obs* et une propriété ϕ, on veut vérifier si un observateur, à partir des observations définies par la fonction *obs*, a la capacité de déduire la propriété ϕ à une certaine étape de l'exécution du système.

Dans le cas du modèle TSCPN, nous proposons d'étiqueter une transition invisible par une action τ. Une τ-transition correspond à une exécution d'un sujet qui est supposé invisible au regard de tous les observateurs du système. Nous préservons en quelque sorte l'anonymat d'un sujet en abstrayant (rendant opaque) ses actions dans le système.

Pour le présent exemple, les transitions a_3, a'_3, a_4 et a'_4 ne doivent pas être visibles par la compagnie B. De plus, les transitions c_1 et c'_1 doivent être maintenues anonymes. Ainsi:

- $obs(a_3) = obs(a_4) = obs(a'_3) = obs(a'_4) = \tau$
- $obs(c_1) = obs(c'_1) = \gamma$

Le modèle a deux actions possibles : $a_1b_1a_2a_3c_1a_4$ et $a_1b_1a_2a'_3c'_1a'_4$ qui conduisent aux observations suivantes :

- $obs(a_1b_1a_2a_3c_1a_4) = a_1b_1a_2\tau\gamma\tau$
- $obs(a_1b_1a_2a'_3c'_1a'_4) = a_1b_1a_2\tau\gamma\tau$

Ces observations étant équivalentes, il est garantit que les actions des compagnies C et C' sont opaques pour la compagnie B.

La vérification se ramène ainsi à un problème d'équivalence observationnelle entre des traces d'exécution. Si celles-ci sont équivalentes, nous pouvons dire que la propriété est opaque et le système préserve l'anonymat des processus qui ont exécuté les séquences de transitions composant la trace.

6.4 Conclusion

En conclusion, nous précisons que cette nouvelle extension du modèle TSCPN est nettement le fruit d'un compromis entre l'expressivité du modèle et les possibilités de vérification. En effet, c'est dans une perspective d'obtenir un graphe de classes plus compact qui permet aussi de vérifier les propriétés de sécurité, que nous avons du revoir et adapter la définition du modèle. Nos principales contributions se résument dans les points suivants :

- nous proposons une extension du modèle TSCPN qui exploite mieux le pouvoir d'expression des réseaux de Petri colorés.
- nous proposons une simplification de la règle de franchissement par l'utilisation des DBMs.
- nous proposons une opération de relaxation qui produit les classes relaxées sous leurs formes canoniques.
- nous sommes parvenus à une approche de contraction de l'espace des états que nous considérons satisfaisante sachant qu'elle permet de préserver les propriétés du modèle que l'on veut vérifier.
- nous montrons comment exprimer une nouvelle propriété, soit l'opacité, sur le modèle TSCPN étendu.

Nous mentionons de plus qu'il est possible de recourir à la vérification à la volée "*on the fly*" pour réduire plus finement la taille du graphe des classes. Cette technique consiste à construire de manière incrémentale le graphe des classes selon les besoins de la vérification. Elle permet de détecter des erreurs dans des graphes dont la taille est trop importante pour qu'ils puissent être entièrement explorés.

Pour introduire le modèle TSCPN étendu dans (Rakkay and Boucheneb, 2006), nous avons appliqué les règles de Bell-LaPadula en tant que politique mandataire multi-niveaux. Toutefois, nous précisons qu'il est tout à fait possible d'exprimer d'autres politiques de sécurité . En effet, pour appliquer une politique de sécurité, cela revient à imposer des règles de filtrage (suivant les exigences du système en termes de sécurité) au niveau des transitions du modèle TSCPN étendu. Une application directe et simple consiste à exprimer le modèle d'intégrité de Biba (Biba, 1977) qui est très similaire à celui de Bell-LaPadula mais qui s'intéresse à la propriété de l'intégrité.

Dans le prochain chapitre, nous présentons un second travail qui s'est achevé par une formalisation des modèles RBAC en utilisant les réseaux de Petri colorés et l'outil CP-Ntools.

CHAPITRE 7

APPROCHE D'ANALYSE DE SÉCURITÉ DES MODÈLES D'ACCÈS À BASE DE RÔLES AU MOYEN DES CPN ET CPNTOOLS

L'implémentation des modèles RBAC est un processus assez complexe et nécessite un effort de modélisation exhaustif de son fonctionnement à cause de la forte interaction entre les organisations. Toutefois, les réseaux de Petri en tant que support de modélisation offre aussi un support pour mieux gérer la spécification d'une politique de sécurité RBAC, de sa mise à jour, en plus de la capture des propriétés dynamiques du modèle, notamment les flots d'autorisation. D'ailleurs, il s'est avéré, suite à de nombreux travaux, que les réseaux de Petri présentent un niveau d'abstraction suffisant pour permettre la gestion des autorisations et les propriétés s'y rattachant. Il est possible d'y exprimer l'ensemble des éléments constituant une politique de sécurité, à savoir : les sujets, les objets, les actions, les rôles, etc.

Notre but est de fournir des guides pour formaliser une politique RBAC en s'appuyant sur les réseaux de Petri colorés et leurs outils d'édition et d'analyse existants. Pour cela, nous proposons une description générique d'un modèle RBAC enrrichi par certaines fonctionnalités et aspects pertinents tirés du modèle GTRBAC. La description est construite de façon modulaire et incrémentale au moyen de réseaux de Petri colorés hiérarchisés (où certaines transitions masquent un réseau de Petri). Cette idée découle d'une proposition déjà existante que nous affinons toutefois, en utilisant l'environnement logiciel CPNtools (Jensen et al., 2007). Cet outil permet l'édition et la simulation des réseaux de Petri colorés ainsi que la génération et l'analyse de l'espace d'états. Il s'adapte d'autant plus à notre étude qu'il permet également de créer des modèles hiérarchisés. Nous montrons aussi que les réseaux de Petri représentent un formalisme

intuitif pour aider à la détection des incohérences et incomplètudes dans les règles d'une politique de sécurité RBAC (section 7.3). Les anomalies auxquelles on s'intéresse peuvent être exprimées par des propriétés d'atteignabilité pour déceler la redondance et l'incomplétude des règles, des propriétés de sûreté pour vérifier l'inconsistance, ou des propriétés de vivacité pour la présence d'états de blocage. Ces anomalies peuvent être détectées en procédant à une analyse du graphe de marquages calculé à partir du réseau de Petri modélisant la politique RBAC. Cette proposition a fait l'objet d'un second article (Rakkay and Boucheneb, 2009) publié dans un numéro spécial de la revue "*Transactions on Computational Science*" sous le thème "*Security in Computing*".

Dans le présent chapitre, nous commencerons d'abord par présenter les contraintes de cohérence sur lesquelles sera basée, l'analyse du modèle RBAC éventuellement construit. Ensuite, nous détaillerons les étapes de modélisation sur CPNtools ainsi que la vérification de la cohérence. Nous terminerons ce chapitre par une reflexion sur une perspective d'ajouter les temporisations supportées par CPNtools pour exprimer des modèles RBAC temporels.

7.1 Contraintes de cohérence

Comme nous l'avons préalablement mentionné, une politique de sécurité est un ensemble de règles qui définit le comportement attendu du système sous cette politique. Le système est dit en conformité avec la politique sous-jacente si chaque état du système peut être déduit de l'ensemble des règles / axiomes de la politique. Un état ou un comportement erratique du système peut être attribué à un éventuel défaut dans la politique. Cette irrégularité peut-être en raison soit d'une incohérence dans la politique ou d'une incomplétude. Une incohérence politique se produit si deux ou plusieurs règles d'un ensemble de règles comprenant la politique se contredisent. L'incomplétude implique que l'ensemble de règles définissant la politique n'est pas suffisante pour capturer tous

les états du système. Dans ce contexte, la vérification de sécurité peut apporter la preuve que les propriétés ou les règles d'une politique de sécurité sont bien appliquées dans le système. (Shafiq et al., 2005) utilise un ensemble de règles de cohérence, qui sont essentiellement des extensions de règles de cohérence définis dans (Gavrila and Barkley, 1998). Cet ensemble de règles étendu permet de représenter diverses contraintes de RBAC notamment, les contraintes de cardinalité, d'héritage et de séparation des tâches.

7.1.1 Contraintes de cardinalité

- Le nombre d'usagers autorisés à un rôle ne doit pas excéder la cardinalité fixé à ce rôle. En d'autres mots, un rôle ne peut être utilisé par plus que n usagers à la fois, c'est la règle de cardinalité des rôles "*Authorization Role Cardinality*".

- Le nombre de rôles permis pour tout usager ne doit pas excéder le nombre de rôles maximal que l'usager est autorisé à activer. Un usager a un nombre maximal de rôles auquel il est autorisé, soit la règle de cardinalité des usagers "*Authorization User Cardinality*".

- À tout moment, le nombre de rôles activés par un usager ne doit pas excéder le nombre de rôles maximal qu'il a la permission d'activer. C'est la règle de cardinalité d'activation des usagers "*Activation User Cardinality*".

- Le nombre d'usagers ayant activé un rôle durant leur session ne doit pas excéder la cardinalité des rôles actifs spécifiée pour ce rôle. C'est la règle de cardinalité d'activation des rôles "*Role Activation Cardinality*".

- Le nombre de sessions activées par un usager ne doit pas excéder le nombre de sessions maximal auquel il est autorisé. En d'autres mots, un usager a un nombre maximal de sessions. C'est la règle de cardinalité des sessions usagers "*User Session Cardinality*".

7.1.2 Contraintes de la hiérarchie des rôles

- Si le rôle r_1 hérite du rôle r_2 avec r_1 et r_2 qui sont distincts, alors r_2 ne peut pas hériter r_1.
- Deux rôles distincts assignés à un même usager ne peuvent hériter (directement ou indirectement) l'un de l'autre.

7.1.3 Contraintes de séparation des tâches

- Deux rôles assignés à un même usager ne sont pas en séparation statique de tâches.
- Un rôle ne peut pas être en séparation statique avec lui-même.
- La séparation statique des tâches est une relation symétrique.
- Si un rôle hérite (directement ou indirectement) d'un autre rôle et le rôle hérité est en séparation de tâches statique avec un troisième rôle, alors le rôle qui hérite est aussi en séparation statique de tâches avec le troisième rôle.
- Si un rôle hérite d'un autre rôle junior, alors l'ensemble des usagers en conflit d'assignation du rôle hérité (junior) est un sous-ensemble de l'ensemble des usagers en conflit d'assignation du rôle qui hérite.
- Seulement un usager de l'ensemble des usagers en conflit d'assignation pour un rôle r, peut être assigné le rôle r (Exclusion mutuelle statique entre usagers).
- Si deux rôles distincts r_1 et r_2 où $r_2 \leq r_1$ ont un nombre d'usagers en conflit d'assignation qui est commun, il y a seulement un usager de l'ensemble commun qui peut être assigné l'un ou l'autre des deux rôles (r_1 et r_2), et pas les deux (Exclusion mutuelle statique entre rôles).

- L'ensemble de rôles activés par un usager est un sous-ensemble des rôles auxquels il est autorisé.
- Deux rôles en séparation dynamique de tâches n'appartiennent pas à l'ensemble des rôles actifs d'un même usager.
- La séparation dynamique et la séparation statique sont des ensembles disjoints.
- Un rôle ne peut pas être en séparation dynamique avec lui-même.
- La séparation dynamique des tâches est une relation symétrique.
- Si un rôle hérite d'un autre rôle junior, alors l'ensemble des usagers en conflit d'activation du rôle hérité (junior) est un sous-ensemble de l'ensemble des usagers en conflit d'activation du rôle qui hérite.
- Si deux utilisateurs u_1 et u_2 sont en conflit d'activation pour un rôle r, alors celui-ci ne peut pas être activé simultanément par u_1 et u_2 .

7.2 Modélisation de RBAC avec des réseaux de Petri colorés

Nous avons investigué l'idée d'utiliser les réseaux de Petri tel que proposé dans (Shafiq et al., 2005) pour modéliser l'administration des rôles selon une approche modulaire en intégrant certaines caractéristiques du modèle GTRBAC. En effet, nous distingons entre différents états d'un rôle "*assigned*/*deassigned*", "*enabled*/*disabled*" et "*active*/*inactive*". Le modèle évolue lorsque des évènements (commandes) surviennent suite à des actions de l'usager ou de l'administrateur ou en réponse à d'autres actions. Nous présentons ci-après, la traduction d'un modèle RBAC en réseaux de Petri colorés.

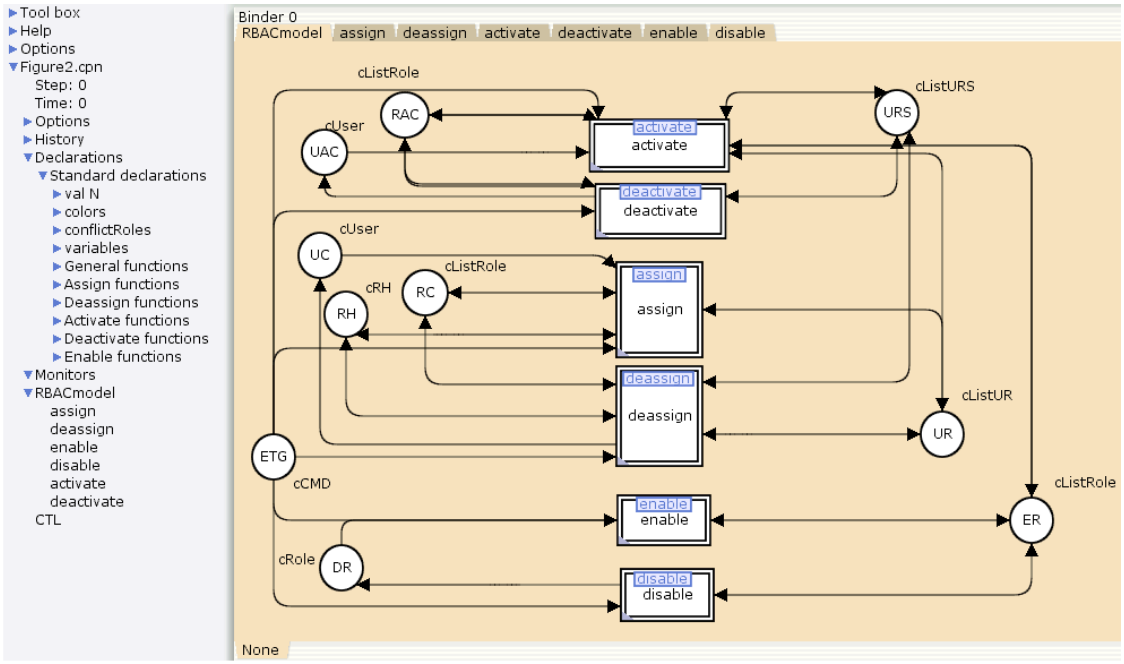


Figure 7.1 Représention du modèle RBAC en CPN

7.2.1 Structure du réseau et déclarations

Dans cette section, nous allons décrire le modèle RBAC en utilisant les réseaux de Petri colorés. La structure du réseau CPN se compose de sept modules (pages) (Jensen et al., 2007). Au plus haut niveau, on retrouve le modèle RBAC illustré à la figure 7.1, où les six principales fonctions sont représentées par des transitions de substitution. De cette façon, chaque fonction est modélisée par un sous-module: *Assign*, *De-Assign*, *Activate*, *Deactivate*, *Enable* et *Disable* respectivement.

7.2.1.1 Domaines de couleurs Δ

Dans notre représentation du modèle RBAC, nous avons défini des jetons de couleurs différentes qui contiennent diverses informations. Nous représentons les ensembles de

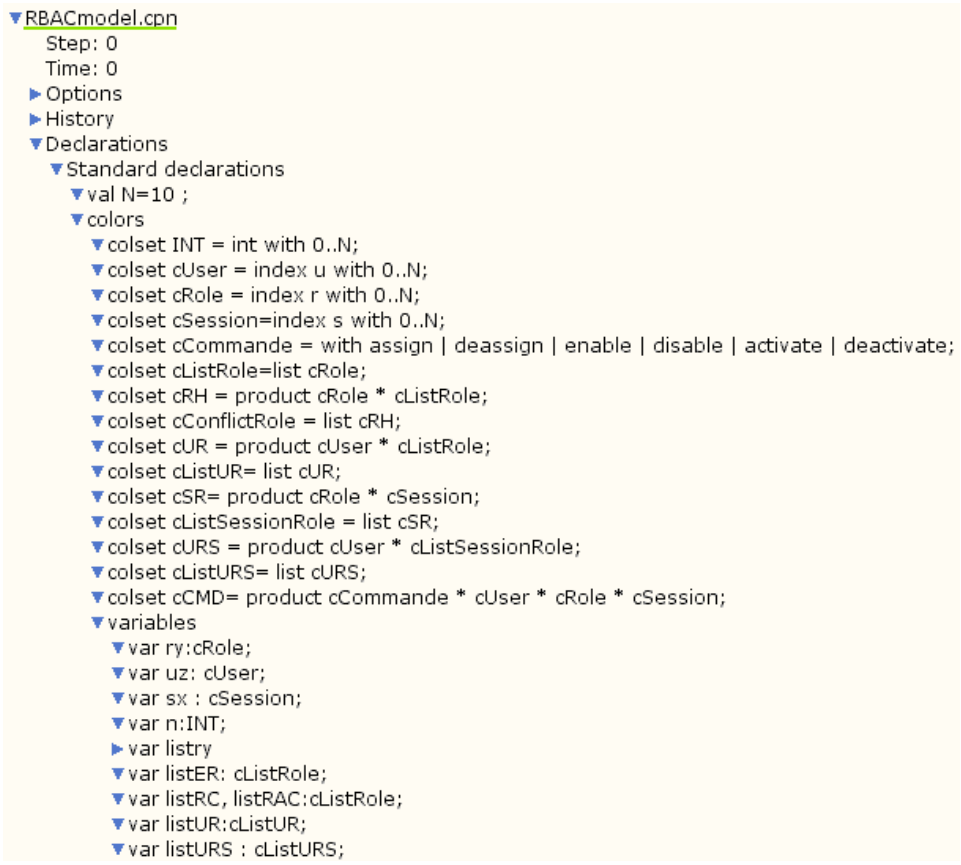


Figure 7.2 Ensemble de couleurs et variables du modèle RBAC en CPN

sujets, de rôles et de sessions respectivement par les couleurs $cUser$, $cRole$ et $cSession$. Les autres couleurs sont une composition de ces couleurs atomiques comme on peut le voir sur la figure 7.2.

- La couleur "*Role Hierarchy*", notée cRH définit la hiérarchie des rôles en associant à chaque rôle r_y, la liste de ses rôles juniors $listr_y$ de couleur $cListRole$ qui est une liste de rôles.

- La couleur "*User Role Assignment*", notée cUR est composé d'un sujet u_z et de la liste des rôles auxquels il est autorisé (de couleur $cListRole$).

- La couleur "*Session Role*", notée cSR contient un rôle et la session à laquelle il est

activé.

- La couleur "*User Role Session activation*", notée $cURS$ se compose d'un sujet et de l'ensemble des rôles qu'il a activé avec les sessions correspondantes.

- La couleur "*Command*", notée $cCMD$ représente l'ensemble des commandes à exécuter dans le modèle. C'est une composition des couleurs $cCommand$, $cUser$, $cRole$ et $cSession$. Par exemple, une commande "*assign (user, role)*" peut être considérée comme un vecteur *(assign, user, role)*. La même chose pour les autres commandes *(desassign, user, role)*, *(enable, role)*, *(disable, role)*, *(activate, user, role, session)* et *(deactivate, user, role, session)*. Pour généraliser, nous utilisons la couleur $cCommand$ pour représenter les différentes actions "*assign*", "*deassign*", "*enable*", "*disable*", "*activate*" et "*deactivate*".

Par la suite, à partir de l'ensemble des domaines de couleurs déclarés, nous définissons les jetons suivants:

- $<p, u>$ est le jeton sujet où p est la place où il réside et $u \in cUser$.
- $<p, r>$ est le jeton rôle dans la place p avec $r \in cRole$.
- $<p, r, list_r>$ exprime une relation de hiérarchie entre rôles. Tous les rôles juniors d'un rôle r sont représentés par une liste de rôles $list\_r$ (y compris r).
- $<p, u, r>$ est un jeton qui représente un rôle r (à l'état "*assigned*") assigné au sujet u.
- $<p, u, r, s>$ est un jeton qui représente un rôle r à l'état actif par un sujet u à la session s.
- $<p, cmd, u, r, s>$ est un jeton de commande de couleur $cCMD$.

7.2.1.2 Places du réseau P

Les places capturent les états des différents éléments du modèle RBAC, à savoir les différents états des rôles, les différentes commandes à exécuter et les différentes cardinalités.
La place ETG "*Event Token Generator*" est une place qui conserve les jetons de commandes pour assigner (désaffecter) des rôles aux usagers, rendre un rôle disponible ou indisponible, activer ou désactiver un rôle. Pour qu'une transition soit sensibilisée, il faut qu'il y ait un jeton dans cette place. Ainsi, elle joue le rôle d'un contrôleur qui permet d'analyser le comportement du système suite au lancement des différentes commandes (évènements externes ou internes).
Les places RC (*Role Cardinality*), UC (*User Cardinality*), RAC (*Role Activation cardinality*) et UAC (*User Activation cardinality*) sont utilisées pour exprimer les différentes contraintes de cardinalité.

- La place RC conserve une liste de jetons de rôles ($C(RC) = cListRole$). Elle limite le nombre de sujets autorisés par rôle. S'il y a k jetons de couleur r_y dans cette place et aucun sujet n'est assigné le rôle r_y, il y a donc au plus k usagers qui peuvent être assignés le rôle r_y. Par exemple, $<RC, [r_0, r_0, r_0, r_1, r_1]>$ signifie que les rôles r_0 et r_1 peuvent être assignés à maximum 3 et 2 sujets respectivement.

- La place UC conserve les jetons de sujets. Elle limite le nombre de rôles permis pour un sujet. S'il y a l jetons de couleur u_z dans cette place et aucun rôle n'a été assigné à l'usager u_z, il peut donc endosser au plus l rôles. Par exemple, $3<UC, u_0>$ + $2<UC, u_1>$ signifie que les sujets u_0 et u_1 sont autorisés à maximum 3 et 2 rôles respectivement.

- La place RAC conserve une liste de jetons de rôles. Elle limite le nombre d'activations simultanées de certains rôles. S'il y a m jetons r_y dans cette place, cela veut

dire que m copies de r_y au maximum peuvent être activées simultanément.

- La place UAC conserve les jetons de type $cUser$. Elle limite le nombre d'activations concurrentes de rôles pour un sujet. S'il y a j jetons u_z dans UAC, cela veut dire que l'usager u_z peut effectuer, au plus, j activations concurrentes. Ces activations peuvent impliquer un même rôle plusieurs fois ou divers rôles pour un nombre de fois donné tel que le nombre total des activations concurrentes n'excède pas j.

Pour distinguer entre les différents états d'un rôle (assigned/deassigned, enabled/disabled et active/inactive), nous utilisons les places UR (*User Role Assignment/Authorization*), DR (*Disabled Roles*), ER (*Enabled Roles*) et URS (*User Role Session activation*) pour capturer ces différents états.

- La place UR conserve des jetons de rôles assignés aux sujets. La présence d'un jeton $<UR, u_z, listRole>$ dans cette place signifie que l'usager u_z est autorisé à tous les rôles de $listRole$.

- La place ER conserve une liste de jetons de rôles. Une liste $[r_1,r_2]$ dans ER indique que les rôles r_1 et r_2 sont à l'état "*enabled*".

- La place DR conserve des jetons de rôles (C(DR) = $cRole$) à l'état "*disabled*" state. Initiallement, tous les rôles se trouvent à cet état.

- La place URS conserve la liste des sujets ayant activés des rôles pendant des sessions données. Par exemple, le jeton $<URS,u_z, listSessionRole>$ signifie que le sujet u_z assume les rôles qui figurent dans la liste $listSessionRole$ aux sessions correspondantes.

Finalement, nous représentons la hiérarchie des rôles par la place RH (*Role Hierarchy*). Cette place contient des jetons de couleur (C(RH) = cRH) pour exprimer les relations de

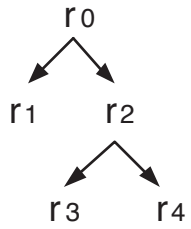


Figure 7.3 Hiérarchie de rôles

hiérarchie courantes du système. Par exemple, les relations de hiérarchie de la figure 7.3 sont représentées comme suit : $<RH,r_0,listr_0>$ + $<RH,r_1,listr_1>$ + $<RH,r_2,listr_2>$ + $<RH,\ r_3,\ listr_3>$ + $<RH,\ r_4,\ listr_4>$ où $listr_0$ = $[r_0..r_4]$, $listr_1$ = $[r_1]$, $listr_2$ = $[r_2..r_4]$, $listr_3$ = $[r_3]$, $listr_4$ = $[r_4]$.

7.2.1.3 Arcs, expressions sur les arcs et gardes

Les arcs, les expressions sur les arcs et les gardes modélisent les différentes contraintes de cardinalité, d'héritage et de séparation de tâches à satisfaire pour assurer la cohérence du modèle. Les expressions sur les arcs et les gardes sont exprimées par des fonctions écrites en langage ML(Jensen et al., 2007). L'utilisation du langage ML dans CPNtools permet de construire des fonctions de haut niveau pour exprimer les gardes et la transformation des jetons. Nous distinguons entre des fonctions générales et des fonctions spécifiques à chaque transition de substitution. À titre d'exemple, nous montrons dans le tableau 7.1 l'ensemble des fonctions générales.

7.2.1.4 Transitions du réseau

Les transitions représentent les différentes actions (évènements): *User-role assignment* / *deassignment*, *Role enabling* / *disabling*, *Role activation* / *deactivation*. Ainsi, le réseau

Tableau 7.1 Fonctions générales utilisées dans la définition des propriétés de consistence

exist_role(r_y*, listRole)*	Évaluée à true si $r_y \in$ listRole
exist_user(uz, listUR)	Évaluée à true si u_z est assigné un rôle
exist_role_session(ry, sx, listSR)	Évaluée à true si r_y est actif à la session s_x
get_conflict_roleset(ry, listConflict)	Retourne l'ensemble des rôles en conflit avec le rôle r_y
get_role_list(uz, listUR)	Retourne l'ensemble des rôles assignés à l'usager u_z
get_session_list(uz, listURS)	Retourne l'ensemble des rôles activés par l'usager u_z
add_session_role(ry, sx, listSR)	Ajoute un rôle nouvellement activé avec sa session à la liste $listSR$
remove_role_session(ry, sx, listSR)	Retire un rôle de la liste $listSR$ après sa désactivation à la session s_x.
removeUR(uz, listUR)	Retire un usager de la liste des usagers autorisés à des rôles
removeURS(uz, listURS)	Retire un usager de la liste des usagers actifs

coloré se compose de six pages liées hiérarchiquement.

7.2.2 Dynamique du modèle

Le modèle passe d'un état à un autre par franchissement de transitions. À partir du moment où une transition est sensibilisée, elle peut être franchie à n'importe quel instant. Une transition représente un évènement. Ainsi, lorsqu'une transition est sensibilisée, cela veut dire que les contraintes associées à cet évènement sont remplies. Les conditions de sensibilisation et de franchissement sont différentes selon les évènements. À titre d'exemple, la figure 7.4 montre la page de la transition "*assign*". Les places d'entrée et de sortie ont été affectées à leurs correspondantes dans le niveau principal du réseau. Cette transition assigne, par la commande $<ETG, assign, u_z, r_y, s_x>$, le rôle r_y au sujet u_z. La transition peut être franchie à tout moment si toutes les conditions sont satisfaites. Nous explicitons ci-après, les conditions associées à chacune des transitions.

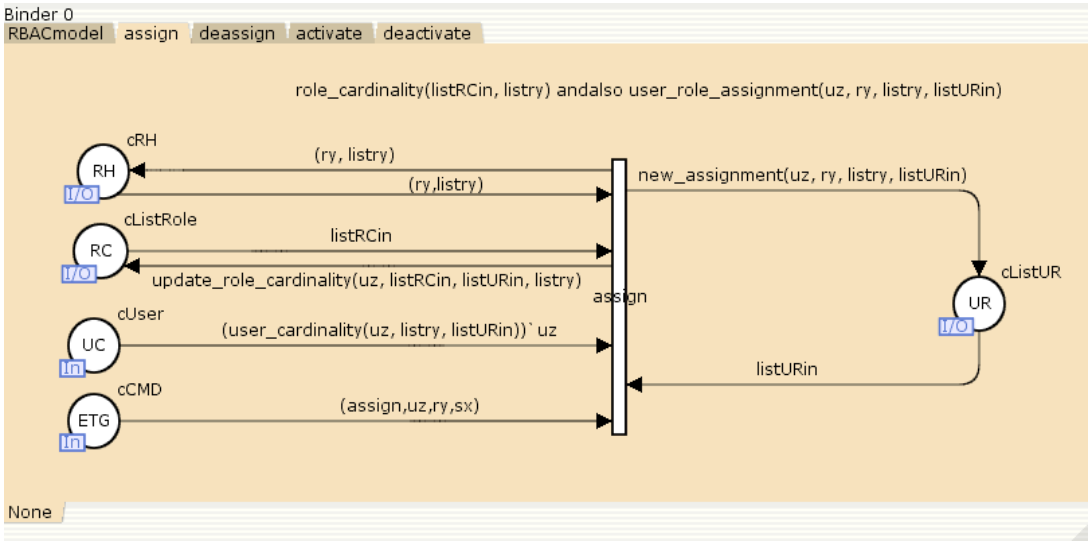


Figure 7.4 Représentation CPN de la transition *assign*

7.2.2.1 Transition *assign*

Tout d'abord, il doit y avoir un jeton de commande dans la place ETG. La place UC doit contenir au moins n jetons $<UC,u_z>$ pour satisfaire la cardinalité des sujets. Le nombre des jetons de sujets n est déterminé par la fonction *user_cardinality(uz, listry, listUR)*. Si le sujet u_z n'a aucun rôle assigné ou aucun rôle de ceux qui sont listés par $listr_y$, le franchissement consommerait "Size $listr_y$" jetons. Dans le cas où le sujet a déjà des rôles assignés, le franchissement consommera les jetons nécessaires pour assigner les nouveaux rôles et la cardinalité du sujet sera ajustée en conséquence. De plus, pour satisfaire la contrainte de cardinalité des rôles, il faut que la garde *role_cardinality*($listRC$, $listr_y$) soit évaluée $true$. Cette fonction vérifie, à partir des jetons de la place RC, que tous les rôles de la liste $listr_y$ ont une cardinalité suffisante. Finalement, la garde *user_role_assignment(uz, ry, listUR)* doit être satisfaite pour valider l'assignation du rôle r_y et ses rôles juniors au sujet u_z. Cette garde vérifie deux conditions, à savoir que le sujet u_z n'a pas déjà le rôle r_y dans sa liste de rôles assignés, et si le sujet u_z est déjà assigné d'autres rôles, on s'assure que r_y est en séparation statique avec

l'ensemble de ces rôles. Cette dernière condition est vérifiée au moyen des fonctions *direct_conflict_role_assign(ry, listRole)* et *indirect_conflict_role_assign(listry, listRole)*.

Si toutes les conditions sont satisfaites, le franchissement de la transition *assign* conduit à un nouvel état où un multi-ensemble de jetons sera créé dans la place UR calculé par la fonction *new_assignment(uz, ry, listry, listUR)*. Cette fonction autorise l'usager u_z à utiliser le rôle r_y et tous ses rôles juniors puisque dorénavant ils existent dans sa liste de rôles autorisés. Aussi, la cardinalité des rôles est mise à jour par la fonction *update_role_cardinality(uz, listRC, listUR, listry)*. Ainsi, par le biais de cette transition, l'usager u_z sera autorisé à r_y et tous les rôles qui héritent de r_y.

7.2.2.2 Transition *deassign*

En exécutant cette transition, l'usager u_z ne sera plus autorisé au rôle r_y ni aux rôles qui héritent de r_y (l'ensemble de ces rôles ne figurera plus dans la liste de rôles associée à l'usager u_z). Autrement dit, la transition *deassign*, illustrée à la figure 7.5, permet donc de supprimer un rôle et tous ses rôles juniors de la liste des rôles autorisés à un sujet. Pour cela, il y a essentiellement deux conditions à satisfaire :

- Le rôle à désaffecter est un rôle qui est préalablement assigné à l'usager,
- Ce rôle ainsi que ses rôles juniors ne sont pas à l'état actif,

Ces conditions sont respectivement exprimées par les gardes *deassign_user_role(uz, ry, listUR)* et *not_active_roles(uz, listry, listURS)*. Ainsi, pour que la transition soit franchie il faut avoir un jeton de commande dans la place ETG et les deux gardes doivent être évaluées à *true*. Après franchissement, on met à jour la liste des jetons de la place UR par la fonction *new_assigned_roles(uz ,listry,listUR)* de tel sorte que le rôle r_y et ses rôles juniors mémorisés dans la liste $listr_y$ seront retirés de la liste des rôles autorisés à

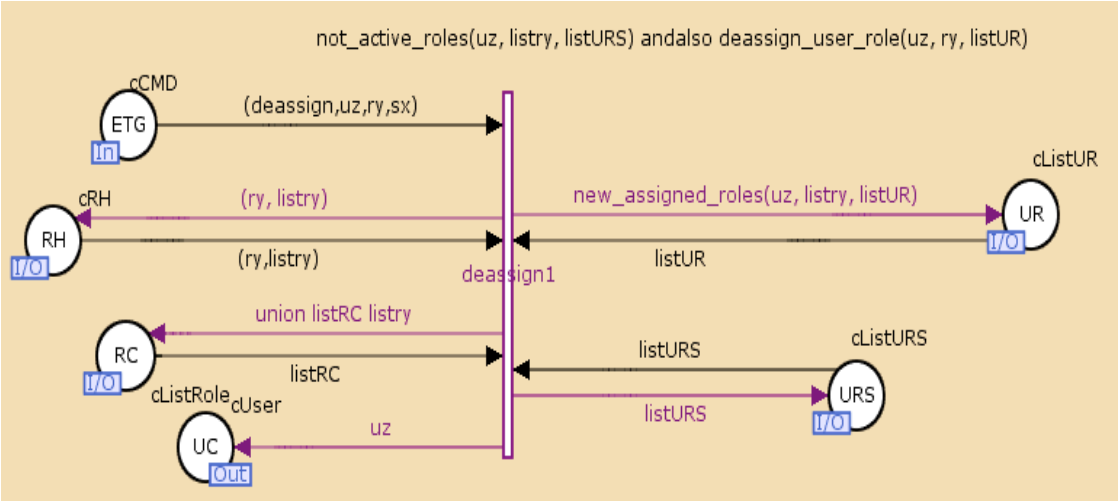


Figure 7.5 Représentation CPN de la transition *deassign*

l'usager u_z. On ajuste aussi la cardinalité des sujets en produisant un jeton u_z dans la place UC ainsi que la cardinalité des rôles dans la place RC en faisant l'union de deux listes *union listRC listry)*.

7.2.2.3 Transition *enable*

La transition *enable* rend le rôle r_y disponible pour qu'éventuellement il soit activé par un usager auquel il est assigné. Ainsi, pour que cette transition soit franchie, on vérifie qu'il existe un rôle r_y à l'état disable, c'est-à-dire qu'il y a un jeton dans la place DR. On vérifie aussi que le rôle n'est pas déjà à l'état enable par la garde *exist_role(ry ,listER)* qui doit être évaluée à false. De plus, on vérifie la séparation des tâches *noConflict_role_enabled(ry ,listER)* au cas où le rôle r_y serait en conflit avec d'autres rôles s'ils sont mis disponibles en même temps (figure 7.6). Le franchissement de cette transition va insérer un jeton r_y dans la liste des jetons disponibles dans la place ER. À partir de ce moment, le rôle r_y peut être activé par tout sujet autorisé.

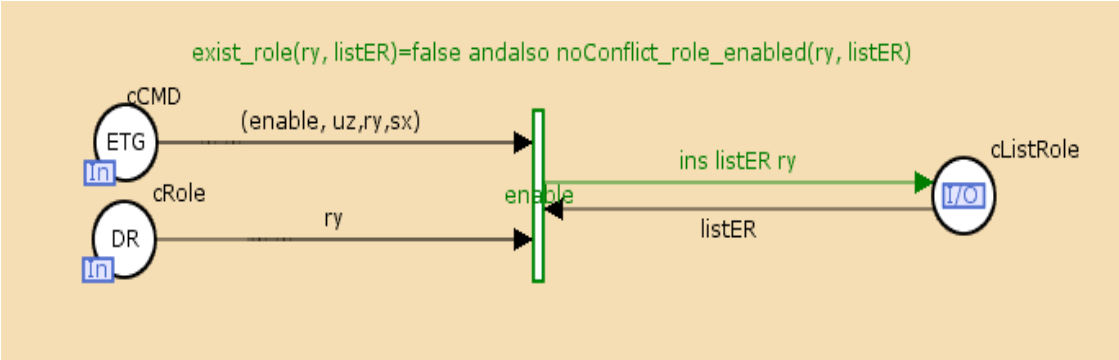


Figure 7.6 Représentation CPN de la transition *enable*

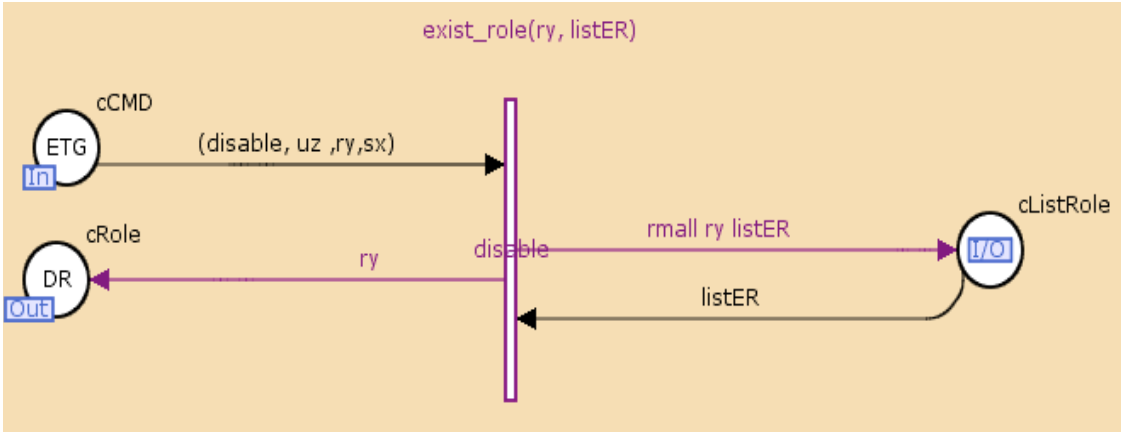


Figure 7.7 Représentation CPN de la transition *disable*

7.2.2.4 Transition *disable*

La transition *disable*, représentée à la figure 7.7, modifie l'état d'un rôle *enable* et l'état *disable*. Par conséquent, le rôle ne peut être activé par aucun usager. Pour qu'elle soit franchie, on vérifie qu'il existe un rôle r_y à l'état enable par la garde *exist_role(ry ,listER)*. Si la garde est satisfaite, la transition sera franchie, le jeton r_y sera retiré de la liste des rôles disponibles et sera déposé dans la place DR.

7.2.2.5 Transition *activate*

La transition activate, de la figure 7.8, fait débuter une session active entre l'usager uz et le rôle r_y. Ce dernier passe à l'état actif. Pour qu'un rôle puisse être activé, il faut qu'il soit d'abord assigné à un usager et à l'état enable. Ceci est vérifié respectivement par les gardes *authorized_roles(uz ,ry,listUR)* et *exist_role(ry ,listER)*. Il faut aussi respecter les contraintes de cardinalité suivantes : *User Activation Cardinality*, *Role Activation Cardinality* et *User Session Cardinality*. Dans notre modélisation, pour fin de simplification, nous prenons en compte les deux premières contraintes seulement. Toutefois, il est possible d'exprimer la contrainte qui limite le nombre de sessions activées par un usager en ajoutant une autre place, notée USC de couleur $cListUser$ (une liste de $cUser$). Cette place identifie pour chaque usager un nombre de sessions maximal. À chaque fois qu'un usager est affecté à une session, on retirera un jeton de sa liste. À l'inverse, si un usager a terminé une session, on ajoutera un jeton dans sa liste. Les autres contraintes sont vérifiées par la garde *role_activation_cardinality(ry ,listRAC)* qui doit être satisfaite ainsi que la présence d'un jeton u_z dans la place UAC. Finalement, la garde *user_role_activation(uz ,ry,sx,listURS)* doit être satisfaite pour valider l'activation du rôle r_y par le sujet u_z à la session s_x . Cette garde vérifie deux conditions, à savoir que le sujet u_z n'a pas déjà activé le rôle r_y à la session s_x, mais dans le cas où r_y est déjà activé par u_z, on s'assure que r_y est en séparation dynamique avec l'ensemble des rôles préalablement activés par u_z.

Si toutes les conditions sont satisfaites, le franchissement de la transition *activate* conduit à un nouvel état où un multi-ensemble de jetons sera créé dans la place URS calculé par la fonction *new_activation(uz, ry, sx,listURS)*, et d'autres jetons seront consommés des places RAC et UAC pour ajuster la cardinalité.

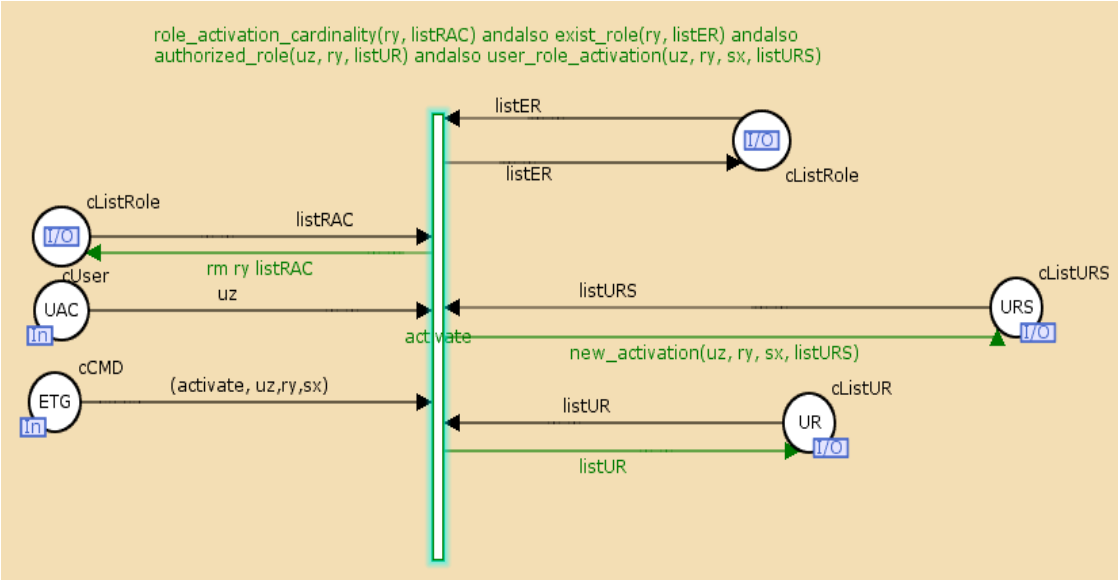


Figure 7.8 Représentation CPN de la transition *activate*

7.2.2.6 Transition *deactivate*

Cette transition met fin à la session active entre l'usager u_z et le rôle r_y. Ce dernier passe à l'état inactif. Pour désactiver un rôle, on vérifie d'abord que celui-ci est déjà à l'état actif par le rôle u_z à la session s_x. La vérification est exprimée par la garde *deactivate_role(uz, ry, sx, listURS)*. Si la condition est satisfaite, la transition peut être franchie, son franchissement met à jour la liste des rôles activés par l'usager u_z calculée par la fonction *new_deactivation(uz, ry, sx, listURS)*, produit des jetons dans les places RAC et UAC pour ajuster la cardinalité.

7.3 Vérification de la consistance du modèle RBAC

L'analyse d'un réseau de Petri peut révéler des caractéristiques importantes d'un système, concernant sa structure et son comportement dynamique. Les résultats de cette analyse sont utilisés pour évaluer le système et en permettre la modification et/ou l'amélioration

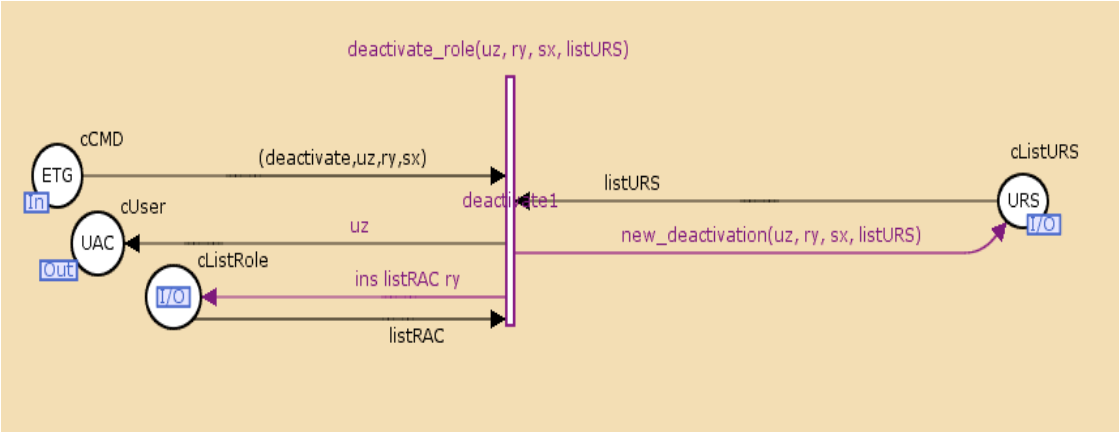


Figure 7.9 Représentation CPN de la transition *deactivate*

le cas échéant.

Le présent modèle est défini par son marquage. Ainsi, la vérification de la cohérence du modèle RBAC est effectuée à partir d'une analyse de chaque état accessible du réseau de Petri coloré. L'état est dit cohérent s'il satisfait à toutes les contraintes de cardinalité, de séparation de tâches et d'héritage. Il s'agit ainsi d'effectuer une analyse d'accessibilité qui repose sur le graphe des marquages correspondant au réseau de Petri. En s'assurant de la cohérence du réseau de Petri, on s'assure également de la garantie des différentes propriétés de sécurité liées à la politique de sécurité.

Les anomalies auxquelles on s'intéresse peuvent être exprimées par des propriétés d'accessibilité pour déceler la redondance et l'incomplétude des règles, des propriétés de sûreté pour vérifier l'inconsistance, ou des propriétés de vivacité pour la présence d'états de blocage. Ces anomalies peuvent être détectées en procédant à une analyse du graphe de marquages. L'outil CPNtools offre un module de génération de graphes d'occurrence sur des modèles de RdP colorés.

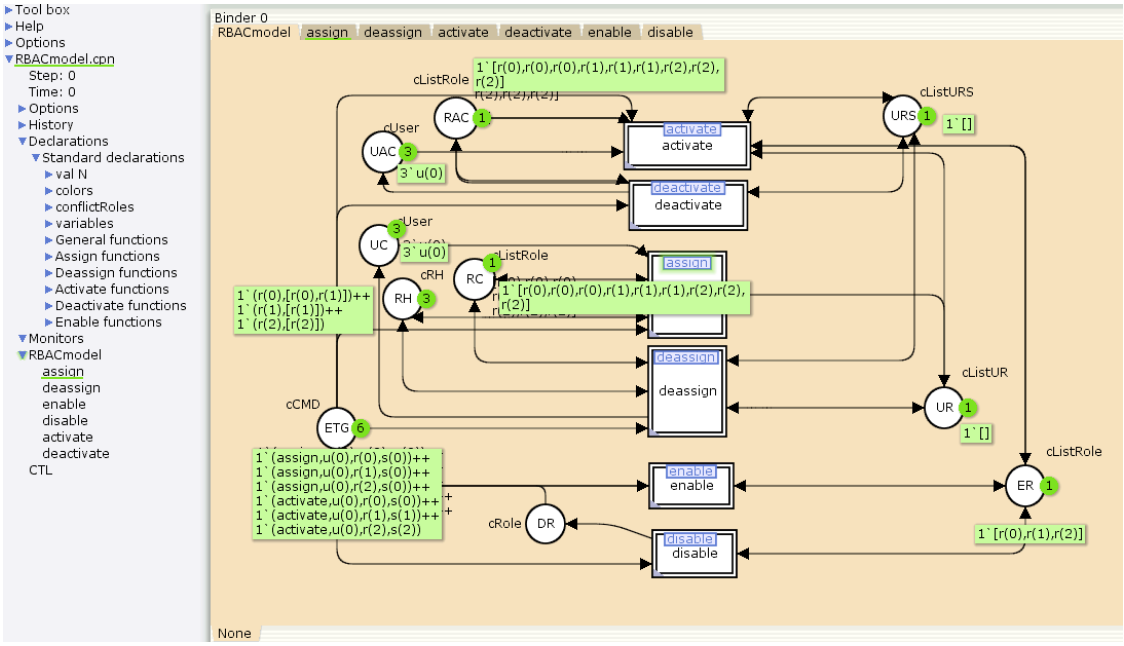


Figure 7.10 Exemple d'une représentation CPN du modèle RBAC

7.3.1 Exemple

Pour illustrer l'utilisation des graphes d'occurence dans la vérification de la cohérence du modèle RBAC, nous considérons l'exemple suivant, tiré de (Shafiq et al., 2005) et représenté à la figure 7.10.

Dans cet exemple, il y a trois rôles r_0, r_1 et r_2 ainsi qu'un sujet u_0. Les spécifications initiales du modèle sont telles que:

- r_1 est un rôle junior à r_0 ($r_1 \leq r_0$ et $r_1 \neq r_0$).
- r_1 et r_2 sont deux rôles en conflit d'assignation, i.e., r_1 et r_2 ne peuvent pas être assignés aux même sujet, ni même activés simultanément par le même sujet.
- Chaque rôle peut être assigné à 3 sujets au plus.

- Chaque rôle peut être activé par au plus 3 sujets.
- Le sujet u_0 peut être assigné au plus 3 rôles.
- Le sujet u_0 peut activer au plus 3 rôles.

Les spécifications suivantes sont traduites au niveau de la représentation CPN comme on peut le voir sur la figure 7.10. De plus, nous identifions les rôles en conflit d'assignation et d'activation par les valeurs *conflicting_role_assignset* et *conflicting_role_activeset* respectivement. Ces valeurs sont représentées par une liste où chaque élément se compose d'un rôle et de la liste des rôles avec lesquels il est en conflit. Dans le cas présent, nous déclarons :

- *val conflicting_role_assignset* = [(r(1), [r(2)]), (r(2), [r(1)])] et
- *val conflicting_role_activeset* = [(r(1), [r(2)]), (r(2), [r(1)])].

Dans un souci de simplification, nous considérons que tous les rôles sont initialement à l'état *enabled* et ils y demeureront, la figure 7.11 montre le rapport généré suite à l'analyse de l'espace d'état de cet exemple.

Le rapport contient beaucoup d'informations utiles relatives au comportement du modèle CPN. Le rapport est un excellent moyen pour repérer les erreurs et constitue une source de référence nécessaire pour exprimer correctement les formules CTL et vérifier les propriétés souhaitées.

7.3.2 Vérification de la cohérence

L'analyse s'avère très considérable dans la mesure où elle permet de révéler des caractéristiques importantes du système modélisé, concernant sa structure et son com-

```
 Statistics
------------------------------------------------------------------------
  State Space
     Nodes:  13
     Arcs:   13
     Secs:   0
     Status: Full
---------------------- Boundedness Properties---------------------------
  Best Integer Bounds   Upper      Lower
     RBACmodel'ER 1          1          1
     RBACmodel'ETG 1         6          2
     RBACmodel'RC 1          1          1
     RBACmodel'RH 1          3          3
    ....................................
  Best Upper Multi-set Bounds
     RBACmodel'ER 1      1`[r(0),r(1),r(2)]
     RBACmodel'RC 1      1`[r(0),r(0),r(0),r(1),r(1),r(1),r(2),r(2)]++
                        1`[r(0),r(0),r(0),r(1),r(1),r(1),r(2),r(2),r(2)] ++
                        1`[r(0),r(0),r(0),r(1),r(1),r(2),r(2),r(2)] ++
                        1`[r(0),r(0),r(1),r(1),r(2),r(2),r(2)]
     RBACmodel'UR 1       1`[]++ 1`[(u(0),[r(0),r(1)])]++ 1`[(u(0),[r(1)])]++ 1`[(u(0),[r(2)])]
     RBACmodel'URS 1      1`[]++ 1`[(u(0),[(r(0),s(0))])]++ 1`[(u(0),[(r(1),s(1))])]++
                        1`[(u(0),[(r(1),s(1)),(r(0),s(0))])]++ 1`[(u(0),[(r(2),s(2))])]
   ....................................
  Best Lower Multi-set Bounds
     RBACmodel'ER 1      1`[r(0),r(1),r(2)]
     RBACmodel'RH 1      1`(r(0),[r(0),r(1)]) ++ 1`(r(1),[r(1)])++ 1`(r(2),[r(2)])
     RBACmodel'UC 1      1`u(0)
     RBACmodel'UR 1      empty
     RBACmodel'URS 1     empty
   ....................................
 -------------------------Home Properties-------------------------------
  Home Markings : None
------------------------Liveness Properties-----------------------------
  Dead Markings : [7,8,11,12,13]
  Dead Transition Instances
     deactivate'deactivate1 1
     deassign'deassign1 1
     disable'disable1 1
     enable'enable 1
  Live Transition Instances : None
 ----------------------Fairness Properties-----------------------------
     No infinite occurrence sequences.
```

Figure 7.11 Rapport d'analyse de l'espace des états du modèle CPN de la figure 7.10

Tableau 7.2 Fonctions d'accessibilité

Fonction	**Description**
Mark. <PageName>' <PlaceName> N M	Fonction qui retourne l'ensemble des jetons qui se trouve à la place<*PlaceName*> de la N^{eme} instance de la page <*PageName*> du marquage M.
SearchNodes (<search area>, <predicate function>, <search limit>, <evaluation function>, <start value>, <combination function>)	Fonction qui parcourt tous les noeuds de l'espace des états spécifié dans <search area> et applique sur chaque noeud, la fonction <evaluation function>. Les résultats de cette opération sont combinés selon <combination function>. La fonction <predicate function> associe chaque noeud à une valeur booléenne et ne conserve que ceux dont la valeur est évaluée à $true$. Pour parcourir tous les noeuds du graphe, on précise la valeur $EntireGraph$ au niveau de <search area> et la valeur $NoLimit$ au niveau de <search limit> pour poursuivre la recherche de tous les noeuds dont <predicate function> est évaluée à *true*.
List.nth(l,n)	Retourne le n^{eme} élément de la liste l avec $0 \leq n < lengthl$.

portement dynamique. L'intérêt de la modélisation est de procéder à la vérification de propriétés spécifiques telles que les propriétés de sécurité. CPNtools offre des fonctions d'exploration pour récupérer les marquages qui répondent à certains critères. Le tableau 7.2 montre un exemple de trois fonctions.

Nous montrons que les réseaux de Petri représentent un formalisme intuitif pour aider à la détection des incohérences et incomplètudes dans les règles d'une politique de sécurité RBAC. Les anomalies auxquelles on s'intéresse peuvent être exprimées par des propriétés d'atteignabilité pour déceler la redondance et l'incomplétude des règles, des propriétés de sûreté pour vérifier l'inconsistance, ou des propriétés de vivacité pour la présence d'états de blocage. Ces anomalies peuvent être détectées en procédant à une

analyse du graphe de marquages calculé à partir du réseau de Petri modélisant la politique RBAC.

Supposons, par exemple, qu'on souhaite vérifier les règles suivantes:
- Deux rôles assignés à un même usager ne sont pas en séparation statique de tâches.
- Deux rôles en séparation dynamique de tâches n'appartiennent pas à l'ensemble des rôles actifs d'un même usager.
- L'ensemble de rôles activés par un usager est un sous-ensemble des rôles auxquels il est autorisé.

Selon l'exemple courant, ces règles se traduisent respectivement par:
- u_0 ne doit pas être assigné les rôles r_1 et r_2 en même temps.
- u_0 ne peut pas activer r_1 et r_2 simultanément.
- Tous les rôles activés par u_0 doivent figurés dans sa liste de rôles autorisés.

Pour vérifier si tous les états accessibles sont consistants avec les trois règles précédentes, nous avons exprimé chacune des règles en fonctions ML tel que représenté à la figure 7.12.

La fonction *assigned_roles* retourne la liste des rôles autorisés à chaque sujet. Chaque élément de cette liste correspond aux rôles autorisés à l'usager u_0, et ce pour chaque état accessible. La fonction *active_roles* retourne la liste des rôles activés par l'usager u_0, et ce pour chaque état accessible. La fonction *authorized_roles* est évaluée sur chaque état accessible pour vérifier si un rôle à l'état actif est aussi un rôle autorisé. La fonction retourne $true$ si cette condition est satisfaite par tout état accessible du modèle, et faux sinon. Pour cet exemple, nous obtenons que les états accessibles ne présentent aucune violation avec la spécification initiale.

Dans le cas où une règle est violée, il est possible de rechercher les causes du problème, par exemple, en examinant de près les états qui ne satisfont pas à la règle et les chemins

Figure 7.12 Model-checking des règles de consistence du modèle RBAC

menant à ces états. Cela donnera une idée pour localiser la source du problème. Le modèle devra ensuite être modifié en conséquence, et les propriétés devront être revérifiées jusqu'à ce qu'elles soient satisfaites. L'erreur pourrait également provenir d'une mauvaise expression des propriétés à partir de la spécification informelle. Dans ce cas, les propriétés devront être réécrites et revérifiées de nouveau (Choppy et al., 2007). Si on arrive à obtenir un modèle de réseau de Petri coloré avec la plupart des propriétés satisfaites, une analyse plus approfondie peut être effectuée, conduisant à d'éventuels changements dans la spécification. Une modélisation formelle du modèle RBAC fournit une description précise du système de sécurité.

7.4 Conclusion

Le modèle RBAC se prête naturellement à la protection des systèmes d'information coopératifs qui mettent en oeuvre des techniques de workflow et plus généralement de groupware (logiciels de groupes composés d'opérations différentes et multiples). De plus, dans plusieurs travaux de recherches (Oren and Haller, 2005), les réseaux de Petri se sont avérés adéquats et ont été utilisés avec succès pour la spécification des applications Workflow et la description de comportements dynamiques complexes. Comme exemple d'application, nous proposons une approche globale, basée sur les réseaux de Petri, pour traiter la gestion des autorisations dans les Workflows où le modèle du Workflow est greffée au modèle RBAC. L'idée est de représenter les applications du système Workflow de façon modulaire (sur différents modules) à l'aide de CPNtools, et grce au modèle RBAC, le système accorde à un utilisateur l'ensemble des ressources (autorisations) nécessaires pour effectuer une tâche particulière. Nous expliciterons cette idée plus en détail dans la section 7.5 du chapitre 7.

En conclusion, nous ajoutons aussi que dans (Rakkay and Boucheneb, 2007), nous avons montré qu'une approche à base des réseaux de Petri colorés temporisés est aussi pertinente pour la vérification et la validation des modèles à base à des rôles aussi bien en termes de rigueur que de souplesse. La dimension temporelle est intéressante à intégrer dans le modèle, notamment pour prévenir l'utilisation abusive des privilèges d'accès. Le modèle GTRBAC définit deux types de contraintes temporelles pour capturer le comportement dynamique du système: les contraintes de durée et les contraintes de périodicité.

Les contraintes de périodicité sont utilisées pour définir des intervalles exacts au cours desquels un rôle peut être activé (ou désactivé), ou encore des intervalles au cours desquels des actions d'assignation d'un rôle à un usager (ou de privilèges à un rôle) sont

valides. Les contraintes de durée expriment la durée où l'assignation (ou l'activation) d'un rôle est valide ou encore la durée de disponibilité d'un le rôle pour une éventuelle activation. Une manière simple pour intégrer le temps consiste à ajouter des temporisations qui sont supportées par l'outil CPNtools. On considère, dans ce cas, une horloge globale et les jetons sont estampillés par la date à laquelle ils sont prêts.

CHAPITRE 8

RÉSEAUX DE PETRI TEMPORELS : MODÈLE $TA_{WS}PN$ *TIMED ARC PETRI NET-WEAK AND STRONG SEMANTICS*

Pour intégrer le temps aux réseaux de Petri, de nombreuses extensions ont été proposées. Nous distinguons deux grandes classes : les réseaux temporisés (Ramchandani, 1974) qui associent une durée à un type de nœud (places, transitions ou arcs) et les réseaux temporels (Merlin, 1974; Berthomieu and Diaz, 1991; Walter, 1983; Khansa, 1997) qui associent un intervalle de temps. Par ailleurs, les réseaux temporisés sont une sous-classe des réseaux temporels.

Dans le présent chapitre, nous nous intéressons aux réseaux temporels. Ces réseaux diffèrent entre eux par l'élément auquel est associée la notion de temps: arcs (réseaux de Petri A-temporels), places (réseaux de Petri P-temporels) ou transitions (réseaux de Petri T-temporels). Chaque extension associe une interprétation spécifique aux contraintes temporelles. Dans les réseaux T-temporels, une contrainte temporelle exprime soit la durée de franchissement ou le temps de sensibilisation de la transition. Dans les réseaux P-temporels, le temps exprime le temps de séjour d'un jeton dans la place avant qu'il ne devienne disponible. Dans les réseaux A-temporels, si la contrainte est associée à un arc entrant dans une transition, elle indique l'intervalle de temps durant lequel un jeton peut être consommé, alors que si la contrainte est associée à un arc sortant, elle indique la disponibilité du jeton produit.

Il existe également deux sémantiques usuelles pour les réseaux de Petri temporels : la sémantique dite faible (*Weak Time Semantics* WTS) (Walter, 1983) et celle dite forte (*Strong Time Semantics* STS) (Merlin, 1974; Khansa, 1997). Dans une sémantique faible, les contraintes de temps constituent des instants possibles d'occurrences d'événe-

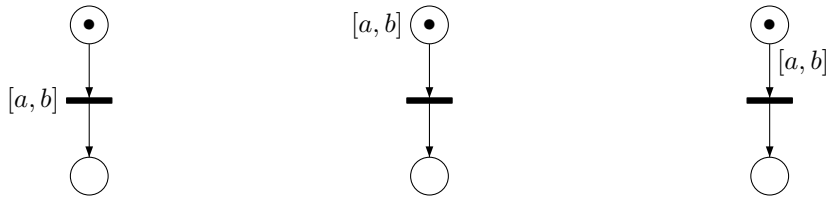
ments (autrement dit, les bornes temporelles supérieures peuvent être dépassées) alors que pour la sémantique forte, l'événement doit nécessairement se produire dans l'intervalle donné.

Dans une perspective de disposer d'un modèle qui permet d'intégrer plus d'une seule sémantique, nous avons enrichi la classe des modèles temporels en proposant une nouvelle extension des réseaux A-temporels généralisés aux sémantiques forte et faible, que nous identifions par l'acronyme $TA_{WS}PN$ (*Timed Arc Petri net - Weak and Strong semantics*)

Dans ce chapitre, nous allons présenter les différentes extensions temporelles, les plus répandues. Par la suite, nous donnerons la définition du modèle $TA_{WS}PN$ et sa sémantique. Nous présenterons après, un aperçu global d'une approche de modélisation des systèmes Workflows au moyen du modèle $TA_{WS}PN$ et leur analyse par model-checking. Nous montrons aussi comment y inclure le contrôle d'accès basé sur les rôles. Enfin, nous concluons ce chapitre par une discussion sur la modélisation des mécanismes de sécurité dans les Workflows Inter-Organisationnels au moyen des réseaux de Petri.

8.1 Réseaux de Petri temporels

Parmi les extensions temporelles des réseaux de Petri, nous considérons celles qui associent des intervalles de temps aux places selon les deux sémantiques, forte ou faible, notées respectivement TP_SPN et TP_WPN, également aux transitions (réseaux de Petri T-temporels TT_SPN, TT_WPN) et aux arcs en entrée des transitions (réseaux de Petri A-temporels TA_SPN, TA_WPN).



a) Réseau de Petri T-temporel b) Réseau de Petri P-temporel c) Réseau de Petri A-temporel

Figure 8.1 Réseaux de Petri temporels

8.1.1 Réseaux de Petri T-temporels

Le modèle de Merlin (TT_SPN) (Merlin, 1974; Berthomieu and Diaz, 1991) est l'une des extensions les plus connues et les plus utilisées. C'est un réseau de Petri qui associe un intervalle de temps $[a, b]$, à chaque transition (figure 8.1.a). Les bornes de cet intervalle sont les temps d'attente minimal et maximal avant le franchissement de la transition. Autrement dit, si la transition est sensibilisée de façon continue durant au moins a unités de temps, elle peut être tirée. Par contre, si la transition est sensibilisée b unités de temps de façon continue, elle doit alors être tirée, à moins qu'elle ne soit désensibilisée par un autre franchissement. C'est un modèle à sémantique de franchissement forte. Il est possible d'appliquer à ce modèle une sémantique faible. Un tel model est désigné par TT_WPN.

Définition 8.1.1 ***(Réseau de Petri TTPN)***
Un réseau de Petri TTPN est une paire (PN, I) *où* PN *est un réseau de Petri ordinaire et* I *une fonction* $I : T \rightarrow INT$ *qui associe, à chaque transition, un intervalle de temps* $[I_{min}(p, t), I_{max}(p, t)]$ *appelé intervalle statique de franchissement de* t.

8.1.2 Réseaux de Petri P-temporels

Les fondements théoriques des réseaux de Petri P-temporels ont été élaborés par Khansa dans sa thèse (Khansa, 1997). Le modèle de Khansa (TP_SPN) associe un intervalle de temps aux places (figure 8.1.b). Une place annotée d'un intervalle $[a, b]$ signifie que tout jeton créé dans cette place est disponible uniquement si son âge est bien dans l'intervalle $[a, b]$. Si un jeton n'est pas consommé durant les b unités de temps, il meurt (désormais, il n'est plus utilisable). Le modèle de Khansa a une sémantique forte. Il est possible d'appliquer à ce modèle une sémantique faible. Un tel model est désigné par TP_WPN.

Définition 8.1.2 ***(Réseau de Petri TPPN)***
Un réseau de Petri TPPN est une paire (PN, I) où PN est un réseau de Petri ordinaire et I une fonction $I : T \rightarrow INT$ qui associe, à chaque place, un intervalle de temps $[I_{min}(p, t), I_{max}(p, t)]$ appelé intervalle statique de temps de séjour d'un jeton dans la place p.

8.1.3 Réseaux de Petri A-temporels

Le modèle de Walter (TA_WPN) (Walter, 1983) associe un intervalle de temps à chaque arc reliant une place à une transition (figure 8.1.c). Un arc (p, t) annoté d'un intervalle $[a, b]$ signifie que tout jeton créé dans la place p est disponible pour la transition t, uniquement si son âge est bien dans l'intervalle $[a, b]$. Une transition ne peut être tirée que si tous les jetons qu'elle utilise lui sont disponibles. Le franchissement de la transition n'est pas forcé (sémantique faible) et les jetons peuvent mourir. Il est également possible d'appliquer à ce modèle une sémantique forte. Un tel modèle est désigné par TA_SPN.

Définition 8.1.3 ***(Réseau de Petri TAPN)***

Un réseau de Petri TAPN est une paire (PN, I) *où* PN *est un réseau de Petri ordinaire et* I *une fonction* $I : T \rightarrow INT$ *qui associe, à chaque arc entrant, un intervalle de temps* $[I_{min}(p,t), I_{max}(p,t)]$ *appelé intervalle statique de disponibilité*.

8.1.4 Comparaison des extensions temporelles

Une étude complète de l'expressivité de ces modèles en terme de bisimilarité temporelle (sémantique temporelle) a été proposée par (Boyer and Roux, 2007). Globalement, il s'avère que :

- les réseaux A-temporels et P-temporels sont comparables. Leur règle de franchissement consiste en une conjonction des intervalles statiques des jetons à consommer. Tandis que la règle de franchissement des modèles T-temporels ne tient compte que des jetons les plus récents.
- les réseaux A-temporels sont les plus expressifs et généralisent tous les modèles. Comme on peut le voir sur la figure 8.2, on peut facilement traduire un modèle P-temporel en un modèle A-temporel. Cependant, il est difficile (sans outils adaptés) de traduire un modèle T-temporel en un modèle A-temporel. À titre d'exemple, dans la thèse de doctorat de (Boyer, 2001), un simple modèle T-temporel composé de 3 places, 2 transitions et 4 arcs a nécessité l'ajout de 8 places, 19 transitions et 38 arcs.
- Les modèles A-temporels et P-temporels à sémantique forte généralisent ceux à sémantique faible.

La figure 8.3, tirée de (Boyer and Roux, 2007), montre des résultats d'expressivité[1] des différentes extensions. En sémantique forte, les réseaux T-temporels et P-temporels sont

[1] $\approx$ désigne une équivalence de bisimilarité.

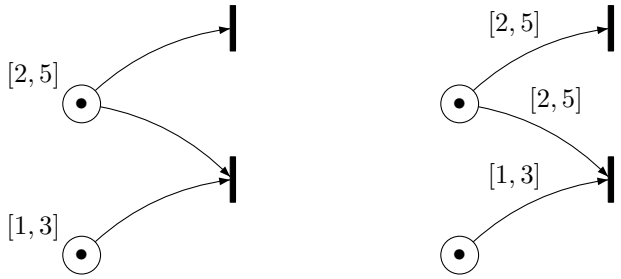


Figure 8.2 Traduction d'un modèle TPPN vers un modèle TAPN

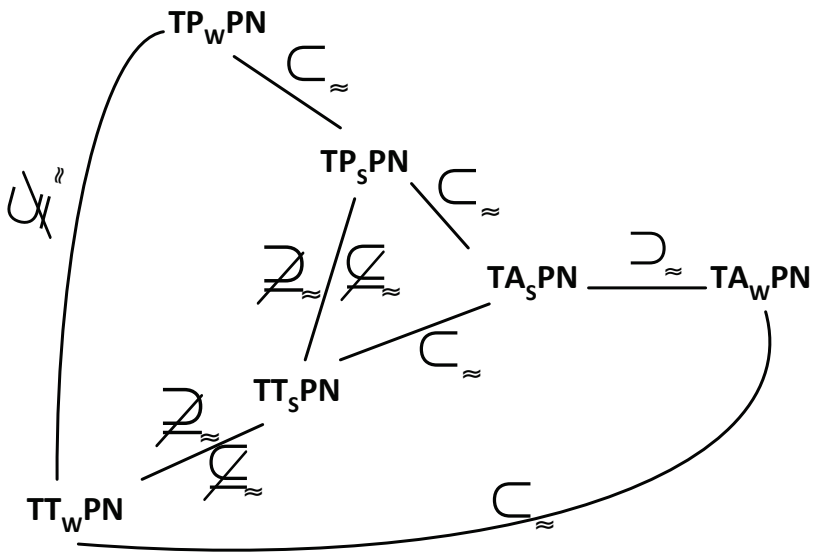
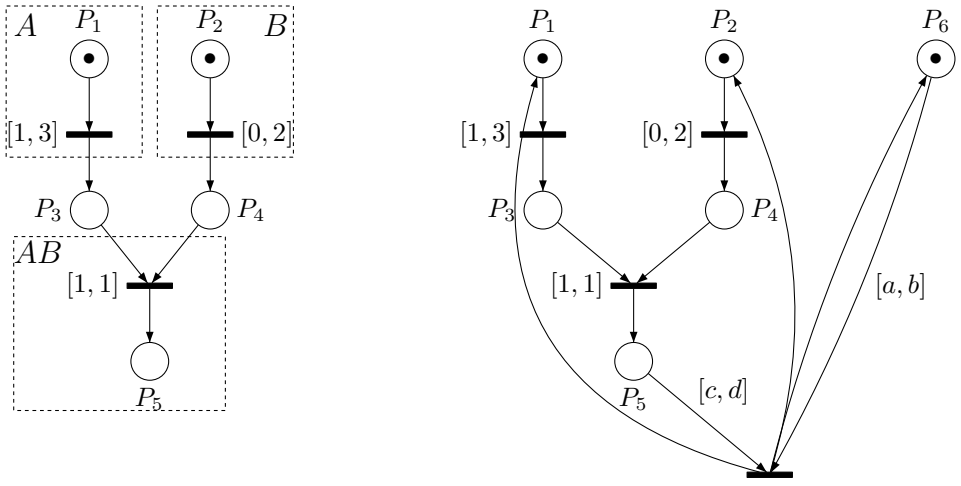


Figure 8.3 Relation entre les différentes extensions temporelles (Boyer and Roux, 2007)

incomparables, tandis que les réseaux A-temporels sont plus expressifs que les réseaux P-temporels et les réseaux T-temporels.

Bien qu'il soit possible de traduire un modèle vers un autre, cette traduction entraîne souvent des pertes en coût et en performance non négligeables, notamment par l'ajout d'autres places et transitions. Par ailleurs, chaque modèle a sa propre sémantique, c'est-à-dire qu'on associe le même type de contraintes à toutes les transitions. Or, il existe divers systèmes réels qui intègrent plus qu'un seul type de contraintes.



a) Représentation en modèle de Merlin b) Vérification de la périodicité de la production

Figure 8.4 Un système de production et d'assemblage de pièces d'autos

Par exemple, les réseaux de Petri T-temporels sont très utilisés pour modéliser et analyser des systèmes de production manufaturière, notamment pour déterminer le temps de production optimal. Cependant, ces systèmes sont sujets à plusieurs types de contraintes de temps qui ne peuvent pas être prises en compte par un seul modèle. Nous illustrons ce point par un simple système de production et d'assemblage de pièces d'autos représenté par le modèle de Merlin à la figure 8.4.a.

Le modèle représente un produit fini AB qui nécessite deux pièces à procurer de deux stocks de pièces A et B représentés respectivement par les places P_3 et P_4. Les matières premières des pièces A et B sont initialement présentes dans les places P_1 et P_2 respectivement. Nous avons trois contraintes temporelles relatives à la durée de fabrication des pièces A et B ainsi que la durée d'assemblage pour produire AB. L'assemblage ne peut s'effectuer que si les deux pièces sont présentes. Pour assurer cette synchronisation, nous utilisons le modèle de Merlin où les intervalles sont associés aux transitions.

Supposons que la pièce assemblée AB est disponible à l'intérieur d'un intervalle $[c, d]$. On veut vérifier si le processus de fabrication peut être répété périodiquement suivant une période $[a, b]$. Il s'agit donc de faire un rendez-vous pour prendre en compte les deux contraintes. Cependant, le rendez-vous ne peut pas être exprimé par le modèle de Merlin sachant qu'il n'y a pas nécessairement une intersection entre les deux intervalles. En effet, en utilisant le modèle de Merlin, on considère toujours les jetons les plus récents et donc nécessairement ce sera celui de la place P_5 (puisque P_6 est déjà marquée). Ainsi la contrainte de périodicité ne sera jamais prise en compte. En revanche, si on spécifiait les contraintes au niveau des arcs tel que représenté à la figure 8.4.b, les deux contraintes seront prises en compte. Or, il n'existe pas de modèle temporel qui permet d'exprimer à la fois différentes sémantiques (forte et faible). Cela dit, imposer une même sémantique à tous les composants d'un système pourrait s'avérer limitatif.

Ainsi, dans (Rakkay et al., 2007), nous proposons s'étendre les réseaux de Petri à arcs temporels en offrant la possibilité de spécifier pour chaque transition une sémantique de franchissement forte ou faible similaires. Nous montrons ensuite que les techniques de construction de graphes des zones peuvent être adaptées au modèle proposé.

8.2 Modèle $TA_{WS}PN$

Le modèle $TA_{WS}PN$ est un réseau de Petri coloré qui associe à chaque arc connectant une place à une transition, un intervalle de temps et à chaque transition l'une des quatre sémantiques de franchissement : All, $None$, $Last_S$, $Last_W$. Autrement dit, le modèle $TA_{WS}PN$ offre la possibilité de spécifier pour chaque transition du modèle une sémantique de franchissement différente, forte ou faible. Cela offre ainsi une grande flexibilité dans la modélisation de systèmes temporisés complexes sans complexifier les méthodes d'analyse classiques. L'intérêt de ce modèle est d'offrir également des possibilités de vérification comparables à celles des autres extensions temporelles des réseaux

de Petri. En effet, le modèle $TA_{WS}PN$ offre une technique de model-checking basée sur la construction de graphes des zones (Gardey et al., 2003).

8.2.1 Définition du modèle $TA_{WS}PN$

Définition 8.2.1 ***(Modèle*** $TA_{WS}PN$***)***
$TA_{WS}PN$ est un tuple $(P, T, Pre, Post, M_0, I, Strong)$ où :

- *P et T sont respectivement des ensembles finis de places et de transitions tel que: ($P \cap T = \emptyset$) ;*
- *Pre et $Post$: $P \times T \rightarrow N$ sont respectivement les fonctions d'incidence avant et arrière, $T \rightarrow P_{MS}$ où P_{MS} est l'ensemble de tous les multi-ensembles sur P;*
- *$M_0 \in P_{MS}$ est le marquage initial;*
- *$I : P \times T \rightarrow INT$ associe, à chaque arc entrant, un intervalle de temps $[I_{min}(p,t), I_{max}(p,t)]$ appelé intervalle statique des jetons de la place p pour la transition t. $INT = \{[y,z] \in Q^+ \times (Q^+ \cup \{\infty\}) \mid y \leq z\}$ dénote l'ensemble des intervalles de temps à bornes rationnelles.*
- *$Strong : T \rightarrow \{All, None, Last_S, Last_W\}$ associe une sémantique de franchissement à chaque transition.*

Définition 8.2.2 ***(Jeton temporisé)***
Un jeton temporisé est un couple (p, v) où p est sa place et v est la valeur de son horloge (son âge).

8.2.2 Évolution du modèle $TA_{WS}PN$

Pour qu'une transition soit tirée, il faut qu'elle soit sensibilisée et que les contraintes spécifiant son intervalle de franchissement soient vérifiées. Selon le type de synchronisation, ces contraintes peuvent porter sur l'ensemble ou uniquement les plus récents jetons participant à la sensibilisation de la transition. Notons que la durée de sensibilisation d'une transition sensibilisée est égale à l'âge des plus récents jetons participant à sa sensibilisation.

La sémantique de franchissement *All* correspond à une sémantique forte des réseaux de Petri P-temporels et A-temporels (TP_SPN et TA_SPN). La sémantique *None* correspond à une sémantique faible des réseaux de Petri TP_WPN et TA_WPN. Dans le cas d'une transition *All* ou *None*, la transition est franchissable si elle est sensibilisée et les horloges de tous ses jetons sont dans leurs intervalles statiques. Dans le cas d'une transition *All*, la transition doit être franchie avant que ces horloges ne dépassent leurs intervalles statiques, sauf si elle est désensibilisée par le tir d'une autre transition. Par contre, dans le cas d'une transition *None*, le franchissement n'est pas forcé. Les jetons des places en amont d'une transition *All* ou *None* ne peuvent être utilisés par elle en dehors de leurs intervalles statiques. Si un jeton n'est pas utilisé alors qu'il a excédé son intervalle statique, il devient jeton mort "*dead token*" pour cette transition.

Les sémantiques $Last_S$ et $Last_W$ étendent la sémantique du modèle de Merlin aux réseaux de Petri à arcs temporels avec la possibilité d'utiliser une sémantique faible. Une transition de type $Last_S$ ou $Last_W$ est franchissable si elle est sensibilisée et les horloges de ses plus récents jetons [2] sont dans leurs intervalles statiques. Autrement dit, c'est le dernier jeton qui valide une transition qui va fixer son échéance de franchissement. Une transition de type $Last_S$ doit être franchie avant que ces horloges ne

[2] Les plus récents jetons parmi ceux utilisés par la transition.

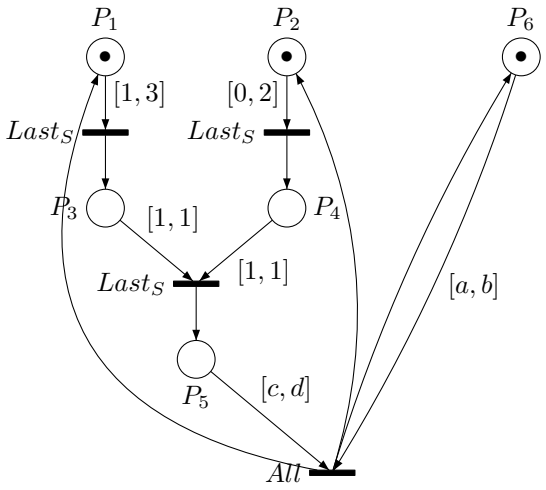


Figure 8.5 Le modèle $TA_{WS}PN$ du système de la figure 8.4

dépassent leurs intervalles statiques, sauf si elle est désensibilisée par le tir d'une autre transition. Le franchissement de la transition n'est pas forcé dans le cas de $Last_W$. Notons qu'un jeton peut être utilisé par une transition de type $Last_W$ ou $Last_S$ même si son horloge dépasse son intervalle statique. Ainsi, les jetons ne meurent jamais pour les transitions de type $Last_S$ ou $Last_W$.

Ainsi, nous pouvons représenter l'exemple de la figure 8.4 par le modèle $TA_{WS}PN$ tel que montré à la figure 8.5.

Pour caractériser l'état du modèle, on associe à chaque jeton une horloge représentant son âge. L'horloge est initialisée à 0 à sa création. Elle croît ensuite avec le temps, de manière synchrone, jusqu'à ce que le jeton soit consommé. Nous donnons ci-après, quelques définitions utiles.

Définition 8.2.3 ***(État du modèle / Événements d'un état)***

- *Un état q du modèle $TA_{WS}PN$ est un multi-ensemble sur l'ensemble des jetons*

temporisés, i.e.: $q \in (P \times \mathrm{R}_{\geq 0})_{MS}$. *L'état initial est :* $q_0 = \sum_{p \in P} M_0(p) \bullet (p, 0)$.

- *Soient* q *un état et* $Mark(q)$ *le marquage obtenu à partir de* q *en éliminant tous les paramètres temporels, i.e.:* $Mark(q) = \sum_{(p,v) \in q} q(p, v) \bullet p$.

  - *Soit* t *une transition de* T. *La transition* t *est sensibilisée pour* q *si et seulement si, tous les jetons nécessaires à son franchissement sont présents dans* q, *i.e.:* ${}^\bullet t \leq Mark(q)$.

  - *Un événement de* q *est un triplet* $e = (t, in, out)$ *où* t *est une transition sensibilisée par* q, in *représente les jetons à consommer et* out *les jetons à produire par le franchissement de* t, *i.e.* $in \leq q \wedge Mark(in) = {}^\bullet t$ *et* $out = \sum_{p \in P} t^\bullet(p) \bullet (p, 0)$.

  - *On note* $E(q)$ *l'ensemble des événements de* q, *i.e. :* $\{(t, in, out) | t \in T \wedge in \leq q \wedge Mark(in) = {}^\bullet t \wedge out = \sum_{p \in P} t^\bullet(p) \bullet (p, 0)\}$.

  - *Pour tout événement* $e = (t, in, out)$, *nous désignons respectivement par* ${}^\bullet e$ *et* $e^\bullet$ *les multi-ensembles de jetons temporisés consommés et produits suite au franchissement de* e *(i.e. :* ${}^\bullet e = in$ *et* $e^\bullet = out$*).*

Partant de son état initial, le modèle évolue par progression de temps (les horloges évoluent avec le temps) ou par franchissement d'événements. Dans ce qui suit, nous considérons un état q du modèle $TA_{WS}PN$, un événement $e = (t, in, out)$ de q et un réel positif dv.

Définition 8.2.4 ***(Intervalle de franchissement d'un événement)***

- *Soit* $W(e)$ *l'ensemble des jetons à considérer pour déterminer l'intervalle de franchissement de* e, *i.e. :*

$$W(e) = \begin{cases} {}^\bullet e & \text{si } Strong(t) \notin \{Last_S, Last_W\}; \\ \{(p,v) \in {}^\bullet e \mid \forall (p',v') \in {}^\bullet e, (p',v') \preceq (p,v)\} & \text{sinon} \end{cases}$$

Où la relation $(p', v') \preceq (p, v)$ *signifie que le jeton* (p', v') *a été créé avant ou au même temps que le jeton* (p, v)[3].

- *L'intervalle de franchissement de l'événement* e*, noté* $FD(e)$ *est l'intersection des intervalles de disponibilité de tous les jetons de* $W(e)$*. Les bornes de cet intervalle, notées* $FDmin(e)$ *et* $FDmax(e)$ *sont calculées comme suit :*
  - $FD_{min}(e) = max(0, Max_{(p,v)\in W(e)}(Imin(p,t) - v))$ *et*
  - $FD_{max}(e) = Min_{(p,v)\in W(e)}(Imax(p,t) - v)$
- *Si* $FD(e)$ *est vide, l'événement* e *est considéré invalide et ne peut plus être franchi.*

Dans ce modèle, dans le cas où $Strong(t) = Last_S$ ou All, le franchissement de tout événement $e = (t, in, out)$ doit être forcé lorsque son délai de franchissement maximal atteint 0. Par contre, si $Strong(t) = None$ ou $Last_W$, le franchissement de l'événement n'est pas forcé. Dans ce cas, l'événement devient non franchissable lorsque son délai de franchissement maximal atteint des valeurs négatives. La progression de temps à partir de chaque état est ainsi contrainte par les franchissements qui doivent être forcés.
Soient q un état du modèle et $E_F(q)$ l'ensemble des événements qui doivent être forcés lorsque leurs délais de franchissement maximaux atteignent 0, i.e. :
$E_F(q) = \{e = (t, in, out) \in E(q) \mid \ Strong(t) \in \{All, Last_S\}\}$.

Définition 8.2.5 ***Franchissement d'un événement et progression de temps***
Soit e_f *un événement de l'état de* q*.*

[3] *Notons que deux jetons peuvent avoir le même âge sans qu'ils soient créés par le même franchissement.*

- *l'événement e_f est franchissable à l'état q si son délai de franchissement minimal a atteint 0 i.e. : $FD_{min}(e_f) = 0 \wedge FD_{max}(e_f) \geq 0$.*

- *Si l'événement e_f peut se produire à l'état q, son occurrence est instantanée et mène vers l'état : $q' = q -^{\bullet} e_f + e_f^{\bullet}$.*

- *Une progression de temps de dv unités est réalisable, à partir de l'état q, sans qu'il n'y ait aucun franchissement, si et seulement si la valeur dv est inférieure ou égale au délai de franchissement maximal de tout événement de $E_F(q)$, i.e. : $\forall e \in E_F(q), dv \leq FD_{max}(e)$.*

- *Après cette progression de temps, l'horloge de chacun des jetons (p, v) est augmentée de dv unités de temps. L'état atteint suite à cette progression du temps est désigné par $q + dv$, i.e. : $q + dv = \sum_{(p,v)\in q} q(p, v) \bullet (p, v + dv)$.*

Définition 8.2.6 ***(Sémantique du modèle $TA_{WS}PN$)***
La sémantique du modèle $TA_{WS}PN$ est définie par le système de transitions $(Q, q_0, \rightarrow)$:

- $Q = (P \times \mathbb{R}_{\geq 0})_{MS}$,

- $q_0 = \sum_{p\in P} M_0(p) \bullet (p, 0)$,

- $\rightarrow \subseteq (Q \times (E \cup \mathbb{R}_{\geq 0}) \times Q)$ *est la relation de transition définie par:* $\forall q, e_f, dv, q'$

  - $q \xrightarrow{dv} q'$ *ssi* $\begin{cases} \forall\, e \in E_F(q),\ dv \leq FD_{max}(e) & \text{et} \\ q' = \sum_{(p,v)\in q} q(p, v) \bullet (p, v + dv) \end{cases}$

  - $q \xrightarrow{e_f} q'$ *ssi* $\begin{cases} e_f \in E(q); \\ FD_{min}(e_f) = 0 \wedge FD_{max}(e_f) \geq 0 & \text{et} \\ q' = q -^{\bullet} e_f + e_f^{\bullet} \end{cases}$

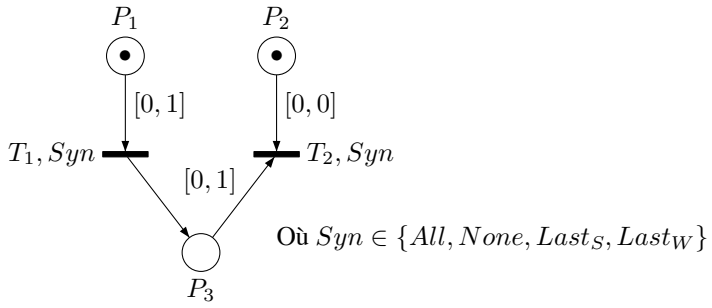


Figure 8.6 Un $TA_{WS}PN$ ayant un nombre borné de zones

8.2.3 Exemple

Considérons les modèles de la figure 8.6. Ces modèles ont le même état initial: $q_0 = (P_1, 0) + (P_2, 0)$. Cet état a un seul événement $e_1 = (t_1, (P_1, 0), (P_3, 0))$. Sachant que l'événement e_1 utilise un seul jeton, son intervalle de franchissement est le même pour les 4 types de synchronisation $([0, 1])$. Par contre, le franchissement de l'événement est forcé après une progression d'une unité de temps pour $Syn \in \{All, Last_S\}$. Alors qu'il ne l'est pas pour les autres cas.

Supposons que tous ces modèles subissent une progression d'une unité de temps puis franchissent l'événement e_1. Ces modèles atteindront alors le même état $q_1 = (P_2, 1) + (P_3, 0)$. Cet état a un seul événement $e_2 = (t_2, (P_2, 1) + (P_3, 0), 0)$. À l'état q_1, le jeton $(P_2, 1)$ est mort pour $Syn \in \{All, None\}$ mais il ne l'est pas pour les autres cas. Pour $Syn \in \{All, None\}$, l'événement e_2 n'est donc pas franchissable à l'état q_1 puisque $W(e_2) = (P_2, 1) + (P_3, 0)$, $FD_{min}(e_2) = 0$ et $FD_{max}(e_2) = -1$. En revanche, pour $Syn \in \{Last_S, Last_W\}$, l'événement e_2 est franchissable car
$W(e_2) = (P_3, 0)$, $FD_{min}(e_2) = 0$ et $FD_{max}(e_2) = 1$.

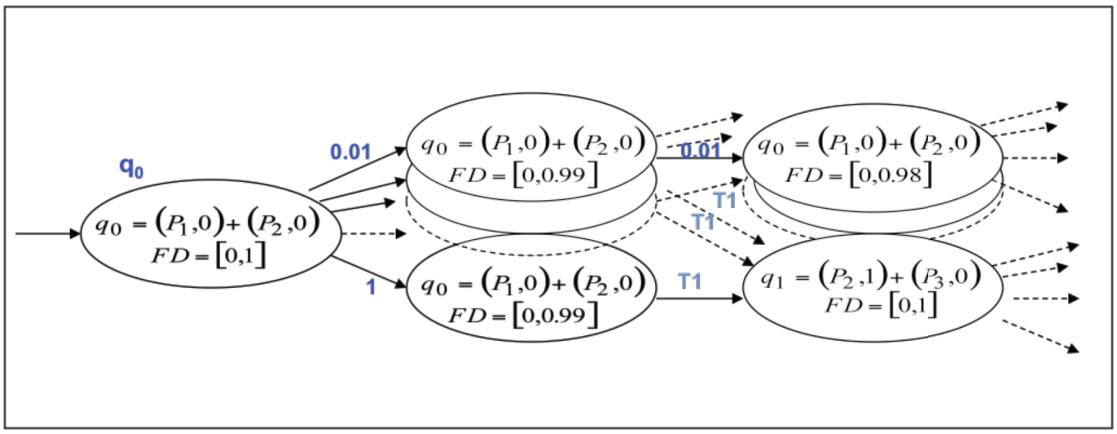


Figure 8.7 Explosion de l'espace des états

8.3 Abstraction du temps

À partir de la sémantique du modèle, on peut voir que, suite à la progression du temps, certains jetons peuvent mourir. Si ces derniers sont éliminés, le marquage va aussi changer par progression du temps. Par conséquent, un graphe représentant l'ensemble de toutes ces possibilités serait très large (Figure 8.7). Pour atténuer cette explosion combinatoire, nous proposons de regrouper, en un seul chemin, toutes les évolutions composées de la même séquence de franchissement (traces), et en un seul nœud, appelé, zone d'états, tous les états accessibles par les évolutions regroupées. Les états regroupés ensemble n'ont pas nécessairement les mêmes jetons vivants puisque un même jeton peut être mort dans un état de la zone et vivant dans un autre état de la même zone. Toutefois, on peut les représenter par un multi-ensemble de jetons où les valeurs des horloges sont remplacées par leurs noms (marquage symbolique) et une formule logique caractérisant les valuations des horloges de tous les états regroupés dans la zone d'états. Chaque valuation de l'horloge correspond à un état. La valeur de l'horloge de chaque jeton va déterminer s'il s'agit d'un jeton mort ou vivant.

8.3.1 Caractérisation des zones d'états

Définition 8.3.1 ***(Zone d'états)***
Une zone d'états α est définie par un marquage symbolique et une formule logique, i.e., (SM, FT) où :

- *SM (marquage symbolique) est un multi-ensemble de jetons temporisés où les horloges sont nommées (au lieu de mettre leurs valeurs).*
- *FT est une formule logique caractérisant les valuations des horloges de tous les états regroupés dans la zone α. Chaque valuation de l'horloge correspond à un état de la zone α.*

La zone d'états initiale est composé uniquement de l'état initiale où toutes les horloges sont nulles. Cette zone est notée $\alpha_0 = (SM_0, FT_0)$ où SM_0 est obtenu à partir de l'état initial en remplaçant les valeurs des horloges par les noms des horloges et $FT_0 = \bigwedge_{(p,h)\in SM_0}(h = 0)$.

FD_{min}, FD_{max}, ${}^\bullet e$, $e^\bullet$, $W(e)$, E et E_F sont définis de la même manière que dans les sections précédentes, à part que dans ce nouveau contexte, les horloges sont nommées au lieu de mettre leurs valeurs numériques.

Exemple
Pour illustrer ceci, considérons les modèles de la figure 8.6. La zone initiale de ces modèles est (SM_0, FT_0) où $SM_0 = (P_1, h_1) + (P_2, h_2)$ et $FT_0 = (h_1 = 0 \wedge h_2 = 0)$. Cette zone a un seul événement $e_1 = (t_1, (P_1, h_1), (P_3, h_3))$. Nous avons donc : ${}^\bullet e_1 = (P_1, h_1), e_1^\bullet = (P_3, h_3), W(e_1) = (P_1, h_1), FD_{min}(e_1) = max(0, 0-h_1), FD_{max}(e_1) = 1 - h_1, E(SM) = \{e_1\}$. Par contre, $E_F(SM) = \{e_1\}$ pour $Syn \in \{All, Last_s\}$, alors que $E_F(SM) = \emptyset$ pour les autres cas.

8.3.2 Calcul des zones d'états

Le calcul des zones d'états est basé sur l'extension des opérations de progression de temps et de franchissement aux zones. L'opération $\overrightarrow{\alpha}$, où α est une zone, est obtenue à partir de la zone α en y ajoutant tous les états accessibles par progression de temps, i.e. $\overrightarrow{\alpha} = \{q' \mid \exists q \in \alpha, \exists dv \in \mathbb{R}_{\geq 0}, q \xrightarrow{dv} q'\}$. L'opération $Next(\alpha, e)$ retourne la zone successeur de la zone α par l'évènement e, i.e.: $Next(\alpha, e) = \{q' \mid \exists q \in \alpha, q \xrightarrow{e} q'\}$.

À partir de la zone initiale, les zones d'états successeurs sont calculées de manière itérative, d'abord en ajoutant dans une zone, les états obtenus par progression de temps (Figure 8.8), puis en calculant ses successeurs par franchissement d'événements (Figure 8.9).

Figure 8.8 Calcul des états accessibles par écoulement du temps

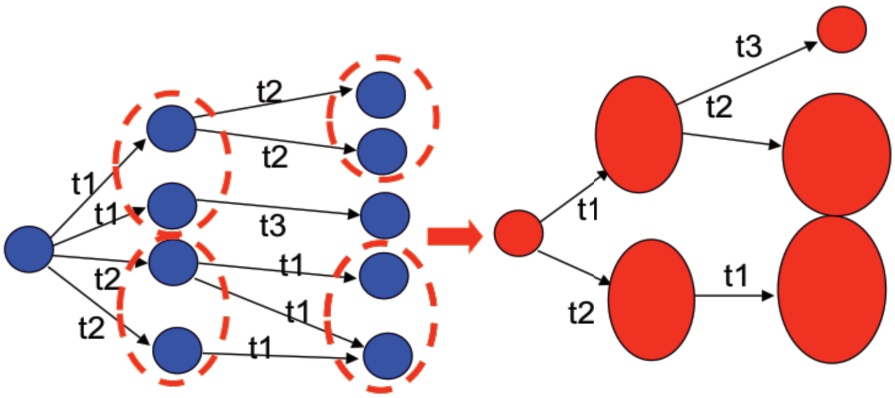


Figure 8.9 Calcul de la zone successeur par un événement

Le calcul des zones d'états accessibles nécessite de tester la consistance (le test du vide), l'équivalence des formules (test d'égalité) et d'autres opérations telles que l'ajout et l'élimination de contraintes atomiques de la forme: $x - y \leq c, x \leq c$ ou $-x \leq c$, où

x et y sont des horloges et $c \in (Q \cup \{-\infty, \infty\})$. Ces opérations sont simplifiées par l'utilisation des matrices de bornes sous formes canoniques (Behrmann et al., 2002). Les formules logiques des zones d'états sont aussi des conjonctions de contraintes atomiques et peuvent donc être représentées par des $DBMs$.

Soient $\alpha = (SM, FT)$ une zone d'états et V l'ensemble des horloges de SM. La forme canonique de FT est : $\bigwedge_{x,y \in V \cup \{o\}} x - y \leq Max_{FT}(x - y)$, où $Max_{FT}(x - y)$ est la valeur maximale de $x - y$ dans le domaine de FT, et o un symbole représentant la valeur 0. Cette forme canonique est représentée par la matrice de bornes H d'ordre $|V \cup \{o\}|$ définie par : $\forall (x, y),\ H_{xy} = Max_{FT}(x - y)$.

Proposition 8.3.1 *Calcul de* $\overrightarrow{\alpha}$
Soit $\alpha = (SM, FT)$ *une zone d'états. La zone d'états* $\overrightarrow{\alpha} = \{q' \mid \exists q \in \alpha, \exists dv \in \mathbb{R}_{\geq 0}, q \xrightarrow{dv} q'\}$ *est caractérisée par la formule* $\overrightarrow{FT}$ *qui peut être calculée de la façon suivante :*

1 *Initialiser* $\overrightarrow{FT}$ *à la forme canonique de* FT*;*

2 *Éliminer toutes les contraintes atomiques de la forme* $h - o \leq c, h \in V$*;*

3 *Pour tout* $e = (t, in, out) \in E_F(SM')$ *et* $(p, h) \in W(e)$*, ajouter la contrainte* $h \leq Imax(p, t)$*;*

4 *Calculer la forme canonique de la formule ainsi obtenue.*

Preuve 8.3.1 *Après la deuxième étape, les bornes supérieures de toutes les horloges sont remplacées par* ∞*. Par conséquent, tous les états accessibles par progression de temps sont, sans aucune restriction, ajoutés. La troisième étape va restreindre la progression de sorte qu'elle ne dépasse pas les délais de franchissement maximaux de tous*

les événements qui seront forcés. À la quatrième étape, la formule obtenue est mise sous forme canonique.

Exemple

L'application de cette opération à la zone initiale des modèles 8.6 produit la zone $\overrightarrow{\alpha_0} = (SM_0, \overrightarrow{FT_0})$, où $\overrightarrow{FT_0}$ est la forme canonique de :

$$\begin{cases} (0 \leq h_1 \leq 1 \wedge h_2 = h_1) & \text{pour Syn} \in \{\text{All}, \text{Last}_\text{S}\}, \\ (0 \leq h_1 \leq \infty \wedge h_2 = h_1) & \text{pour Syn} \in \{\text{None}, \text{Last}_\text{W}\}. \end{cases}$$

Proposition 8.3.2 *Calcul de $Next(\alpha, e_f)$*

Soient $\alpha = (SM, H)$ une zone d'états et $e_f = (t_f, in_f, out_f)$ un événement de cette zone. La zone d'états $Next(\alpha, e) = \{q' \mid \exists q \in \alpha, q \xrightarrow{e} q'\}$ peut être calculée de la façon suivante :

- *L'événement e_f est franchissable à partir de α (i.e.:$Next(\alpha, e_f) \neq \emptyset$) ssi la formule obtenue en ajoutant, pour chaque jeton $(p, h) \in W(e_f)$, la contrainte $Imin(p, t_f) \leq h \leq Imax(p, t_f)$ est consistante.*
- *Si l'événement e_f peut se produire à partir de la zone d'états α, son occurrence est instantanée et mène vers la zone d'états $Next(\alpha, e_f) = (SM', FT')$, où $SM' = SM - {}^\bullet e_f + e_f^\bullet$ et FT' est calculée de la façon suivante :*
  1. *Initialiser FT' à $FT \wedge \bigwedge_{(p,h) \in {}^\bullet e_f} Imin(p, t_f) \leq h \leq Imax(p, t_f)$.*
  2. *Mettre FT' sous forme canonique;*
  3. *Eliminer les horloges de tous les jetons consommés par l'événement e_f (i.e.: tous les jetons de ${}^\bullet e_f$);*
  4. *Ajouter la contrainte $h = 0$ pour chaque jeton (p, h) créé par e_f $(i.e. : (p, h) \in e_f^\bullet)$ puis mettre la formule obtenue sous forme canonique.*

Preuve 8.3.2 *L'événement e_f est franchissable à partir de $\alpha = (SM, FT)$ ssi il existe au moins un état dans α à partir duquel e_f est franchissable. Selon la définition 8.2.5, on a $Next(\alpha, e_f) \neq \emptyset$ ssi la formule obtenue à partir de FT, par l'ajout des contraintes $FD_{min}(e_f) = 0 \wedge 0 \leq FD_{max}(e_f)$, est consistante, i.e. :*
(i) $FT \wedge \bigwedge_{(p,h)\in {}^\bullet e_f}(max(0, Imin(p, t_f) - h) = 0 \wedge 0 \leq Imax(p, t_f) - h)$ est consistante.
Si l'événement e_f est franchissable à partir de α, son franchissement est instantané et mène vers une nouvelle zone $Next(\alpha, e_f) = (SM', FT')$, où
$SM' = SM -^\bullet e_f + e_f^\bullet$ et FT' est obtenue à partir de la forme canonique de (i) après y avoir éliminé les horloges des jetons consommés et y avoir ajouté la contrainte $h = 0$ pour tout jeton créé (p, h).

Exemple

Considérons les zones d'états $\overrightarrow{\alpha_0}$ calculées précédemment pour les modèles de la figure 8.6. Toutes ces zones d'états ont un seul événement $e_1 = (t_1, (P_1, h_1), (P_3, h_3))$. Son franchissement mène vers la même zone d'états $\alpha_1 = (SM_1, FT_1)$ où $SM_1 = (P_2, h_2) + (P_3, h_3)$ et FT_1 est la forme canonique de $(0 \leq h_2 \leq 1 \wedge h_3 = 0)$.

8.3.3 Graphe des zones d'états

Le graphe des zones du modèle $TA_{WS}PN$ est une structure $(\mathbb{Z}, \longmapsto, \overrightarrow{\alpha_0})$, où:

- α_0 est la zone d'états initiale,
- $\longmapsto$ est la relation de transition définie par:

$$\forall \alpha, \alpha', e_f,\ \alpha \xmapsto{e_f} \alpha' \text{ ssi } \begin{cases} e_f \in E(SM); \\ Next(\alpha, e_f) \neq \emptyset; \\ \alpha' = \overrightarrow{Next(\alpha, e_f)} \end{cases}$$

- $\mathbb{Z} = \{\alpha | \alpha_0 \xmapsto{\star} \alpha\}$, où $\xmapsto{\star}$ est la fermeture réflexive et transitive de $\longmapsto$.

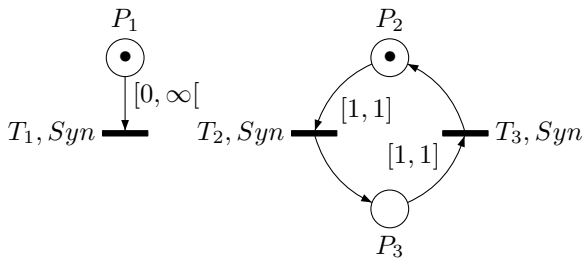


Figure 8.10 Un $TA_{WS}PN$ ayant un nombre borné de zones

Cependant, le graphe des zones d'états résultant peut être infini même pour un modèle borné. En effet, nous pouvons avoir une infinité de zones accessibles dans le cas où certains jetons ne sont jamais consommés (leurs horloges croissent indéfiniment). C'est le cas du modèle de la figure 8.10. Le jeton de la place P_1 n'est jamais consommé si le modèle exécute, à répétition, la séquence de transition $T_2; T_3$. Afin de remédier à cette limitation, nous proposons d'appliquer une opération d'approximation, similaire à celle utilisée pour les automates temporisés afin d'assurer la convergence du calcul du graphe d'accessibilité. Chaque nouvelle zone calculée est d'abord approximée, avant de la comparer avec celles déjà calculées.

8.3.4 Approximation des zones d'états

L'approximation d'une zone d'états consiste à étendre le domaine des horloges en ajoutant des valeurs qui ont la même influence sur le comportement du modèle que celles qui sont déjà dans le domaine. Pour l'appliquer, il faut déterminer pour chaque horloge la valeur qu'il faudrait atteindre avant d'étendre le domaine de l'horloge (valeur limite). Dans le cas des automates temporisés, la limite d'une horloge est la plus grande valeur finie qui est comparée, dans le modèle, à cette horloge. Dans notre cas, l'horloge d'un jeton (p, h) est comparée aux intervalles statiques des arcs en sortie de la place p. Ainsi, les horloges de tous les jetons de la même place partagent la même limite définie par :

$lim_p = max(Max_{t \in p^\bullet \wedge Imax(p,t) < \infty} Imax(p,t),\ Max_{t \in p^\bullet \wedge Imax(p,t) = \infty} Imin(p,t))$.
Par exemple, pour les modèles de la figure 8.6, les limites des places sont : $lim_{P_1} = 1, lim_{P_2} = 0, lim_{P_3} = 0$.

Notons qu'un jeton qui a dépassé sa limite n'est pas forcément mort. Un jeton (p,h) est mort pour une transition t de $p^\bullet$ si $Imax(p,t) < h$ et $Strong(t) \in \{All, None\}$. Il est définitivement mort s'il est mort pour toutes les transitions en sortie de p.

Soient (SM, FT) une zone d'états sous forme canonique et H la DBM de FT. L'approximation de H est la forme canonique de la DBM H' définie par :

$$\forall (x,y) \in (V \cup \{o\})^2, H'_{xy} = \begin{cases} \infty & \text{si } H_{xy} > k_x; \\ -k_y & \text{si } H_{xy} < -k_y; \\ H_{xy} & \text{sinon.} \end{cases}$$

Où $k_x = lim_p$, si p est la place du jeton associé à l'horloge x ($x \neq o$), $k_o = \infty$ sinon.

Théorème 8.3.1 *(i) Le graphe des zones d'états approximées préserve les marquages et les traces du modèle.*

(ii) Le graphe des zones d'états approximées est fini ssi $TA_{WS}PN$ est borné.

Preuve 8.3.3 *(i) La preuve de ce théorème repose sur les résultats établis par (Bouyer, 2002) pour les automates temporisés à contraintes non triangulaires*[4]*. Pour ces automates, il a été prouvé que l'approximation préserve les traces du modèle. Dans notre modèle, toutes les contraintes du modèle sont non triangulaires*[5]*. De plus, la borne k_x de chaque horloge x est en fait la plus grande constante finie qui est comparée à l'horloge x.*

[4] *Une contrainte triangulaire est de la forme $x - y \leq c$ ou $x - y < c$, où x et y sont des horloges autres que l'horloge zéro.*

[5] *Les horloges des jetons sont comparées aux bornes des intervalles associés aux arcs.*

Tableau 8.1 Zones d'états accessibles (approximées) des modèles des figures 8.6 et 8.10

	Fig 8.6, $Syn \in \{All, Last_S\}$	Fig 8.6, $Syn \in \{None, Last_W\}$
α_0	$(P_1, h_1) + (P_2, h_2)$ $0 \leq h_1 = h_2 \leq 1$	$(P_1, h_1) + (P_2, h_2)$ $0 \leq h_1 = h_2$
α_1	$(P_2, h_2) + (P_3, h_3)$ $0 \leq h_3 \leq h_2 \wedge h_3 \leq 1$	$(P_2, h_2) + (P_3, h_3)$ $0 \leq h_3 \leq h_2$
α_2	0 true	0 true
	Fig 8.10, $Syn \in \{All, Last_S\}$	Fig 8.10., $Syn \in \{None, Last_W\}$
α_0	$(P_1, h_1) + (P_2, h_2)$ $0 \leq h_1 \wedge 0 \leq h_2 \leq 1$	$(P_1, h_1) + (P_2, h_2)$ $0 \leq h_1, h_2 \leq \infty$
α_1	(P_2, h_2) $0 \leq h_2 \leq 1$	(P_2, h_2) $0 \leq h_2 \leq 1$
α_2	$(P_1, h_1) + (P_3, h_3)$ $0 \leq h_1 \wedge 0 \leq h_3 \leq 1$	$(P_1, h_1) + (P_3, h_3)$ $0 \leq h_1 \wedge 0 \leq h_3$
α_3	(P_3, h_4) $0 \leq h_4 \leq 1$	(P_3, h_4) $0 \leq h_4$

(ii) $\Rightarrow$*) est trivial.* $\Leftarrow$*) Si le modèle est borné, il a un nombre fini de marquages accessibles. Pour chaque marquage, on a également un nombre fini de matrices de bornes approximées sous forme canonique (Bouyer, 2002).*

8.3.5 Exemple

Le modèle $TA_{WS}PN$ a fait l'objet d'une implémentation. Le programme prend en entrée un modèle $TA_{WS}PN$ et génère le graphe des zones (ZBG) ou le graphe des classes (SCG) avec ou sans approximation. Nous résumons dans le tableau 8.1, les zones d'états accessibles obtenues pour les modèles des figures 8.6 et 8.10.

Notons qu'afin de réduire la complexité de vérification, le nombre d'horloges peut être réduit en regroupant toutes les horloges égales. La plus grande limite des horloges du groupe sera, dans ce cas, considérée comme étant la limite du groupe. Il est également

possible d'ignorer les horloges des jetons qui ne sont ni parmi les plus récents, ni utilisables par des transitions de type All ou $None$.

8.4 Modélisation des Workflows à l'aide du modèle $TA_{WS}PN$

Nous proposons une approche de modélisation pour les systèmes Workflows basée sur le modèle $TA_{WS}PN$ qui est à la fois un réseau de Petri coloré et temporel.

Tout d'abord, à partir des travaux que nous avons présentés dans la section 4 du chapitre 2, on peut voir que les réseaux de pétri colorés répondent bien aux critères liées au pouvoir d'expression. Avec ces réseaux, les différentes instances d'un système Workflow peuvent être représentées sur le même modèle au moyen de différentes couleurs dont chacune décrit l'exécution d'une instance particulière du processus. Les réseaux colorés offrent des techniques de décomposition hiérarchique, des outils d'analyse et permettent l'expression et la preuve de propriétés sur les spécifications. De plus, en utilisant des réseaux colorés, il est possible de représenter dans un même modèle, le contrôle de flux, de données et d'autorisation par différentes couleurs.

Par ailleurs, les systèmes Workflows sont des systèmes hétérogènes caractérisées par différentes contraintes temporelles tel que mentionné par (Yu et al., 2004). Il y a des contraintes associées aux sujets pour exprimer le moment et la durée d'assignation d'un rôle à un usager, des contraintes associées aux données pour spécifier leur disponibilité et validité, des contraintes sur la durée d'exécution d'une tâche et éventuellement du processus en entier, etc. Par conséquent, nous pensons qu'il serait approprié d'utiliser le modèle $TA_{WS}PN$ puisqu'il intègre différentes sémantiques et offre une approche d'analyse qui présenterait un bon outil de vérification pour de nombreuses propriétés des systèmes workflows.

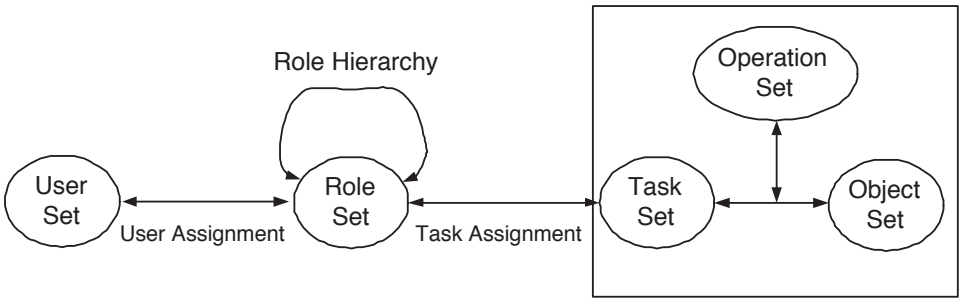


Figure 8.11 Un modèle RBAC pour les Workflows

D'autre part, nous proposons d'intégrer le contrôle d'accès basé sur les rôles (RBAC) dans la modélisation des Workflows. Plus exactement, l'idée est d'utiliser les réseaux de Petri colorés tel que nous l'avons présenté dans le chapitre 6, et de greffer le modèle résultant à celui du Workflow. Une approche globale, basée sur les réseaux de Petri, permettrait de tracer à la fois le flux des données, de contrôle et d'autorisation entre les différents intervenants du sytème. La figure 8.11, tiré de (Liu and Sun, 2004) illustre la relation que nous voulons exprimer entre le modèle RBAC et le workflow.

L'idée est d'accorder les autorisations, non pas via des droits de bas niveau, mais plutôt via des abstractions de niveau plus élevé correspondant à des opérations et des tâches de l'application. Ainsi, grâce à la notion de rôle, on accorde à un utilisateur l'ensemble des ressources nécessaires pour effectuer une tâche particulière. Le concept est déjà mis en application dans plusieurs systèmes, par exemple, le serveur *Planning Server* de Microsoft (une application d'entreprise pour la prise en charge les processus de gestion) utilise un modèle de sécurité basé sur les rôles pour l'autorisation. Il utilise les rôles pour protéger les métadonnées, les données et l'écriture différée, et pour activer les actions de workflow.

Nous reprenons donc la même idée puisque le modèle RBAC se prête naturellement à la protection des Workflows. Nous proposons toutefois, une approche modulaire qui s'effectue en trois étapes :

1- Modélisation du Workflow : représenter le système Workflow au moyen du modèle $TA_{WS}PN$ pour intégrer les différentes contraintes temporelles. Pour cela, nous adoptons la démarche de (Gou et al., 2001) où les différents modules sont composés ensemble. L'assignation ressource-activité se fait aussi selon les rôles des ressources. Cependant, cet assignation est contrôlé par le modèle RBAC présenté au chapitre 6, ce qui constitue la prochaine étape de modélisation.

2- Modélisation de la politique RBAC : exprimer les contraintes de sécurité relatives au Workflow via le modèle RBAC présenté au chapitre 6. Ce module va contrôler l'exécution des tâches du Workflow.

3- Connection des deux modèles : connecter les différents modules de sorte qu'une tâche ne peut être exécutée que si les ressources requises sont disponibles (le sujet est assigné le(s) rôle(s) requis et ces rôles sont actifs). Ainsi, une transition (une tâche) est franchie seulement si elle est exécutée par un sujet autorisé à accéder les objets requis à son exécution.

8.5 Conclusion

Pour conclure cette partie, nous remarquons que l'approche que nous proposons peut être généralisée à la modélisation des Workflows Inter-Organisationnels (WIO). L'objectif du WIO est de coordonner des processus workflow répartis, autonomes et hétérogènes, issus et s'exécutant dans différentes organisations. Dans un WIO, chaque organisation peut avoir sa propre politique de sécurité en fonction de ses exigences en termes de contrôle d'accès. Une difficulté majeure est de réussir à déployer des mécanismes de contrôle d'accès pour un WIO, sans pour autant affecter les politiques de sécurité de chacune des organisations participantes.

Dans la littérature, nous avons trouvé deux propositions intéressantes qui implémente des

mécanismes sécurité dans un système WIO par le biais des réseaux de Petri. (Sachin, 2005) et (Mikolajczak and Gami, 2008) proposent une approche à base des réseaux de Petri pour gérer l'interaction des processus Workflow des organisations participantes et les munir de dispositifs de sécurité afin que l'ensemble des Workflows forme un tout (WIO) fonctionnel, sécurisé et performant. Les approches de (Sachin, 2005) et (Mikolajczak and Gami, 2008) se distinguent entre elles par la nature des mécanismes de sécurité déployés par les organisations.

L'article de (Sachin, 2005) propose une méthode incrémentale à base des réseaux de Petri classiques pour incorporer des fonctions de sécurité telles que l'authentification et la non-répudiation au sein d'un WIO. Tout d'abord, le Workflow de chaque organisation participante est modélisé au moyen d'un réseau de Petri. Ensuite, un diagramme de séquence est utilisé pour spécifier les échanges d'informations et l'ordre des évènements entre les organisations participantes. Ce diagramme est combiné à l'ensemble des Workflows locaux afin de créer le WIO. Enfin, les dispositifs de sécurité sont progressivement ajoutées au WIO jusqu'à incorporer autant de dispositifs possibles selon le niveau de sécurité recherché. Par la suite, en appliquant des tests et des simulations, il est possible de valider et de vérifier, avant le déploiement du WIO, la cohérence et la sécurité de l'ensemble des interactions.

Les travaux de (Mikolajczak and Gami, 2008) présentent une méthodologie de modélisation et d'analyse de la sécurité des WIO semblable qui implémente cependant, une politique de sécurité mandataire multi-niveaux (Bell-LaPadula) pour assurer le contrôle d'accès. Cette méthode s'applique dans le cas d'une coopération de WIO de type lâche qui correspond à une coopération occasionnelle entre organisations, sans aucune contrainte structurelle sur la coopération, et où les organisations participantes ne sont pas connues à priori mais recrutées dynamiquement. Tout d'abord, le WIO est modélisé au moyen d'une composition de réseaux de Petri combinés ensemble en se basant sur des diagrammes de séquence. Ensuite, on représente séparément la politique de sécurité de

chaque Workflow sous forme de treillis de niveaux de sécurité. Chacun des treillis associe à chaque sujet et chaque objet, dans une organisation, un niveau de sécurité. Ces treillis indiquent les règles d'accès en lecture et en écriture autorisées dans une organisation selon les règles de Bell-LaPadula afin de contrôler l'accès aux informations secrètes (confidentialité) et éviter la divulgation, c'est-à-dire éviter que les informations d'une classification élevée soient copiées dans les documents d'une classification inférieure. La vérification des accès se fait lors de l'exécution des réseaux de Petri de sorte qu'avant de franchir une transition, le système doit d'abord valider l'accès en se référant au treillis de sécurité de chaque organisation. La transition est franchie seulement si elle est exécutée par un sujet autorisé à accéder les objets requis à son exécution (selon les règles de Bell-LaPadula). Dans le cas où l'accès est refusé, le franchissement n'a pas lieu.

Les propositions de (Sachin, 2005) et (Mikolajczak and Gami, 2008) n'appliquent aucune extension aux réseaux de Petri. L'analyse se fait au moyen de simulation en exécutant le réseau de Petri résultant pour détecter les cohérences dans le WIO correspondant. Dans le cas (Mikolajczak and Gami, 2008), cette simulation est effectuée en combinaison avec des treillis de sécurité. L'inconvénient de ces deux approches réside dans le risque d'avoir un réseau de Petri trop volumineux, même pour des systèmes réels peu complexes.

CHAPITRE 9

CONCLUSION

Dans la présente thèse, nous avons traité de la mise en oeuvre des méthodes formelles dans le cadre de la sécurité des systèmes d'information. En effet, la difficulté de conception des systèmes complexes tient aux problèmes de spécification, d'organisation, d'intégration et de vérification d'ensembles d'actions nombreuses et compliquées. De plus, ces systèmes doivent être fonctionnels, stables, fiables et protégés.

La nature dynamique et hybride des systèmes d'information, nous a conduit à choisir les réseaux de Petri pour vérifier à la fois les propriétés de sûreté et de sécurité. Les réseaux de Petri sont très utilisés dans la modélisation des systèmes à événements discrets et dans les études de sûreté de fonctionnement des systèmes dynamiques. Aussi, plusieurs extensions des réseaux de Petri ont été élaborées pour répondre à la modélisation des problèmes spécifiques et pour maîtriser la taille et la lisibilité des modèles. Récemment, un nombre de propositions intéressantes à base des réseaux de Petri sont apparues pour traiter les problèmes de sécurité et la gestion des autorisations dans les systèmes d'information. En effet, les réseaux de Petri démontrent un atout fondamental qui réside dans les possibilités de validation qu'ils offrent. Ils ont le double avantage de fournir un support graphique naturel qui est d'une aide précieuse dans l'analyse, et de posséder des propriétés analytiques qui permettent une évaluation simple du comportement du système étudié.

Ainsi dans cette thèse, nous avons tenté d'enrichir le formalisme des réseaux de Petri pour raisonner sur le contrôle d'accès, le contrôle de flot d'information et introduire la dimension temporelle, particulièrement dans les systèmes Workflows.

En effet, pour garantir la sécurité d'un système, le contrôle d'accès est un mécanisme qui définit et impose ce qui est permis et interdit de faire. Toutefois, il peut arriver que des utilisateurs autorisés parviennent à acheminer des données légalement acquises à des tierces non autorisées. Pour prendre en compte ce problème, les mécanismes de contrôles d'accès ne suffisent pas et des mécanismes de contrôle de flot d'information sont nécessaires.

D'autre part, le facteur temps joue un rôle primordial dans les SI, notamment pour ajouter une nouvelle dimension aux mesures de sécurité. Si on limitait l'accès aux informations à une période de temps bien précise, on pourra garantir leur validité. En revanche, plus l'information demeure longtemps dans le système, plus son utilisation devient douteuse et par conséquent inaccessible.

Ainsi, nous avons ainsi mis à profit les outils issus des réseaux de Petri pour prendre en compte ces trois éléments ensemble lors de la modélisation et la vérification des SI. Dans ce qui suit, nous synthétiserons les travaux que nous avons effectués, puis, nous exposerons leurs limitations. Enfin, nous donnerons des orientations pour guider les futures recherches dans le domaine.

9.1 Sommaire des contributions

1- **Proposition d'un cadre formel pour modéliser et vérifier des systèmes soumis à des politiques de contrôle d'accès (politiques mandataires multi-niveaux)**. Nous avons proposé une extension du modèle TSCPN qui implémente une politique mandataire multi-niveaux et permet d'exprimer des conditions temporelles sur la disponibilité et la validité des données. En ajoutant le temps, il est possible de maintenir l'historique de l'évolution des données et déterminer les situations où l'information n'a pas été rendu disponible à temps ainsi que celles o l'information n'est probablement

plus valide. L'analyse repose sur la construction du graphe des classes, ce qui malheureusement peut produire un graphe infini et à branchements infinis, si aucune solution n'est utilisée pour y remédier. La solution que nous proposons repose sur des méthodes éfficaces qui consiste d'abord en un calcul symbolique de l'espace d'états. Cette approche consiste à partitionner l'espace d'états en un nombre fini de classes d'états partageant le même marquage. Ensuite, nous avons proposé d'utiliser la relaxation des classes d'états et l'abstraction par inclusion. Nous avons aussi proposé une méthode qui utilise les DBMs, structures de données plus légères à utiliser, pour simplifier le calcul et la relaxation des classes d'états. Nous considérons que cette approche de contraction de l'espace des états est satisfaisantes puisqu'elle permet de préserver les propriétés du modèle que l'on veut vérifier. Finalement, nous avons montré comment exprimer la notion d'opacité pour garantir que les actions secrètes d'un système ne seront pas observables par une tierce partie.

2- **Modélisation des modèles RBAC au moyen des réseaux de Petri Colorés et l'outil CPNtools**. Dans un second temps, nous avons proposé une description modulaire et incrémentale des modèles RBAC au moyen de réseaux de Petri colorés et l'outil CPNtools. Comparativement à d'autres propositions, nous pensons que cette démarche présente les avantages suivants :

- le modèle résultant est une spécification formelle qui exprime les règles et propriétés définissant un modèle RBAC. Ici, nous avons représenté les contraintes de cardinalité, d'héritage et de séparation de tâches. Toutefois, il est possible d'enrichir le modèle par de nouvelles règles ou propriétés en s'appuyant sur le même squelette, d'ajouter de nouveaux modules selon les besoins du contexte ou d'en désactiver si nécessaire.

- la représentation graphique d'un modèle de contrôle d'accès est d'un intérêt pratique indéniable. De plus, l'outil CPNtools offre un environnement de modélisation

et de vérification facile et puissant

- l'analyse comportementale du modèle CPN résultant aide à la détection des incohérences et incomplètudes dans les règles du modèle RBAC.

- Le modèle peut être associé à un Workflow de sorte que les rôles sont accordés selon les tâches que les sujets du système doivent effectuer. Les autorisations sont ainsi vérifiées au fil de l'exécution des activités et le prototypage peut être fait après la validation du modèle, donc après élimination des erreurs de conception détectées par l'analyse;

- la structure des réseaux de Petri permet de synthétiser clairement le flux d'autorisation, de même que les interactions (synchronisation, échange des données) entre les activités de l'application;

3- **Définition d'une extension des réseaux de Petri Colorés à arcs temporels généralisés à plusieurs sémantiques**. Dans un troisième temps, nous avons défini une extension des réseaux de Petri Colorés à arcs temporels généralisés à plusieurs sémantiques. Dans les réseaux de Petri temporels existants, il n'est pas possible d'exprimer plus d'une seule sémantique dans un modèle. Cette limitation peut se répécuter négativement sur la taille du modèle ou encore sa représentativité du système réel. Le modèle $TA_{WS}PN$, quant à lui, offre la possibilité de spécifier pour chaque transition une sémantique de franchissement différente, et ce grce à une étiquette qui indique le type de synchronisation. Nous avons adapté les techniques de construction du graphe des zones au modèle $TA_{WS}PN$ de sorte que le modèle évolue suivant le type de synchronisation. Le graphe des zones est construit en ajoutant dans une zone, les états obtenus par progression de temps, puis en calculant ses successeurs par franchissement d'événements. Toutefois, le graphe des zones d'états résultant peut être infini même pour un modèle borné. Afin d'y remédier à, nous avons proposé d'appliquer une opération d'approximation, qui consiste à étendre

le domaine des horloges en ajoutant des valeurs qui ont la même influence sur le comportement du modèle que celles qui sont déjà dans le domaine.

4- **Application aux systèmes Workflows**. Pour finir, nous avons proposé une approche globale et modulaire, basée sur les réseaux de Petri colorés (par exemple le modèle $TA_{WS}PN$), pour traiter la gestion des autorisations dans les Workflows qui repose sur trois étapes :

A- Modélisation du Workflow

B- Modélisation de la politique RBAC.

C- Connection des deux modèles.

Ici, le modèle RBAC a pour rôle de contrôler l'exécution de l'ensemble des tâches du Workflow. Un sujet ne peut exécuter une tche que s'il a obtenu les rôles nécessaires pour être autorisé à accéder les données requises. Ces dernières doivent de leur part, être disponibles et valides. Par ailleurs, cette approche peut être généralisée à la modélisation des Workflows Inter-Organisationnels (WIO) où chaque workflow peut avoir sa propre politique. Cela a pour but de fournir une étude globale qui tient compte d'un ensemble d'aspects à la fois, afin éviter la dérive entre la spécification et la réalisation du système.

Pour clôturer cette étude, nous abordons dans la section suivante les points qui nous semblent être des prolongements possibles et souhaitables à nos travaux.

9.2 Limitations et perspectives

Nous abordons ici les axes de recherches qu'il reste à explorer ou à approfondir. De façon générale, nous citons les points suivants :

1- **Explosion de la taille du graphe des classes**

En pratique, les procédures de vérification formelle par *model-checking* reposent sur le calcul et l'exploration de l'espace d'états. Bien que nous ayons proposé des méthodes pour contracter l'espace des états et atténuer l'explosion de la taille du graphe de classes, ce problème (à cause des contraintes temporelles entre autres) demeure une limite intrinsèque des méthodes énumératives.

2- **Distance entre le modèle formel et le système réel**

Le recours aux méthodes formelles est un bon moyen pour éviter la dérive entre la spécification et la réalisation d'un système. Or, en général, la réalisation d'un système à partir d'un modèle formel est délicate. À vrai dire, le problème le plus crucial réside dans l'interprétation des spécifications exprimées en langage naturel et de leur traduction en termes mathématiques. Cette traduction suppose malheureusement, une abstraction au cours de laquelle des informations pertinentes peuvent être perdues. Notre modèle n'est donc valable que pour une classe de propriétés considérées et ne représente pas l'intégralité des comportements réels du système. Cependant, dans la pratique, les méthodes formelles restent préférables à une approche informelle.

3- **Développement d'outils**

Les méthodes formelles sont difficiles à appréhender pour un concepteur non spécialiste. Le développement d'outils plus ou moins d'automatisation pour aider le concepteur à étudier les problèmes soulevés est un besoin privilégié pour favoriser l'utilisation du model-checking.

Toutes les limitations énoncées ci-dessus peuvent être des indications pour des recherches futures. Plus particulièrement, nous ajoutons les points suivants :

4- **Enrichissement des modèles**

Il est aussi possible d'envisager l'enrichissement de nos propositions. Étant donné que

le modèle $TA_{WS}PN$ est plus général, nous pourrons y intégrer les notions de sécurité pour exprimer le contrôle d'accès mandataire ou à base de rôles. Ces notions peuvent être représentées de la même façon que dans le modèle $TSCPN$ (pour le contrôle mandataire) et le modèle CPN de RBAC. L'avantage est l'expression des contraintes temporelles suivant différentes sémantiques. Une étude globale, sur des systèmes réels (par exemple les Workflows Inter-Organisationnels), qui tient compte d'un ensemble d'aspects (temporels et contrôle d'accès) pourrait ainsi être envisageable.

Pour conclure, nous mentionnons que la proposition de (Mondal and Sural, 2008) appuie l'idée que la vérification automatique des modèles RBAC est une avenue d'avenir dans le domaine de la sécurité des systèmes d'information. Tel que expliqué dans la section 3.3.3, (Mondal and Sural, 2008) proposent une représentation modulaire d'un modèle RBAC temporel au moyen des automates temporisés. La construction se base sur l'outil Uppaal (Behrmann et al., 2004). Le modèle permet de représenter la hiérachie des rôles grâce à une synchronisation entre des automates temporisés associés à des rôles hiérarchiquement dépendant. Les propriétés de sécurité sont spécifiées en logique CTL.

De plus, sachant que la relation entre les réseaux de Petri temporels et les automates temporisés a déjà fait l'objet de quelques travaux, il pourrait s'avérer intéressant de combiner notre proposition à celle de (Mondal and Sural, 2008).

RÉFÉRENCES

Abadi, M. and Gordon, A. D. (1997). A calculus for cryptographic protocols: The spi calculus. In *Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press.

Abrial, J.-R. (1996). *The B-book: assigning programs to meanings*. Cambridge University Press, New York, NY, USA.

Ahmed, T. and Tripathi, A. R. (2003). Static verification of security requirements in role based cscw systems. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, New York, NY, USA, pages 196–203. ACM.

Amadio, R. M. and Prasad, S. (1994). Localities and failures. In *In Proc. 14th Foundations of Software Technology and Theoretical Computer Science*, pages 205–216. Springer-Verlag.

Arnold, A. (1990). Système de transitions finis et sémantique des processus communicants. *Techniques et Sciences Informatiques*, **9**, 193–215.

Atluri, V. and Huang, W. (1996). An authorization model for workflows. In *ESORICS '96: Proceedings of the 4th European Symposium on Research in Computer Security*, London, UK, pages 44–64. Springer-Verlag.

Attali, I., Bastide, R., Blay-Fornarino, M., Pinna-Déry, A.-M., and Palanque, P. (1998). Spécification formelle d'applications workflow : étude et mise en oeuvre de 3 formalismes. *Technique et Science Informatiques (TSI)*, **17**(2), 181–209.

Barker, S. and Stuckey, P. J. (2003). Flexible access control policy specification with constraint logic programming. *ACM Trans. Inf. Syst. Secur*, **6**, 2003.

Behrmann, G., Bengtsson, J., David, A., Larsen, K. G., Pettersson, P., and Yi, W. (2002). Uppaal implementation secrets. In *FTRTFT '02: Proceedings of the 7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, London, UK, pages 3–22. Springer-Verlag.

Behrmann, G., David, A., and Larsen, K. G. (2004). A tutorial on UPPAAL. In Bernardo, M. and Corradini, F., editors, *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, number 3185 in LNCS, pages 200–236. Springer–Verlag.

Bell, D. E. and LaPadula, L. J. (1973). Secure computer systems: Mathematical foundations. Technical Report MTR-2547, MITRE Corporation, Bedford.

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

Bertino, E., A., B. P., and E., F. (2000). TRBAC: a temporal role-based access control model. In *Symposium on Access Control Models and Technologies. Proceedings of the fifth ACM workshop on Role-based access control*, pages 21–30. ACM.

Biba, K. J. (1977). Integrity considerations for secure computer systems. Technical report, MITRE Corp.

Bodei, C., Degano, P., Nielson, F., and Nielson, H. R. (1999). Static analysis of processes for no and read-up nad no write-down. In *FoSSaCS '99: Proceedings of the Second International Conference on Foundations of Software Science and Computation Structure, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'99*, London, UK, pages 120–134. Springer-Verlag.

Bouyer, P. (2002). Timed automata may cause some troubles. Research Report LSV-02-9, Laboratoire Spécification et Vérification, ENS Cachan, France. 29 pages.

Boyer, M. (juillet 2001). *Contribution à la modélisation des systèmes à temps contraint et application au multimdia*. PhD thesis, University of Toulouse, France.

Boyer, M. and Roux, O. H. (2007). Comparison of the expressiveness of Arc, Place and Transition Time Petri Nets. In Kleijn, J. and Yakovlev, A., editors, *International Conference on Application and Theory of Petri Nets and Other Models of Concurrency (ATPN), Siedlce, Poland, 25/06/07-29/06/07*, LNCS, http://www.springerlink.com/, pages 63–82. Springer-Verlag. (distinction dcerne : Best Paper award).

Braghin, C., Gorla, D., and Sassone, V. (2004). A distributed calculus for role-based access control. *Computer Security Foundations Workshop, 2004. Proceedings. 17th IEEE*, pages 48–60.

Brewer, D. F. C. and Nash, M. J. (1989). The chinese wall security policy. In *IEEE Symposium on Security and Privacy*, pages 206–214.

Brookes, S. D., Hoare, C. A. R., and Roscoe, A. W. (1984). A theory of communicating sequential processes. *J. ACM*, **31**(3), 560–599.

Bryans, J., Koutny, M., and Ryan, P. Y. A. (2004). Modelling dynamic opacity using petri nets with silent actions. In *Formal Aspects in Security and Trust*, pages 159–172.

Bryans, J., Koutny, M., and Ryan, P. Y. A. (2005). Modelling opacity using petri nets. *Electr. Notes Theor. Comput. Sci.*, **121**, 101–115.

Burch, J. R., Clarke, E. M., McMillan, K. L., Dill, D. L., and Hwang, L. J. (1992). Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, **98**(2), 142–170.

Busi, N. and Gorrieri, R. (2003). A survey on non-interference with petri nets. In *Lectures on Concurrency and Petri Nets*, pages 328–344.

Catach, L. (1989). *Les logiques multimodales*. PhD thesis, Universit de Pierre et Marie Curie (Paris 6), Paris, France.

Chellas, B. F. (1980). *Modal Logic: An Introduction*. Cambridge University Press, Cambridge, MA.

Chen, L., Anderson, S., and Moller, F. (1990). A timed calculus of communicating systems. Technical Report report-90-127, Edinburgh University, LFCS.

Choppy, C., Petrucci, L., and Reggio, G. (2007). Designing coloured petri net models: a method. In *Proceedings of the 8th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools Aarhus*, Denmark, pages 22–24.

Church, A. (1936). An unsolvable problem of elementary number theory. *Am. J. Math.*, **58**, 345–363.

Clark, D. D. and Wilson, D. R. (1987). A comparison of commercial and military computer security policies. *sp*, **00**, 184.

Cormen, T. T., Leiserson, C. E., and Rivest, R. L. (1990). *Introduction to algorithms*. MIT Press, Cambridge, MA, USA.

CTCPEC (1993). The canadian trusted computer product evaluation criteria. *Canadian System Security Center, Communications Security Establishment of Canada*. V. 3.0e.

De Michelis, G., Ellis, C., and Memmi, G., editors (1994). *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, Zaragoza, Spain.

Diaz, M. (2001). *Les Réseaux de Petri - Modèles fondamentaux*. Paris.

Dill, D. L. (1990). Timing assumptions and verification of finite-state concurrent systems. In *Proceedings of the international workshop on Automatic verification methods for finite state systems*, New York, NY, USA, pages 197–212. Springer-Verlag New York, Inc.

Drouineaud, M., Bortin, M., Torrini, P., and Sohr, K. (2004). A first step towards formal verification of security policy properties for RBAC. In *QSIC '04: Proceedings of the Quality Software, Fourth International Conference*, Washington, DC, USA, pages 60–67. IEEE Computer Society.

Ellis, C., Keddara, K., and Rozenberg, G. (1995). Dynamic change within workflow systems. In *COCS '95: Proceedings of conference on Organizational computing systems*, New York, NY, USA, pages 10–21. ACM.

Esparza, J. and Silva, M. (1991). Circuits, handles, bridges and nets. In *Proceedings of the 10th International Conference on Applications and Theory of Petri Nets*, London, UK, pages 210–242. Springer-Verlag.

Gardey, G., Roux, O. H., and Roux, O. F. (2003). Using zone graph method for computing the state space of a time petri net. In Larsen, K. G. and Niebert, P., editors, *FORMATS*, volume 2791 of *Lecture Notes in Computer Science*, pages 246–259. Springer.

Gavrila, S. I. and Barkley, J. F. (1998). Formal specification for role based access control user/role and role/role relationship management. In *RBAC '98: Proceedings of the third ACM workshop on Role-based access control*, New York, NY, USA, pages 81–90. ACM.

Genrich, H. J. (1987). Predicate/transition nets. In *Proceedings of an Advanced Course on Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986-Part I*, London, UK, pages 207–247. Springer-Verlag.

Gou, H., Huang, B., Liu, W., Li, Y., and Ren, S. (2001). Modeling distributed business processes of virtual enterprises based on the object-oriented approach and petri nets. *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, **3**, 2052–2057 vol.3.

Hadjidj, R. and Boucheneb, H. (2005). Much compact time petri net state class spaces useful to restore ctl* properties. In *ACSD '05: Proceedings of the Fifth International Conference on Application of Concurrency to System Design*, Washington, DC, USA, pages 224–233. IEEE Computer Society.

Hansen, F. and Oleshchuk, V. A. (2005). Conformance checking of RBAC policy and its implementation. In *ISPEC*, pages 144–155.

Hoare, C. A. R. (1978). Communicating sequential processes. *Commun. ACM*, **21**(8), 666–677.

Holzmann, G. J. (1997). The model checker SPIN. *Software Engineering*, **23**(5), 279–295.

ITSEC (1991). Information technology security evaluation criteria. *Commission European Communities*.

Jablonski, S. and Bussler, C. (1996). *Workflow Management. Modeling Concepts, Architecture and Implementation*. London et al.: International Thomson Computer Press.

Jackson, D. (2004). *ALLOY 3.0 Reference Manual*.

Jackson, D., Schechter, I., and Shlyahter, H. (2000). Alcoa: the alloy constraint analyzer. In *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, New York, NY, USA, pages 730–733. ACM.

Jagadeesan, R., Jeffrey, A., Pitcher, C., and Riely, J. (2006). λ-RBAC: Programming with role-based access control. In *In ICALP 06*, pages 456–467. Springer.

JCSEC (1992). The japanese computer security evaluation criteria. *Le ministre de l'industrie et du commerce*. Draft V1.0.

Jensen, K. (1997). *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. Three Volumes.

Jensen, K., Kristensen, L. M., and Wells, L. (2007). Coloured petri nets and CPN tools for modelling and validation of concurrent systems. *STTT*, **9**(3-4), 213–254.

Jiang, Y., Lin, C., Yin, H., and Tan, Z. (2004). Security analysis of mandatory access control model. *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, **6**, 5013–5018 vol.6.

Joshi, J. (2003). *A Generalized Temporal Role Based Access Control Model for Developing Secure Systems*. PhD thesis, Purdue Univ.

Juopperi, J. (1995). PrT-net based analysis of information flow security nets. Research Report A34, Helsinki University of Technology, Department of Computer Science and Engineering, Digital Systems Laboratory, Espoo, Finland.

Juszczyszyn, K. (2003). Verifying enterprise's mandatory access control policies with coloured petri nets. In *WETICE '03: Proceedings of the Twelfth International Workshop on Enabling Technologies*, Washington, DC, USA, page 184. IEEE Computer Society.

Kalam, A. A. E., Benferhat, S., Miege, A., Baida, R. E., Cuppens, C., Saurel, C., Balbiani, P., Deswarte, Y., and Trouessin, G. (2003). Organization based access control. *policy*, **00**, 120.

Kalam, A. A. E. and Deswarte, Y. (2003). Security model for health care computing and communication systems. In *SEC*, pages 277–288.

Khansa, W. (1997). *Réseaux de Petri p-temporels: contribution à l'étude des systèmes à événements discrets*. PhD thesis, University of Savoie, France.

Kleene, S. C. (1967). *Mathematical Logic*. New York, Wiley.

Knorr, K. (2000). Dynamic access control through petri net workflows. In *ACSAC '00: Proceedings of the 16th Annual Computer Security Applications Conference*, Washington, DC, USA, page 159. IEEE Computer Society.

Knorr, K. (2001). Multilevel security and information flow in petri net workflows. Technical report, In: Proceedings of the 9th International Conference on Telecommunication Systems - Modeling and Analysis, Special Session on Security Aspects of Telecommunication Systems.

Koch, M., Mancini, L. V., and Parisi-Presicce, F. (2002). A graph-based formalism for RBAC. *ACM Trans. Inf. Syst. Secur.*, **5**(3), 332–365.

Kumar, A., Karnik, N., and Chafle, G. (2002). Context sensitivity in role-based access control. *SIGOPS Oper. Syst. Rev.*, **36**(3), 53–66.

Laborde, R., Nasser, B., Grasset, F., Barrre, F., and Benzekri, A. (2004). A formal approach for the evaluation of network security mechanisms based on RBAC policies. In *2nd International Workshop on Security Issues with Petri Nets and other Computational Models (WISP'04), Bologna, Italy*, pages 117–142. Electronic Notes in Theoretical Computer Science, Volume 121, Elsevier. Dates de conférence : juin 2004.

Lampson, B. W. (1974). Protection. *SIGOPS Oper. Syst. Rev.*, **8**(1), 18–24.

Ling, S. and Schmidt, H. (2000). Time petri nets for workflow modelling and analysis. *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, **4**, 3039–3044 vol.4.

Liu, J. and Sun, L. (2004). The application of role-based access control in workflow management systems. *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, **6**, 5492–5496 vol.6.

Lu, Y., Zhang, L., Liu, Y., and Sun, J. (2006). Using *pi*-calculus to formalize domain administration of RBAC. In *ISPEC*, pages 278–289.

Mahdaoui, L., Boukhedouma, S., and Alimazighi, Z. (2005). Modélisation de processus e-learning par les workflows. In *3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications (SETIT)*, Tunisia.

Mankai, M. (2005). Vérification et analyse des politiques de controle d'acces : application au langage xacml. Master's thesis, Université du Québec en Outaouais.

Mclean, J. (1987). Reasoning about security models. *Security and Privacy, IEEE Symposium on*, **0**, 123.

Meadows, C. (1996). The NRL protocol analyzer: An overview. *J. Log. Program.*, **26**(2), 113–131.

Melliar-Smith, M. and Rushby, J. (1985). The enhanced HDM system for specification and verification. *SIGSOFT Softw. Eng. Notes*, **10**(4), 41–43.

Merlin, P. M. (1974). *A study of the recoverability of computing systems*. PhD thesis, Department of Information and Computer Science, University of California, Irvine, CA.

Mikolajczak, B. and Gami, N. (2008). Design and verification of loosely coupled inter-organizational workflows with multi-level security. *JCP*, **3**(1), 63–78.

Milner, R. (1980). *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer.

Milner, R. (1989). *Communication and concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Milner, R., Parrow, J., and Walker, D. (1993). Modal logics for mobile processes. *Theoretical Computer Science*, **114**(1), 149–171.

Mondal, S. and Sural, S. (2008). A verification framework for temporal rbac with role hierarchy (short paper). In *ICISS*, pages 140–147.

Oren, E. and Haller, A. (2005). Formal frameworks for workflow modelling. Technical report, DERI.

Osborn, S. L. (2002). Information flow analysis of an RBAC system. In *In 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002*, pages 163–168. Press.

Owre, S., Rushby, J. M., , and Shankar, N. (1992). PVS: A prototype verification system. In Kapur, D., editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, Saratoga, NY, pages 748–752. Springer-Verlag.

Parrow, J. (2001). *An introduction to the pi-calculus*, pages 479–543. Handbook of Process Algebra. Elsevier.

Petri, C. A. (1962). *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn.

PRG (1980). *Programming Research Group*. Oxford University.

Rakkay, H. (2005). Modélisation et vérification du contrôle de flots d'informaton au moyen de réseaux de petri colorés temporisés. Master's thesis, École Polytechnique de Montréal.

Rakkay, H. and Boucheneb, H. (2006). Timed secure colored petri net based analysis of information flow. *Annals of Telecommunications*, pages 1314–1346.

Rakkay, H. and Boucheneb, H. (2007). Using timed colored petri net to formalize temporal role based access control policies. In *Proceedings of the 7th International Conference on New Technologies of Distributed Systems*, Marrakesh, Morocco, pages 109–120.

Rakkay, H. and Boucheneb, H. (2009). Security analysis of role based access control models using colored petri nets and cpntools. *Transactions on Computational Science*, **4**, 149–176.

Rakkay, H., Boucheneb, H., and Roux, O. H. (2007). Réseaux de Petri à arcs temporels généralisés aux sémantiques faible et forte. In *6ieme Colloque Francophone sur la Modélisation des Systèmes Réactifs, (MSR'07)*, Lyon, France.

Ramchandani, C. (1974). Analysis of asynchronous concurrent systems by timed petri nets. Technical report, Cambridge, MA, USA.

Reisig, W. (1985). *Petri Nets: An Introduction*, volume 4 of *Monographs in Theoretical Computer Science. An EATCS Series*. Springer.

Riely, J. and Hennessy, M. (1998). A typed language for distributed mobile processes. In *In Proceedings of the 25th POPL*, pages 378–390. ACM Press.

Russell, N., van der Aalst, W. M. P., ter Hofstede, A. H. M., and Edmond, D. (2005). Workflow resource patterns: Identification, representation and tool support. In *CAiSE*, pages 216–232.

Salimifard, K. and Wright, M. (2001). Petri net-based modelling of workflow systems: An overview. *European Journal of Operational Research*, **134**(3), 664–676.

Sandhu, R. and Munawer, Q. (1999). The ARBAC99 model for administration of roles. In *ACSAC '99: Proceedings of the 15th Annual Computer Security Applications Conference*, Washington, DC, USA, page 229. IEEE Computer Society.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *Computer*, **29**(2), 38–47.

Schaad, A., Moffett, J., and Jacob, J. (2001). The role-based access control system of a european bank: a case study and discussion. In *SACMAT '01: Proceedings of the sixth ACM symposium on Access control models and technologies*, New York, NY, USA, pages 3–9. ACM.

Schneider, F. B. (2000). Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, **3**(1), 30–50.

Shafiq, B., Masood, A., Joshi, J., and Ghafoor, A. (2005). A role-based access control policy verification framework for real-time systems. In *WORDS '05: Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems*, Washington, DC, USA, pages 13–20. IEEE Computer Society.

Shamir, A. (1979). How to share a secret. *Commun. ACM*, **22**(11), 612–613.

Shin, W. (2005). *An Extension of Role Based Access Control for Trusted Operating Systems and Its Coloured Petri Net Model.* PhD thesis, Gwangju Institute of Science and Technology, Korea.

Spivey, J. M. (1992). *The Z notation: a reference manual.* Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK.

Talbot, J. (2005). Model-checking pour les ambients : des algbres de processus aux donnes semi-structures. Master's thesis, Universit des sciences et technologies de Lille, France.

Thomas, R. K. (1997). Team-based access control (TMAC): a primitive for applying role-based access controls in collaborative environments. In *RBAC '97: Proceedings of the second ACM workshop on Role-based access control*, New York, NY, USA, pages 13–19. ACM.

Thomas, R. K. and Sandhu, R. S. (1998). Task-based authorization controls (tbac): A family of models for active and enterprise-oriented autorization management. In *Proceedings of the IFIP TC11 WG11.3 Eleventh International Conference on Database Securty XI*, London, UK, UK, pages 166–181. Chapman & Hall, Ltd.

Van der Aalst, W. (1996). Three good reasons for using a petri-net-based workflow management system. In Navathe, S. and Wakayama, T., editors, *Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC96)*, Camebridge, Massachusetts, pages 179–201.

Van der Aalst, W. (1998). The application of petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, **8**(1), 21–66.

van der Aalst, W. M. P. (1993). Interval timed coloured petri nets and their analysis. In *Proceedings of the 14th International Conference on Application and Theory of Petri Nets*, London, UK, pages 453–472. Springer-Verlag.

Varadharajan, V. (1991). Hook-up property for information flow secure nets. *Computer Security Foundations Workshop IV, 1991. Proceedings*, pages 154–175.

Walter, B. (1983). Timed net for modeling and analysing protocols with time. In *IFIP Conference on Protocol Specification Testing and Verification*, North Holland.

Yi, Z., Yong, Z., and Weinong, W. (2004). Modeling and analyzing of workflow authorization management. *J. Netw. Syst. Manage.*, **12**(4), 507–535.

Yu, Y., Tang, Y., Liang, L., and sheng Feng, Z. (2004). Temporal extension of workflow meta-model and its application. *Computer Supported Cooperative Work in Design, 2004. Proceedings. The 8th International Conference on*, **2**, 293–297 Vol.2.

Yuan, C., He, Y., He, J., and Zhou, Z. (2006). A verifiable formal specification for RBAC model with constraints of separation of duty. In *Inscrypt*, pages 196–210.

Zao, J., Wee, H., Chu, J., and Jackson, D. (2002). RBAC schema verification using lightweight formal model and constraint analysis. Technical report, MIT, Cambridge.

Zhang, Z.-L., Hong, F., and Liao, J.-G. (2006a). Modeling chinese wall policy using colored petri nets. In *CIT '06: Proceedings of the Sixth IEEE International Conference on Computer and Information Technology*, Washington, DC, USA, page 162. IEEE Computer Society.

Zhang, Z.-L., Hong, F., and Xiao, H.-J. (2006b). Verification of strict integrity policy via petri nets. In *ICSNC '06: Proceedings of the International Conference on Systems and Networks Communication*, Washington, DC, USA, page 23. IEEE Computer Society.