TOUHAMI MECHRI

Approche algébrique pour la sécurisation des réseaux informatiques

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en informatique
pour l'obtention du grade de Maître ès Sciences (M. Sc.)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

2007

Résumé

Se procurer les outils les plus récents et les plus performants liés à la sécurisation de réseaux informatique est loin d'être suffisant pour réduire les risques d'intrusions. En effet, le maillon le plus faible dans la chaîne de la sécurité informatique est souvent l'intervention humaine qui est parfois nécessaire pour installer et configurer ces outils. Limiter cette intervention humaine permettra sans doute de réduire à la fois les risques et les coûts engendrés par la sécurité. Il est important, par exemple, de développer des méthodes sûres permettant de configurer automatiquement un réseau informatique de sorte que son comportement soit conforme à une politique de sécurité donnée.

C'est dans cet axe de recherche que se situe ce travail. En effet, nous proposons une méthode formelle permettant de générer à partir d'une politique de sécurité (spécifiée par une formule logique) et d'un réseau informatique (spécifié par un processus) une configuration sécuritaire de ce réseau.

Avant-propos

Toute ma reconnaissance chaleureuse et sincère va à mon directeur de recherche, Dr. Mohamed Mejri, pour sa grande disponibilité, sa patience et ses conseils.

Je remercie également les professeurs Dr. Mourad Debbabi et Dr. Béchir Ktari, pour avoir accepté de lire et évaluer ce travail.

Je remercie sincèrement mes parents, ma fidèle femme Djamila, mes frères Kamel, Fahim, Nadir, Zakaria, Anouar, Saif Eddinne et mes soeurs Merriem, Sarra, Besma, Souria et tous mes cousins (Yahia, Faicel, Abd El Majid, etc.) pour leur encouragement et soutien.

Je tiens à exprimer ma sincère gratitude à M. Bachir Halimi le président de l'entreprise d'Excendia Inc. et M. Djamel Benredjem qui m'ont aidé pendant toute la durée de mes études par leurs conseils et leurs collaborations.

Des remerciements particuliers s'adressent au gouvernement algérien qui m'a offert une bourse d'étude pour mener à terme ma maîtrise.

J'adresse mes meilleurs et sincères remerciements à mon ami Mahjoub Langar pour ses conseils et son aide durant tout ce travail.

Pour terminer, je remercie tous mes collègues du groupe LSFM et en particulier M. Chamseddine Talhi, Mme Hakima Ould-Slimane et Mme Syrine Ayadi pour leurs conseils et leurs aides.

[À ma mère

À mon père]

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

Motivations

La dépendance de la société moderne relativement aux systèmes d'informations et aux réseaux de télécommunication et d'information est devenue inévitable. En effet, ces systèmes d'informations et ces réseaux sont devenus des outils indispensables aux fonctionnements et à l'évolution de toutes les activités des entreprises de notre société. Ils sont aujourd'hui déployés dans tous les secteurs de la vie courante : les banques, les assurances, les hôpitaux, etc. Internet de sa part a donnée aux entreprises comme aux citoyens d'innombrables possibilités de nouveaux produits et services, amélioration de services existants et de gain de temps et d'argent. La seule limite est l'imagination ! Cependant, cette nouvelle technologie a crée des nouveaux problèmes dont le plus important est la sécurité. Pour toutes sortes de raisons (argent, concurrence, revanche, terrorisme, célébrité, etc.), certaines personnes veulent s'introduire dans les systèmes d'informations menaçant ainsi leurs intégrités et leur bon fonctionnement. Dès lors, la sécurisation des systèmes d'informations et des réseaux informatiques est devenue un enjeu important. De nombreuses techniques et outils ont été développés durant les dernières années pour améliorer la sécurité de ces systèmes. Parmi ces techniques et outils, nous trouvons les systèmes de détection d'intrusions (IDS) et les pare-feux. Les IDSs permettent de surveiller les actions effectuées sur un système (machine, réseau, etc.) à la recherche de toutes activités douteuses afin de les signaler aux administrateurs qui prendront les décisions nécessaires pour prévenir les problèmes ou limiter les dégâts. Les pare-feux, quant-à-eux, ont un rôle plus actif et ils permettent de supprimer du trafic tous les paquets indésirables en suivant une politique de sécurité donnée.

Malgré l'importante évolution que la sécurité informatique a vu durant les dernières années, le problème est loin d'être résolu et toute nouvelle contribution est la bienvenue. Par

exemple, la configuration des réseaux informatiques, en mettant en place une politique de sécurité, est une tâche qui se complique en fonction de la taille du réseau et de sa topologie, et ce même pour les spécialistes en sécurité. Une erreur dans ces configurations peut engendrer des conséquences coûteuses et parfois irrémédiables. De ce fait, il est important de développer des méthodes formelles permettant de configurer automatiquement un réseau informatique de sorte que son comportement soit conforme à une politique de sécurité donnée.

Objectifs et méthodologies

Le principal objectif de la présente recherche est l'élaboration d'une technique formelle permettant de forcer un réseau informatique à respecter une politique de sécurité donnée. À partir d'un réseau et une politique de sécurité, nous voulons, dans la mesure du possible, configurer les machines de ce réseau de sorte à le contraindre à respecter la politique de sécurité. Pour ce faire, nous procédons comme suit :

- définir un langage formel (logique) permettant la spécification des politiques de sécurité liées aux réseaux informatiques.
- définir un langage formel (algèbre de processus) pour la spécification des réseaux informatiques.
- définir un opérateur de renforcement qui renforce la politique de sécurité en question.
- prouver que l'opérateur de renforcement a toutes les propriétés désirées (correction, etc.).
- raffiner l'opérateur en question pour donner la possibilité à l'utilisateur de faire le monitorage sélectif : l'utilisateur peut choisir les composantes (les machines) réseau dans lesquelles il veut placer ses moniteurs (sa politique de sécurité).

Organisation

La première partie de ce mémoire est consacrée à l'état de l'art dans le domaine de la sécurité des réseaux informatiques et des méthodes formelles de spécification et de vérification de systèmes distribués. En particulier, nous trouvons dans le chapitre suivant les techniques de sécurisation utilisées dans les réseaux informatiques : les systèmes de détection d'intrusions et les pare-feux. Dans le troisième chapitre, nous rappelons certaines algèbres de processus (CCS, π-calcul, le calcul ambiant) et logiques temporelles (LTL et CTL).

Dans la deuxième partie de ce mémoire, nous présentons notre contribution. Il s'agit d'une technique formelle qui utilise à la fois une algèbre de processus et une logique. En

s'inspirant de certain travaux existants, nous définissions une algèbre de processus, nommé CMN (Calculus Monitored Network), et une logique propositionnelle, nommée L_M. Nous utilisons l'algèbre CMN pour modéliser les réseaux informatiques et le langage L_M pour spécifier des politiques de sécurité. De plus, nous proposons un opérateur de renforcement qui permet, à partir d'un réseau et d'une politique de sécurité de générer un nouveau réseau sécuritaire équivalent au premier. Enfin, nous présentons un nouvel opérateur de renforcement qui donne à l'utilisateur la possibilité de choisir l'emplacement des moniteurs au sein du réseau en question.

Ce mémoire se termine par une conclusion qui discute les principales contributions de ce travail tout en donnant quelques nouvelles perspectives.

Chapitre 2

Sécurité des réseaux informatiques

De nos jours, la plupart des entreprises possèdent des réseaux locaux qui sont connectés à Internet. Cette ouverture vers l'extérieur est à la fois indispensable et dangereuse en même temps. En effet, ouvrir l'entreprise vers le monde signifie aussi laisser la porte ouverte aux étrangers pour essayer de pénétrer le réseau local de l'entreprise, et y accomplir des comportements malicieux (vol d'informations confidentielles, destruction, etc.). Pour parer à ces attaques, la mise en place d'une architecture sécurisée est indispensable. La plupart des architectures d'aujourd'hui sont basées principalement sur des systèmes de détection d'intrusions et des pare-feux. Ces techniques ont pour but de sécuriser au maximum le réseau local de l'entreprise, de détecter les tentatives d'intrusions et d'y parer au mieux possible.

Les systèmes de détection d'intrusions ont pour objectif de détecter toute violation de politiques de sécurité sur un système informatique. Ils permettent ainsi de signaler des attaques (en temps réel ou en différé) portant atteinte à la sécurité de ce système. Les pare-feux, de leur part, offrent un véritable contrôle sur le trafic entrant et sortant dans le but d'empêcher toute tentative de violation d'une politique de sécurité donnée.

Dans ce chapitre, nous étudions brièvement les systèmes de détection d'intrusions et les pare-feux.

2.1 Systèmes de détection d'intrusions

Un système de détection d'intrusions (IDS) est un mécanisme destiné à repérer des activités anormales ou suspectes sur une cible donnée afin de remédier aux problèmes dans

les plus brefs délais. Vu leur utilité pratique, les IDS ont été étudiés massivement durant les 20 dernières années dans le but d'améliorer leur efficacité. Les fruits de ces études sont des différentes classes d'IDSs qui se basent sur différentes techniques de détection dont chacune est mieux appropriée pour un contexte bien particulier. Entre autres, nous trouvons les systèmes de détection d'intrusions qui basent leurs décisions sur des informations trouvées dans des machines hôtes et appelés HIDS et les systèmes de détections d'intrusions qui fondent leurs décisions uniquement sur des informations qui circulent dans un réseau et qui sont appelés NIDS. Plus de détails sur les différentes classes des IDS ainsi que leur évolution sont disponibles dans [33].

Dans ce qui suit, nous donnons, en premier lieu, un bref aperçu sur les vulnérabilités des systèmes informatiques ainsi que la notion d'audit de sécurité. Ensuite nous introduisons les deux grandes approches d'IDS les plus utilisées : l'approche comportementales et l'approche par scénarios. Nous continuons avec une méthode récente basée sur l'utilisation des agents mobiles pour la détection d'intrusions. Nous terminons cette section par présenter quelques outils de détection d'intrusions.

2.1.1 Vulnérabilité des systèmes

Une attaque est une exploitation d'une vulnérabilité présente dans un système. De ce fait, réduire les attaques ne peut se faire qu'avec une bonne compréhension du système et des possibles sources de vulnérabilité afin de trouver les remèdes convenables. Le mot vulnérabilité exprime toutes les faiblesses des ressources informatiques qui peuvent être exploitées par des personnes mal-intentionnées. Dans [17], D. Denning explique la présence de vulnérabilités dans des systèmes d'informations par, entre autres, les raisons suivantes :

- Une bonne sécurité coûte généralement très chère et la plupart des organismes n'ont pas le budget suffisant pour s'offrir ce besoin.
- Les outils de sécurité utilisés ne peuvent pas être sûrs à 100%, voir qu'ils sont souvent inefficaces.
- Les politiques de sécurité sont couramment complexes, incomplètes et parfois inconsistantes.

Dans [52], A. Sundaram ajoute d'autres raisons comme :

- les bugs dans les programmes qui sont courants et qui sont toujours exploitables par les attaquants.
- les faiblesses dues à la gestion et à la configuration des systèmes.

2.1.2 Audit de sécurité

L'audit de sécurité permet d'enregistrer tout ou une partie des actions effectuées sur le système. L'analyse de ces informations permet de détecter d'éventuelles intrusions. Les systèmes d'exploitation disposent généralement d'un système d'audit intégré. Les différents événements du système sont enregistrés dans un journal d'audit qui devra être analysé fréquemment, voire en permanence. Dans les réseaux, il est indispensable de se disposer d'une base d'audit permettant d'estampiller et d'enregistrer certains événements.

Selon les travaux de Mé et Alanou indiqué dans [33], il s'agit de collecter des informations concernant les différentes composantes (processus, mémoire, fichiers, entrées/sorties, etc.) du système afin de comprendre le "qui a fait à quoi, quand et comment ?" et d'empêcher les intrusions dans la mesure du possible. L'audit permet également d'obtenir des informations relatives à chaque application (le lancement ou l'arrêt des différents modules, les variables d'entrée et de sortie et les différentes commandes exécutées). D'autres informations sur les violations éventuelles de la sécurité (exemple : tentatives de commandes non autorisées) ainsi que des statistiques sur l'utilisation de certaines composantes (cpu, réseau, etc.) du système peuvent également être collectées.

2.1.3 Classification des systèmes de détection d'intrusions

1. **Approche par scénarios**

   Cette approche consiste à chercher dans les activités de l'entité surveillée les empreintes ou les signatures d'attaques connues. Chacune de ces signatures décrit une attaque bien précise et chaque attaque peut être détectée par un seul ou une séquence d'événements obtenus à partir d'une ou plusieurs sondes (collecteur d'informations). Ces derniers permettent de classifier tous les événements d'attaques qui peuvent provenir, soit d'un hôte (exemple : fichiers audit, trace d'exécution des commandes, etc.), soit d'un réseau. La figure 2.1 représente un modèle générique de système de détection d'intrusions adapté pour l'approche par scénarios. Cette démarche est très similaire à celle des outils antivirus et présente les mêmes inconvénients que ceux-ci. Il est aisé de comprendre que ce type d'IDSs ne peut détecter que les attaques dont ils possèdent les signatures. Ils nécessitent également des mises à jour régulières de leur base de signatures et leur efficacité dépend du contenu de cette base. Si les signatures sont erronées ou incorrectement conçues, l'ensemble du système est par conséquent inefficace. Ce modèle est par contre très simple à implémenter et à optimiser.

   Plusieurs mécanismes ont été proposés afin de localiser les signatures d'attaques dans

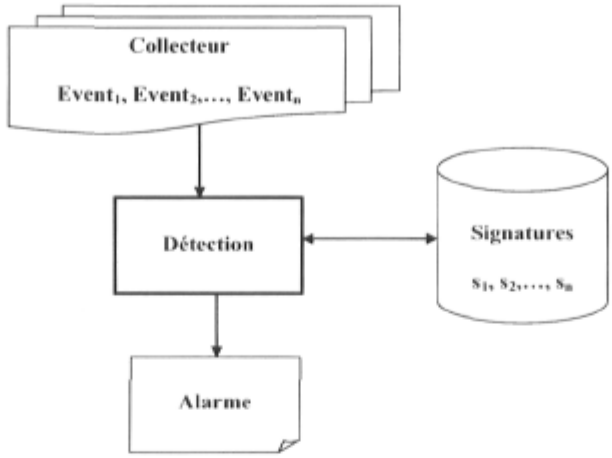


FIG. 2.1 – Modèle de détection pour l'approche par scénarios.

les traces d'audit. Parmi ces mécanismes, on peut citer :

- **Analyse des transitions d'états** : les signatures d'attaques sont vues comme des systèmes de transitions étiquetées. En Partant d'un état initial et en analysant les séquences d'actions effectuées sur le système, nous pouvons détecter des états indésirables qui reflètent des tentatives d'intrusions.
- **Réseaux de neurones** : Les réseaux de neurones sont souvent utilisés pour répartir en différentes classes une population (un ensemble d'individus). Pour la détection d'intrusions, la population est l'ensemble d'actions effectuées sur le système et on cherche à les répartir en au moins deux classes : les actions malicieuses ou douteuses et les actions non malicieuses. Grâce à leur flexibilité et leur rapidité, les réseaux de neurones permettent de faire une analyse efficace du flux d'audit en temps réel. Il est cependant difficile de comprendre dans tous les cas les raisons qui ont amené à placer une action dans une classe ou dans une autre.
- **Reconnaissance de forme** (Pattern Matching) : Il s'agit de représenter les signatures d'attaques par des séquences abstraites (avec possibilité d'inclure des variables) d'événements, de modéliser le trafic par une séquence concrète d'actions et de voir s'il est possible de les unifier.

2. **Approche comportementale**

Cette approche à été proposée par J. Anderson dans [7] en 1980, puis révisée et étendue dans [18] par D. Denning en 1987. Elle consiste à détecter si un utilisateur a fait un comportement anormal par rapport à ses habitudes. Elle utilise pour cela un modèle statistique développé par Denning dans [18] et elle se base sur un profil du comportement normal de l'utilisateur, au vu de plusieurs variables aléatoires. Lors de l'analyse, on calcule un taux de déviation entre le comportement courant et le comportement passé.

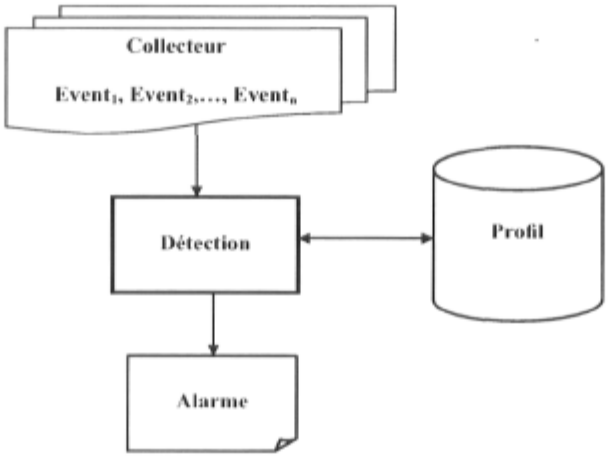


FIG. 2.2 – Modèle de détection pour l'approche comprotementale.

Si ce taux dépasse un certain seuil, le système déclare qu'il est attaqué. Par exemple, un employeur qui travaille dans une compagnie et qui se connecte la nuit à certaines heures, en plus de la journée pourrait amener l'IDS à signaler un comportement inhabituel.

Le principal avantage des IDS comportementaux est la détection de nouveaux types d'attaques. En effet, ces IDS ne sont pas programmés pour reconnaître des attaques spécifiques mais plutôt pour signaler toute activité anormale. La figure 3.1 montre un exemple de modèle de détection utilisant l'approche par profil. Cette dernière utilise le profil construit à partir des événements passés pour le comparer aux événements actuels du collecteur [35]. Cependant, cette approche peut entraîner beaucoup de fausses alertes comme il est possible de ne pas détecter certaines attaques.

L'approche comportementale est basée sur plusieurs techniques, on citera dans ce qui suit quelques-unes :

- **Observation de seuils** : Il s'agit de l'utilisation de la méthode de classification de BAYE (observation du seuil). Cette méthode permet de fixer le comportement normal d'un utilisateur par la donnée de seuils à certaines mesures (par exemple le nombre maximum de mots de passe erronés). Si le comportement d'un utilisateur dévie sur la valeur du seuil, le model de détection génère une alarme.
- **Profilage des utilisateurs [12]** : L'idée est d'établir des profils individuels de chacun des usagers du système. Au fur et à mesure que l'utilisateur change ses activités, son profil de travail attendu se met à jour. Il reste cependant difficile de déterminer un profil pour un utilisateur irrégulier ou très dynamique.
- **Profilage des groupes** : Pour réduire le nombre de profil à gérer, on classe les utilisateurs par groupes. Le profil d'un groupe est par la suite calculé en fonction de

l'historique de ses activités. On vérifie que les individus du groupe travaillent d'une manière uniforme et ne dévient pas par rapport à ce qu'a été défini comme profil du groupe. Cependant, il n'est pas évident de trouver le groupe le plus approprié à une personne. D'ailleurs, il est parfois nécessaire de créer un groupe à un seul individu.

- **Profilage d'utilisation de ressources** : Il s'agit d'observer l'utilisation de certaines ressources (comme les CPU, les ports de communication, les comptes, les applications, la mémoire) sur de longues périodes et de comparer les activités courantes par rapport à ce qui a été observé dans le passé. On peut aussi observer les changements dans l'utilisation des protocoles réseau et rechercher les ports qui voient leur trafic augmenter anormalement. Les expériences ont montré, cependant, qu'il est difficile d'interpréter les écarts par rapport au profil normal.

- **Profilage de programmes exécutables** : Certains virus, chevaux de Troie et d'autres programmes malveillants peuvent être détectés en profilant la façon dont les objets du système comme les fichiers ou les imprimantes sont utilisées par les programmes de confiance. En d'autres termes, le profilage de programmes exécutables consiste à observer l'utilisation des ressources du système par certains programmes exécutables dans le but de détecter des déviations par rapport à ces comportements. On peut par exemple détecter le fait qu'un serveur se met à écouter sur des ports autres que ceux qu'il utilise d'habitude.

- **Profilage statistique** : Denning a défini dans [18] une approche statistique permettant de modéliser les comportements des utilisateurs d'un système. Ce modèle statistique permet de déterminer, au vu de n observations $x_1, ..., x_n$ faites sur une variable x, si la valeur x_{n+1} de l'observation $(n+1)$ est acceptable ou non. Explicitement, un profil est constitué d'un ensemble de mesures représentant certaines statistiques sur des événements (exemple : nombre de fois qu'une commande système particulière à été exécutée par un utilisateur, nombre de quantums du temps CPU occupés par un programme, etc.) pendant une certaine période de temps.

- **Approche immunologique** : L'approche immunologique tente de reproduire le comportement des systèmes immunologiques réels pour faire la différence entre ce qui est normal et ce qui ne l'est pas. En fait, l'approche comportementale se base sur la connaissance de ce qui est bien et vérifiée en permanence que l'activité du système est normale. L'approche immunologique propose de rechercher ce qui est mal en connaissant ce qui est bien. En clair, l'approche immunologique consiste à construire un modèle de comportement normal des services à travers des courtes séquences d'appels système. En phase d'apprentissage, on construit une base de séquences d'appels normaux à partir de l'observation d'un service pendant un certain temps. Toute séquence étrangère à cet ensemble est considérée comme une potentielle intrusion.

2.1.4 Utilisation des agents mobiles dans les systèmes de détection d'intrusions

Le domaine de recherche sur les agents mobiles est relativement récent (voir [48]). Le concept d'agents mobiles dans le réseau est apparu en 1994 avec Telescript de General Magic [54] qui ont proposé un système fermé pour le commerce électronique. Les agents sont des entités capables de migrer leur exécution d'une place à une autre via l'instruction **go**, de rencontrer d'autres agents sur la même place via l'instruction **meet** et de communiquer avec d'autres agents sur d'autres places par des connections permettant l'envoi de messages. Beaucoup de travaux de recherche ont été effectués dans le but d'affiner les environnements de développement des systèmes d'agents mobiles, d'approfondir cette nouvelle forme d'architecture des réseaux et de faire valoir son impact sur l'existant en matière de communication et de services. Le but de cette section est de présenter, une vue globale sur l'utilisation des agents mobiles dans les systèmes de détection d'intrusions.

1. **Agent mobile**

   Un agent mobile est défini dans [44] comme étant un programme autonome qui peut se déplacer de son propre chef, de machine en machine sur un réseau hétérogène dans le but de détecter et de combattre les intrusions. Cet agent mobile doit être capable de s'adapter à son environnement, de communiquer avec d'autres agents, de se déplacer et de se protéger. Pour ce dernier point, une des fonctions de l'agent doit être l'identification et l'authentification pour donner l'emplacement et l'identité de celui qui l'a lancé.

   Il est également défini dans [28] comme étant «*une entité logicielle qui fonctionne de manière continue et autonome dans un environnement particulier...capable d'effectuer des activités de manière flexible et intelligente qui réagit bien aux changements d'environnement... Idéalement, un agent fonctionnant continuellement..serait capable de s'enrichir de ses expériences. De plus, nous attendons d'un agent qu'il puisse cohabiter avec d'autres dans un même environnement et accomplir ses tâches en communicant et en coopérant avec eux, voir même effectuer cela en se déplaçant sur différents postes*».

   Le principal avantage de l'utilisation des agents mobiles est la mobilité des ces derniers sous forme des entités indépendantes. Ils peuvent être ajoutés ou retirés d'un système sans altérer les autres composants et ils peuvent être aussi testés seuls avant d'être introduits dans des environnements plus complexes. En outre, ils peuvent faire parti d'un groupe et fournir alors des fonctions simples mais qui peuvent être échangées de façon à donner des résultats plus complexes qu'ils ne pourraient pas obtenir seuls. Le

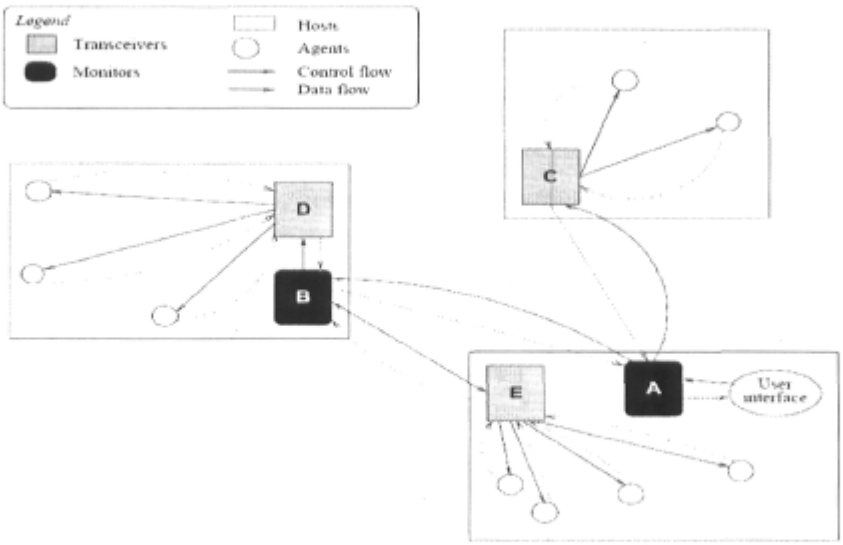


FIG. 2.3 – Couche physique d'un système de détection d'intrusions par des agents mobiles.

fait qu'il n'y ait pas de programme principal qui se sert des autres modules comme esclaves mais plutôt la présence de plusieurs entités intelligentes qui collaborent, fait que si une des entités s'arrête, le système continue son fonctionnement.

2. **Architecture du système avec les agents mobiles**

Cet exemple est donné dans [28] pour mettre en place des systèmes de détection d'intrusions utilisant des agents mobiles pour la collecte des données et leur analyse en employant une structure hiérarchique. Celle-ci permet une grande souplesse par sa faculté à s'adapter à la dimension du réseau.

La figure 2.3 montre un exemple simple d'IDS qui adhère à l'architecture des agents mobiles pour les systèmes de détection d'intrusions. Cette figure montre les trois composantes essentielles de l'architecture : agents, transmetteurs et moniteurs (ou contrôleurs).

Un système de détection d'intrusions basé sur des agents mobiles peut être distribué sur un nombre quelconque de stations au sein d'un réseau. Tous les agents d'une station rapportent les résultats de leurs recherches à un ou plusieurs transmetteurs. Ces transmetteurs surveillent les opérations effectuées par tous les agents. Ils ont la capacité de lancer, d'arrêter et d'envoyer des commandes de configuration à ces agents. Ils peuvent également réaliser de la compression de données à partir des informations reçues par les agents. Finalement, ils rapportent leurs résultats à un ou plusieurs moniteurs. Ces derniers surveillent les opérations de plusieurs transmetteurs à la fois. Ils ont accès aux

données du réseau de manière étendue et de là ils sont capables d'effectuer des corrélations de haut niveau et de détecter des intrusions impliquant plusieurs machines. Ils peuvent être organisés hiérarchiquement de telle sorte qu'ils réfèrent eux-mêmes leurs activités à un contrôleur supérieur. Aussi, un transmetteur peut reporter ses activités à plus d'un contrôleur pour fournir des informations redondantes permettant de résister à une panne provenant de l'un d'entre eux. Pour finir, un contrôleur est responsable de fournir des informations et d'obtenir des commandes de contrôle à partir d'une interface utilisateur.

Pour conclure cette partie, si les agents mobiles possèdent des avantages importants, les inconvénients qu'ils engendrent ne sont pas négligeables. Cependant, l'approche par agents mobiles semble pouvoir donner des résultats meilleurs que les autres technologies et la recherche est entrain de développer une nouvelle architecture pour cette technologie.

2.1.5 Outils de détection d'intrusions

Depuis les années 80, divers outils des systèmes de détection d'intrusions ont été développés. On cite dans ce qui suit les plus importants :

1. **NIDS (Network Based Intrusion Detection Systems)**

   Ce sont les systèmes les plus connus en pratique [19] et ils peuvent être assimilés à un *sniffer* permettant de capturer et de décoder toutes les trames qui transitent par les segments sur lesquels ils sont connectés. Toutefois, contrairement à un sniffer, cette sonde analyse les paquets IP dans leur intégralité afin de repérer des signatures d'attaques déjà connues ou des anomalies dans les entêtes des paquets. Un exemple de ce système est l'outil Snort [23].

2. **HIDS (Host Based Intrusion Detection Systems)**

   Ils représentent le complément naturel des NIDS. Les HIDS[13] utilisent des informations d'audit provenant essentiellement de la machine à protéger à partir de diverses sources (traces d'audit système, historique des commandes exécutées, etc.) pour la détection des comportements malicieux. Plus précisément, ils peuvent offrir les services suivants :

   – Contrer les attaques provenant des applications installées sur le système protégé.

- Vérification de l'intégrité des fichiers sensibles.
- Corrélation des fichiers journaux en provenance d'applications ou d'équipements tiers tels que les routeurs, les pare-feux et les commutateurs.

Un exemple de ce système est l'outil Tripwire [3].

3. **IDES (Intrusion Detection Expert System)**

   Les IDES [38] stipulent que le comportement d'un utilisateur reste presque inchangeable au cours d'un courte période de temps et que la manière dont il se comporte peut être résumée en calculant diverses statistiques sur son comportement. Les IDES construisent les profils des groupes (des utilisateurs censés avoir des comportements proches les unes des autres) et tentent de corréler le comportement actuel d'un utilisateur avec son comportement passé et le comportement passé de son groupe. Un exemple de ce système est l'outil P-BEST [36].

4. **NIDES (Next Generation Expert System)**

   Les NIDES [5] qui sont la continuation des IDES, fonctionnent sur une machine dédiée et indépendante du système surveillé. Le système cible chiffre son audit et il le transmet à la machine de surveillance via le réseau. Le NIDES apprend les habitudes du système cible en étudiant les événements d'accès aux fichiers et aux répertoires, la consommation de ressources, l'activité réseau, les activités des programmes, etc. Il s'appuie sur une approche statistique (modèle de Denning) et une approche qui repose sur les systèmes experts pour tirer ses conclusions. Un exemple de ce système est l'outil LAMBDA [15].

2.2 Pare-feux (Firewalls)

Le but de cette partie est de présenter une autre technique très efficace et plus utilisée avec les systèmes de détection d'intrusions dans le but de renforcer la sécurité des réseaux : il s'agit des pare-feux. Ces derniers sont utilisés pour contrôler, analyser, sécuriser et gérer le trafic dans les réseaux. Cela permet d'utiliser le réseau de la façon pour laquelle il a été prévu et d'empêcher les accès non-autorisés. Un pare-feu contient généralement un ensemble de règles prédéfinies permettant de rejeter ou d'autoriser des connexions et/ou des paquets. De ce fait, la protection d'un réseau local qui utilise des pare-feux (matériels ou logiciels) des accès non autorisés revient à configurer ces derniers d'une manière correcte par rapport aux contraintes de sécurité spécifiées. Plus de détails sur cette technique de sécurisation sont disponibles dans [31].

Règle	**Action**	**IP Src**	**IP Dest**	**prt**	**port Src**	**Port Dst**
1	*Accept*	192.168.10.20	194.154.192.3	tcp	any	25
2	*Accept*	*any*	192.168.10.3	tcp	*any*	80
3	*Accept*	192.168.10.0/24	*any*	tcp	*any*	80
4	*Accept*	*any*	*any*	*any*	*any*	*any*

TAB. 2.1 – Règles de configuration d'un pare-feu

2.2.1 Définition

Les pare-feux (firewalls) sont des dispositifs physiques (matériel) ou logiques (logiciels) conçus pour contrôler le trafic entre le réseau interne et le réseau externe en autorisant uniquement la circulation du trafic qui ne viole pas la politique de sécurité mise en place (voir Fig. 2.4). Ils sont configurés via des règles de filtrages spécifiées par des experts en sécurité des réseaux. Quand un pare-feu reçoit un paquet, il commence par vérifier sa liste de règles de filtrages du début à la fin à la recherche d'une règle permettant d'accepter ou de rejeter le paquet. La règle suivante en est un exemple :

Accept TCP 20.9.17.8 121.11.127.20 any 23

Dans cette règle, le pare-feu est configuré pour accepter n'importe quel paquet du protocole TCP ayant "20.9.17.8" comme adresse IP source et "121.11.127.20" comme adresse IP destination et utilisant n'importe quel port source et 23 comme port destination. Tab. 2.1 donne d'autres exemples de règles de configuration d'un pare-feu.

Les règles du pare-feu permettent de mettre en œuvre un filtrage dépendant de la politique de sécurité adoptée. Les deux approches d'implantation de politiques de sécurité les plus répandues sont :

- Le premier type autorise uniquement les connexions ayant été explicitement autorisées : Tout ce qui n'est pas explicitement autorisé est interdit.
- Le deuxième type empêche les échanges qui ont été explicitement interdits : Tout ce qui n'est pas explicitement interdit est autorisé.

La première méthode est la plus simple, mais elle impose une définition précise des besoins de connexions (quels logiciels, quelles adresses IP, quels ports, quelle direction, etc.).

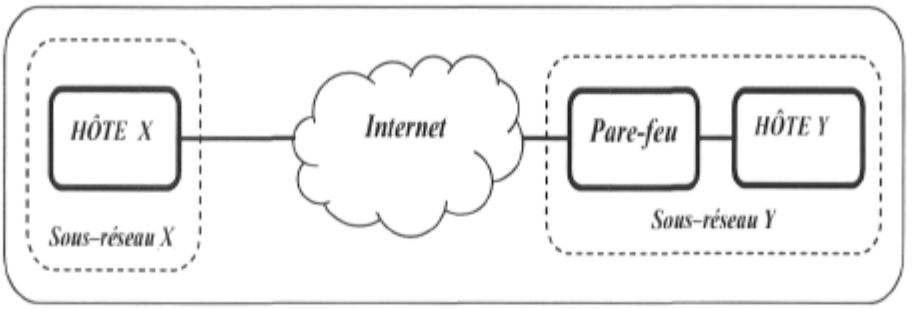


FIG. 2.4 – L'emplacement du pare-feu

2.2.2 Principe de fonctionnement

Essentiellement, il y a trois modes de fonctionnement de pare-feux [29] : le filtrage statique, le filtrage dynamique et le filtrage applicatif.

1. **Filtrage statique (stateless packet filtering)**

   C'est le filtrage de paquets le plus simple et il est une des premières solutions proposées pour les pare-feux et mises en IJuvre sur le marcher. Un pare-feu qui fonctionne selon ce mode de filtrage inspecte les en-têtes de chaque paquet qui le traverse et décide selon la politique de sécurité de le laisser passer ou de le supprimer et ce sans tenir compte des autres paquets. Parmi les champs qui peuvent être analysées et pris en considération lors de la décision d'un pare-feu, nous trouvons :

   - Adresse IP de la machine émettrice.
   - Adresse IP de la machine réceptrice.
   - Type de protocole (TCP, UDP, etc.).
   - Numéro de port : le numéro associé à un service ou une application réseau.
   - Etc.

   Le principal intérêt du filtrage statique réside dans sa simplicité, sa transparence vis-à-vis des utilisateurs, ainsi que la vitesse de traitement des paquets qui le traverse. Cependant, cette solution ne permet de détecter plusieurs attaques (IP Spoofing/ IP Flooding, certain DoS, etc.) qui nécessitent des analyses poussées des paquets en faisant des liens entre eux.

2. **Filtrage dynamique (Stateful Inspection)**

   Cette technique a été proposée pour palier aux certaines limites de pare-feux utilisant le filtrage simple. L'idée est de conserver les traces de sessions et de connexions dans des tables d'états internes aux pare-feux. Ces traces seront également prises en considération par les pare-feux lors de prise de décisions. Ces informations augmentent considérablement les capacités des pare-feux à détecter des attaques sophistiquées. Il reste, cependant, que les failles applicatives (les failles liées aux logiciels), qui sont à l'origine de la plus grande majorité de problèmes de sécurité, ce type de filtrage.

3. **Filtrage applicatif**

   Il a été proposé comme étant une amélioration supplémentaire du filtrage dynamique. Ce mécanisme est attaché au niveau de la couche application, où il peut extraire les données du protocole de niveau 7 pour les étudier. Les requêtes sont traitées par des processus dédiés, par exemple une requête de type *http* sera filtrée par un processus *proxy http*. Le pare-feu rejettera toutes les requêtes qui ne sont pas conformes aux spécifications du protocole et les paquets qui ne respectent pas la politique de sécurité. Cela implique qu'un pare-feu proxy connaît toutes les règles de communication des protocoles qu'il doit filtrer.

2.3 Conclusion

Dans ce premier chapitre, nous avons présenté un bref aperçu de l'état de l'art lié aux principes de fonctionnement de deux principales techniques utilisées pour la sécurisation de réseaux informatiques : les systèmes de détection d'intrusions et les pare-feux. Les IDS se basent sur une approche comportementale ou par scénarios pour repérer et alerter, en temps réel, des tentatives d'attaques. Les pare-feux, quant à eux, jouent un rôle plus actif en essayant d'empêcher les attaques de se produire, et ce en supprimant les paquets indésirable et en utilisant un des modes de filtrage suivants : filtrage simple, filtrage dynamique et filtrage applicatif.

Le chapitre suivant présentera quelques méthodes formelles qui pourront être utilisées pour spécifier et vérifier des réseaux informatiques.

Chapitre 3

Méthodes formelles

Durant les dernières années, la complexité des systèmes informatiques a augmenté d'une manière considérable. La présence des réseaux à haut débit, des machines multiprocesseurs et des applications concurrentes et distribuées ne sont que quelques-unes des rasions qui ont augmentés la complexité de ces systèmes. Par conséquent la tâche de la détection des problèmes dans ces systèmes est devenue de plus en plus coûteuse, subtile et délicate. Par ailleurs, ces composantes informatiques sont aujourd'hui impliquées dans plusieurs systèmes critiques de notre vraie vie (centrale nucléaire, moyens de transport, etc.) et les conséquences des erreurs sont très coûteuses et souvent irréversibles. L'explosion de la fusée Ariane [45] est parmi les exemples les plus récents et plus coûteux qui reflètent bien les conséquences possibles de systèmes mal conçus.

Devant ces faits, l'utilisation de méthodes formelles est devenue incontournable pour la spécification et l'analyse des systèmes informatiques dans le but d'augmenter leurs fiabilités. En effet, ces méthodes ont des fondements mathématiques consistants leurs permettant d'accompagner par des preuves irréfutables les conclusions qu'elles produisent. Principalement, les méthodes formelles sont des langages non ambiguës pour la spécification des systèmes et de leurs propriétés, des techniques permettant la vérification des propriétés de ces systèmes et des outils permettant d'automatiser l'utilisation des ces techniques.

Dans ce chapitre, nous présentons les langages les plus utilisées pour spécifier les systèmes concurrents d'une manière générale et les réseaux informatiques en particulier ainsi que leurs propriétés.

3.1 Algèbre des processus

Une algèbre est définie par un ensemble d'éléments de base ainsi qu'un ensemble d'opérateurs permettant de construire des composantes complexes à partir des éléments basics. Parmi, les algèbres de processus les plus connues dans le domaine de la spécification et la vérification des systèmes concurrents et les réseaux informatiques, nous trouvons : CCS[39] et π-Calculus [42] développées par Milner, CSP[26] proposée par Hoare, ACP [10] introduite par Brookes et Roscoe et le calcul ambient[31] développé par Cardelli et Gordon.

Dans ce qui suit, nous présentons les trois algèbres de processus suivantes : CCS, π-Calcul et le calcul ambiant.

3.1.1 CCS

Le calcul des systèmes communicants (CCS) est un langage mathématique qui nous permet de décrire sans ambiguïté des systèmes concurrents. L'ensemble des éléments de base (actions atomiques) de cette algèbre contient des actions qui expriment l'envoi sur des canaux, notées généralement par a, b, c, etc., des co-actions qui expriment la réception sur des canaux, notées par $\overline{a}$, $\overline{b}$, $\overline{c}$, etc. En plus nous trouvons une action spéciale notée par τ et qui est utilisée souvent pour exprimer l'indéterminisme ou la synchronisation. Quant aux opérateurs de cette algèbre, ils sont les suivants : l'opérateur de préfixage noté par ".", l'opérateur du choix noté par "+", l'opérateur de parallélisme, noté par "|", l'opérateur de renommage noté par "[]" et l'opérateur de restriction noté par "\".

Nous notons par Σ l'ensemble des actions, $\overline{\Sigma}$ l'ensemble des co-actions et $Act = \Sigma \cup \overline{\Sigma} \cup \{\tau\}$. Soit α une action dans Act, la syntaxe de CCS peut être maintenant décrite par la BNF donnée dans Tab 3.1.

D'une manière informelle, la sémantique d'un processus de CCS peut être décrite comme suit :

- **Processus inactif** 0 : C'est le processus le plus élémentaire et il est souvent utilisé pour signaler qu'un système a terminé son évolution.
- **Préfixage** $\alpha.P$: C'est le processus qui exécute l'action α et il se comporte par la suite comme P. Par exemple, le processus $a.0$ est le processus qui exécute l'action a et il termine.
- **Définition** $X = P$: Cette construction syntaxique permet, entre autres, de définir des processus qui ont un comportement infini (via la récursion). Par exemple, $X = a.X$ est

P	:=	0	Processus inactif
		$\alpha.P$	Préfixage.
		$X \stackrel{def}{=} P$	Définition.
		$P_1 + P_2$	Choix.
		$P_1 \mid P_2$	Parallélisme .
		$P[f]$	Renommage.
		$P \backslash L$	Restriction.

TAB. 3.1 – Structure des processus CCS.

le processus qui fait tout le temps a.

- **Choix** $P_1 + P_2$ **:** C'est un processus qui peut se comporter soit comme P_1 soit comme P_2. Ce choix peut être déterministe (contrôler par l'environnement) ou indéterministe. Par exemple, dans le processus $a.P + b.Q$ le choix peut être fixé par l'environnement (un autre processus mis en parallèle avec ce dernier). Par contre le choix est indéterministe dans des processus comme $a.P + a.Q$ ou $a.P + \tau.Q$.
- **Parallélisme** $P_1 | P_2$: Le parallélisme dans CCS est défini via l'entrelacement et la synchronisation. La seule possibilité permettant à deux composantes d'un processus d'évoluer en même temps est via une communication. Autrement, c'est l'entrelacement des actions des composantes parallèles qui câble ce parallélisme. La communication, quant à elle, est bipoints synchrone et le résultat d'une communication est une action invisible τ. Par exemple le processus $a0|b.0$ se comporte comme le processus $a.b.0 + b.a.0$ et le processus $a.0|\overline{a}.0$ se comporte comme le processus $a.\overline{a}.0 + \overline{a}.a.0 + \tau.0$.
- **Renommage** $P[f]$: Via une fonction de renommage f, les actions d'un processus peuvent être renommées pour en définir un nouveau processus. Par exemple, le processus $a.b.0[c/a]$ se comporte comme le processus $c.b.0$. Par ailleurs, une fonction de renommage f doit respecter les conditions suivantes $f(\overline{a}) = \overline{f(a)}$ et $f(\tau) = \tau$.
- **Restriction** $P \backslash L$ **:** Ce processus se comporte comme P à l'exception qu'il ne peut exécuter des actions qui sont dans l'ensemble L. Cet opérateur permet de rendre certain canaux de communication invisibles de l'extérieur. Par exemple, le processus $(a.0|\overline{a}.0) \backslash \{a\}$ se comporte comme $\tau.0$.

D'une manière formelle, la sémantique d'un processus est décrite par un système de transitions étiquetées LTS (Labeled Transition System) qui montre les différentes évolutions possibles (comportements) de ce processus. Un LTS est défini comme suit :

3.1.1. Définition.[LTS] Un système de transitions étiquetées $\mathcal{T} = (Q, I, E, \rightarrow)$ est donné par :

- un ensemble Q d'états,
- un sous-ensemble $I \subseteq Q$ d'états initiaux,
- un ensemble E d'étiquettes, et
- une relation de transition $\rightarrow \subseteq Q \times E \times Q$.

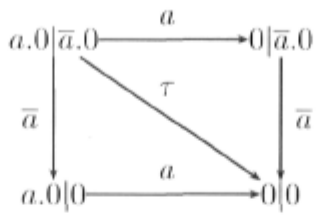
Le système de transition $\mathcal{T}_P$ associé à un processus P peut être construit comme suit :

- Q est le plus petit ensemble d'états qu'on peut atteindre à partir de P en utilisant la relation $\rightarrow$,
- $I = \{P\}$,
- $E = Act$, et
- La relation "$\rightarrow$" est définie par les règles d'inférences données par Tab. 3.2.

Préfixage	$\frac{\square}{\alpha.P \xrightarrow{\alpha} P}$		
Définition	$\frac{P \xrightarrow{\alpha} P'}{X \xrightarrow{\alpha} P'} X \stackrel{def}{=} P.$		
Choix à gauche	$\frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$	Choix à droite	$\frac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} Q'}$
Entralcement$_1$	$\frac{P \xrightarrow{\alpha} P'}{P\|Q \xrightarrow{\alpha} P'}$	Entralcement$_2$	$\frac{Q \xrightarrow{\alpha} Q'}{P\|Q \xrightarrow{\alpha} Q'}$
Communication	$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P\|Q \xrightarrow{\tau} P'\|Q'}.$		
Restriction	$\frac{P \xrightarrow{a} P'}{P\backslash L \xrightarrow{a} P'\backslash L} a \notin L \cup \overline{L}$		
Renommage	$\frac{P \xrightarrow{a} P'}{P[f] \xrightarrow{f(a)} P'[f]}$		

TAB. 3.2 – Sémantique opérationnelle de CCS.

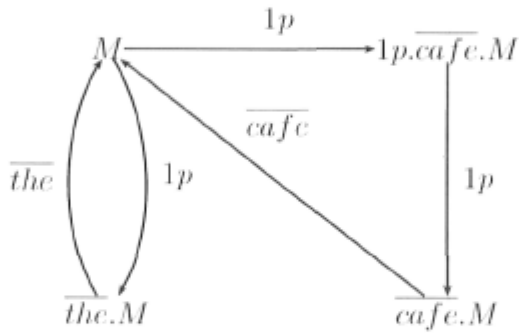
Par exemple le système de transition associé à $a.0|\overline{a}.0$ est :



Le système de transition associée à la machine décrite par le processus suivant :

$$M \stackrel{def}{=} 1p.(\overline{the}.M + 1p.\overline{cafe}.M)$$

est



3.1.2 Equivalence en CCS

Spécifier un système d'une manière non ambiguë est en soit une contribution significative. En effet, ceci permet, entre autres, d'éliminer le malentendu et les mauvaises interprétations qui peuvent survenir dans les spécifications décrites dans les langages naturels. Elle est souvent utilisée comme référence pendant le processus de développement du système pour s'assurer que ce dernier est conforme aux attentes initiales.

Cependant, pour vérifier d'une manière formelle qu'un système est conforme à sa spécification, il est indispensable de développer des techniques d'analyses permettant d'atteindre ces objectifs. Les relations d'équivalence font parti de panoplie d'outils utilisées pour comparer un système avec sa spécification, deux systèmes ensemble, deux spécifications ensemble, etc. Les raisons pour lesquelles cette relation de comparaison est choisie de sorte qu'elle soit une relation d'équivalence sont les suivantes :

- Reflexivité : Il est évident qu'on ne souhaite pas faire sortir de différences quand on compare un système avec une copie identique de lui même, c.à.d : $P \sim P$.
- Symétrie : Si la comparaison entre P et Q ne fait pas sortir de différence alors ont veut qu'il sera le cas quand on compare Q avec P, c.à.d : $(P \sim Q) \Rightarrow (Q \sim P)$.
- Transitivité : Si la comparaison entre P et Q ne fait pas sortir de différences et la comparaison entre Q et R ne fait pas sortir de différences, alors il n'y a pas de raison que la comparaison de P et R fait sortir de différences c.à.d : $((P \sim Q) \wedge (Q \sim R)) \Rightarrow (P \sim R)$.

Il y a d'autres propriétés qu'on souhaite avoir lors de comparaison des processus CCS. Par exemple, on veut que :

$$\begin{array}{lcl} P \mid Q & \sim & Q \mid P \\ P + Q & \sim & Q + P \\ P \mid 0 & \sim & P \\ P + 0 & \sim & P \end{array}$$

Par ailleurs, parmi les propriétés les plus demandées qu'on souhaite trouver dans la relation de comparaison c'est la congruence. Une relation d'équivalence $\sim$ est une congruence par rapport à une fonction n-aire f si pour tout a_1, ..., a_n et pour tout b_1, ..., b_n :

$$((a_1 \sim b_1) \wedge \ldots \wedge (a_n \sim b_n)) \Rightarrow f(a_1, \ldots, a_n) \sim f(b_1, \ldots, b_n)$$

En d'autres termes, si nous avons un système formé de n composantes $a_1, \ldots, a_n$ liées entre elles par une fonction f (le système est $f(a_1, \ldots, a_n)$), alors remplacer une composante a_i par une autre composante b_i équivalente va nous donner toujours un système équivalent, c.à.d :

$$f(a_1, \ldots, a_i, \ldots, a_n) \sim f(a_1, \ldots, b_i, \ldots, a_n)$$

Très utile, cette propriété réduit la tâche de comparaison de deux systèmes à la comparaison de leurs composantes élémentaires. Pour CCS, nous souhaitons manipuler des relations de comparaisons qui ont cette propriété pour tous opérateurs de composition de processus qui sont le préfixe ".", le choix "+", le parallèle "|", le renommage "[]" et la restriction "\" . En d'autres termes, nous voulons que pour tout context $\mathcal{C}[X] \in \{a.X,\ P + X,\ P|X,\ X,\ X[f]\}$ et pout tout processus P_1 et P_2 tels que $P_1 \sim P_2$, nous avons :

$$\mathcal{C}[P_1/X] \sim \mathcal{C}[P_2/X]$$

Dans ce qui suit nous donnons deux relations d'équivalence utilisées avec les algèbres de processus et en particulier avec CCS. La première est la bissimulation forte qui une relation de congruence alors que la deuxième connue sous le nom de bissimulation faible ne l'est pas.

3.1.2. Définition.[Bissimulation forte]

Une relation binaire $R \subseteq \mathcal{P} \times \mathcal{P}$ est une bissimulation forte, si pour tout couple $(P, Q) \in R$, et pour toute action $\alpha \in Act$, nous avons :

1- $P \xrightarrow{\alpha} P'$ alors $\exists Q' \in \mathcal{P}$ tel que $Q \xrightarrow{\alpha} Q'$ et $(P', Q') \in R$.
2- $Q \xrightarrow{\alpha} Q'$ alors $\exists P' \in \mathcal{P}$ tel que $P \xrightarrow{\alpha} P'$ et $(P', Q') \in R$.

D'après la définition précédente, plusieurs relations sont des bissimulations fortes, l'ensemble vide en est une. Nous notons par "$\sim$" la plus grande bissimulation forte, c.à.d :

$$\sim = \bigcup \{R | R \ est \ bisimulation \ forte.\}$$

Le fait de choisir "$\sim$" comme étant la plus grande bissimulation forte simplifie la tâche de prouver que $P \sim Q$ à trouver une bissimulation forte qui contient le couple (P, R). Par exemple, si $P = (a.0 \mid b.0)$ et $Q = (a.b.0 + b.a.0)$, alors $P \sim Q$ puisque la relation suivante est une bissimulation forte :

$$R = \{(a.0 \mid b.0,\ a.b.0 + b.a.0),\ (a.0 \mid 0,\ a.0), (b.0 \mid 0, b.0), (0 \mid 0,\ 0)\}$$

La bissimulation forte "$\sim$" fait trop de discriminations qui peut, dans certaines situations, devenir un inconvénient majeur. Par exemple, les deux processus $a.0$ et $a.\tau.0$ ne sont pas équivalents même si l'action τ reflète l'exécution d'une action interne inobservable de l'extérieur. Pour abstraire les actions invisibles baser la comparaison seulement sur les actions visibles, la notion de bissimulation faible a été introduite comme suit :

3.1.3. Définition.[Bissimulation faible] Une relation binaire $R \subseteq \mathcal{P} \times \mathcal{P}$ est une bisimulation faible, si pour tout couple $(P, Q) \in R$, et toute action $a \in Act$:

1- $P \xrightarrow{a} P'$ alors $\exists Q' \in \mathcal{P}$ tel que $Q \stackrel{\hat{a}}{\Longrightarrow} Q'$ et $(P', Q') \in R$.
2- $Q \xrightarrow{a} Q'$ alors $\exists P' \in \mathcal{P}$ tel que $P \stackrel{\hat{a}}{\Longrightarrow} P'$ et $(P', Q') \in R$.

avec $\hat{a} = a$ si $a \neq \tau$, $\hat{\tau} = \varepsilon$ et $P \stackrel{\hat{a}}{\Longrightarrow} P'$ signifie que

$$P(\xrightarrow{\tau})^* \xrightarrow{a} (\xrightarrow{\tau})^* P'$$

La relation de bissimulation faible, notée par "$\approx$", est définie comme suit :

$$\approx = \bigcup\{R \mid R \; est \; bisimulation \; faible.\}$$

Par exemple si $P = (a.c.0 \mid \overline{c}.b.0)\backslash\{a\}$ et $Q = (a.b.0)$ alors $P \approx Q$ puisque la relation suivante est une bissimulation faible.

$$R = \{(((a.c.0 \mid \overline{c}.b.0)\backslash\{c\},\ a.b.0),\ ((c.0 \mid \overline{c}.b.0)\backslash\{c\},\ b.0), ((0 \mid b.0)\backslash\{c\}, b.0), (0 \mid 0)\backslash\{c\},\ 0)\}$$

On peut vérifier que $\approx$ n'est pas une relation de congruence puisque $\tau.0 \approx 0$ mais $a.0 + \tau.0 \not\approx a.0 + 0$. Cependant nous pouvons trouver la plus grande congruence incluse dans $\approx$ en calculant la plus grande relation contenant les couples (P, Q) qui respectent les conditions suivantes :

1- $P \xrightarrow{a} P'$ alors $\exists Q' \in \mathcal{P}$ tel que $Q \stackrel{a}{\Longrightarrow} Q'$ et $P' \approx Q'$.
2- $Q \xrightarrow{a} Q'$ alors $\exists P' \in \mathcal{P}$ tel que $P \stackrel{a}{\Longrightarrow} P'$ et $P' \approx Q'$.

3.1.3 π-calcul

π-calcul a été introduit par Milner, Parrow et Walker dans [47] comme une extension de CCS, car ce dernier permet de modéliser les systèmes distribués mais ne supporte pas la capacité d'exporter des noms de canaux d'un processus à un autre. Dans CCS par exemple, le processus $\overline{a}.0\backslash a$ ne pourra jamais communiquer sur son canal a puisqu'il est le seul a connaître son existence. Dans le π-calcul, il devient possible de faire connaître des noms de canaux en les transmettant à travers d'autres canaux connus.

Nous présentons dans cette section la syntaxe du π-calcul et sa sémantique opérationnelle. Un exemple d'utilisation de ce langage sera donné par la suite pour facilité sa compréhension.

Syntaxe du π-calcul

Comme le montre Tab. 3.3, la syntaxe du π-calcul est similaire à celle de CCS. Elle ne diffère que par ses actions qui donnent la possibilité d'envoyer et de recevoir des noms de canaux.

Prefixes	$\alpha :=$	$\overline{a}x$	Émettre x sur un canal a
	$\mid$	$a(x)$	Reception x sur un canal a
	$\mid$	τ	Action silencieuse.
Agents	$P :=$	0	Processus inactive,vide
	$\mid$	$\alpha.P$	Préfixage.
	$\mid$	$P_1 + P_2$	Choix.
	$\mid$	$P_1 \mid P_2$	Parallélisme .
	$\mid$	if $x = y$ then P	Égalité.
	$\mid$	if $x \neq y$ then P	Inégalité.
	$\mid$	$(\nu x)P$	Restriction.
	$\mid$	$A(y_1, ..., y_n)$	Identifiant.
Définitions	$A(x_1, ..., x_n)$	$\stackrel{def}{=} P$	Telque $(i \neq j \Rightarrow x_i \neq x_j)$

TAB. 3.3 – Syntaxe du π-calcul.

Les processus de π-calcul peuvent prendre les formes suivantes :

- Processus 0 : Ceci correspond au processus qui n'exécute aucune action (processus vide, inerte).
- Le préfixe $\alpha.P$: Il y a trois préfixes possibles dans ce langage. Le premier $\overline{a}v.P$ correspond à un processus qui émet la valeur v sur le canal a et il se comporte par la suite comme P. Le deuxième $a(x).P$ correspond à un processus qui reçoit une valeur v sur le canal a et se comporte par la suite comme $P\{v/x\}$ (le processus P dans lequel x est remplacée par la valeur reçue v) et le troisième est le préfixe silencieux $\tau.P$ et correspond à un processus qui évolue sans interagir avec l'environnement extérieur. Nous rappelons qu'avec le π-calcul, les données et les canaux peuvent se confondre, ainsi dans les écritures $\overline{a}x$ et $a(x)$, le symbole x peut être un nom de canal, Par exemple, $\overline{a}c$ peut correspondre à l'envoi du nom c, le nom d'un canal, sur le canal a.
- Le choix et le parallélisme : Ces deux opérateurs ont la même signification qu'en CCS.
- Égalité (if $x = y$ then P) : le processus "if $x = y$ then P" se comporte comme P si x et y sont identiques, sinon, il se comporte comme 0.
- Inégalité (if $x \neq y$ then P) : le processus "if $x \neq y$ then P" se comporte comme P si x et y sont différents, sinon, il se comporte comme 0.
- Restriction $(\nu x)P$: il joue un rôle analogue à l'opérateur "\" de CCS. Ainsi, le pro-

cessus $(\nu x)P$ se comporte comme P, mais le nom x est local au P, c'est à dire que P n'utilise pas son nom x pour communiquer avec l'environnement externe.

- Identifiant $A(y_1, ..., y_n)$ où n est l'arité de A : Chaque identifiant a une définition $A(x_1, ..., x_n) \stackrel{def}{=} P$ où les x_i sont distinctes l'une de l'autre. Ansi, $A(y_1, ..., y_n)$ se comporte comme P dans lequel les variables libres x_i sont remplacées par les paramètres y_i. L'avantage principal de cette notation est la possibilité de décrire des processus récursifs. Par exemple un processus qui ne fait qu'émettre indéfiniment la valeur x sur le canal a serait :

$$Envoi(x) \stackrel{def}{=} \overline{a}x.Envoi(x)$$

Sémantique opérationnelle du π-calcul

La sémantique du π-calcul est représentée par Tab. 3.4. On note que la plupart des règles sémantiques de π-calcul ressemblent aux celles de CCS sauf, la règle d'ouverture et les fonctions bn et fn qui apparaissent dans les règles liées au parallélisme et à la restriction.

- Les fonctions $bn(P)$ et $fn(P)$ représentent respectivement les ensembles des noms libres et liés dans le processus P. Par exemple dans le préfixe de reception $a(x).P$, tous les noms de x sont liés dans P contrairement au préfixe d'émission $\overline{a}x.P$, les noms de x sont libres dans P. Plus précisément fn et bn sont définies comme suit :

$$\begin{array}{rcl||rcl} fn(0) & = & \emptyset & bn(0) & = & \emptyset \\ fn(\overline{a}x.P) & = & \{a, x\} \cup fn(P) & bn(\overline{a}x.P) & = & bn(P) \\ fn(a(x).P) & = & \{a\} \cup fn(P) & bn(a(x).P) & = & \{x\} \cup bn(P) \\ fn((\nu x)P) & = & fn(P) & fn((\nu x)P) & = & \{x\} \cup bn(P) \end{array}$$

- La règle d'ouverture : Cette règle n'existe pas dans CCS. Elle permet d'éliminer la restriction faite sur un canal.

Extensions du π-calcul

À travers le temps, π-calcul a subit plusieurs extensions dont en voici quelques unes :

- **π-calcul asynchrone :** le π-calcul synchrone est celui décrit ci-dessus. Dans ce calcul, la communication est bloquante, c'est-à-dire, elle ne peut se faire que s'il y a un proces-

Équivalence	$\dfrac{P' \equiv P,\ P \xrightarrow{\alpha} Q,\ Q \equiv Q'}{P' \xrightarrow{a} Q'}$	
Préfixe	$\dfrac{\square}{a.P \xrightarrow{\alpha} P'}$	
Choix	$\dfrac{P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'}$	
Concordance	$\dfrac{P \xrightarrow{\alpha} P'}{if\ x\ =\ x\ then\ P \xrightarrow{a} P'}.$	
Non concordance	$\dfrac{P \xrightarrow{\alpha} P'}{if\ x \neq y\ then\ P \xrightarrow{a} P'}.$	
Entralcement	$\dfrac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P'}$	$bn(\alpha) \cap fn(Q)\ =\ \emptyset$
Communication	$\dfrac{P \xrightarrow{a(z)} P'\ Q \xrightarrow{\overline{a}u} Q'}{P \mid Q \xrightarrow{\tau} P'\{u/x\} \mid Q'}.$	
Restriction	$\dfrac{P \xrightarrow{a} P'}{(\nu x)P \longrightarrow (\nu x)P'}$	$x \notin bn(\alpha) \cap fn(Q)$
Ouverture	$\dfrac{P \xrightarrow{\overline{a}x} P'}{(\nu x)P \xrightarrow{\overline{a}x} P'}\ a \neq x$	

TAB. 3.4 – Sémantique opérationnelle du π-calcul.

sus qui est prêt à envoyer et un autre qui est prêt à recevoir. Dans la version asynchrone de ce calcul, un processus peut envoyer sans qu'il y ait un autre prêt à recevoir.

- **π-calcul polyadique :** le π-calcul polyadique permet de transmettre un tuple de valeurs lors d'une interaction, c'est à dire, il autorise via une seule et même action la communication de plusieurs noms. Par exemple :

$$P \mid Q \stackrel{def}{=} \overline{a}\langle x, y, z\rangle 0 \mid a(u, v, w)\overline{v}\langle u, w\rangle 0 \xrightarrow{\tau} 0 \mid \overline{y}\langle x, z\rangle 0$$

  Remarquons que la notation pour l'émission en π-calcul polyadique est différente du π-calcul monadique : le tuple est entouré de parenthèses pointues.

- **π-calcul haut niveau :** Des processus peuvent être envoyés par les canaux. La règle de réduction associée à cette évolution est :

$$\overline{c}(R).P|c(v).Q \xrightarrow{\tau} P|Q[R/v]$$

  Le processus $\overline{c}(R).P$ envoie le processus R à $c(v).Q$. On démontre cependant que la capacité d'envoyer des processus n'accroît pas l'expressivité du π-calcul.

3.1.4 Calcul ambient

Il s'agit d'un calcul relativement jeune qui a été introduit par Cardelli et Gordon [31] en 1998. Son but est de fournir un langage formel simple et expressif pour l'étude de la programmation mobile dans les réseaux à large échelle. Sa particularité est qu'il manipule explicitement et d'une manière abstraite la notion de spatialité (espace appelé ambient) dans les processus. Absente dans les autres calculs, cette notion est souvent très utile pour la spécification et la vérification de certains systèmes réactifs et en particulier les réseaux informatiques. Un ambient est un espace (un réseau, une machine, une page web, etc.) ayant un intérieur et un extérieur et il possède un nom unique non modifiable au cours du temps. À l'intérieur d'un ambient, on peut trouver des processus concurrents, comme dans CCS par exemple, ou bien d'autres ambients. Par ailleurs, un ambient peut migrer (notion de mobilité) avec son contenu (déplacer une machine d'un réseau à un autre par exemple) d'un espace à un autre.

La syntaxe du calcul ambient est comme indiquée dans Tab. 3.5.

- La notation $n[P]$ dénote un processus P exécuté dans un ambient nommé n.
- Le processus "0" représente le processus inactif comme dans CCS et π-calcul.
- La restriction de n à P, notée $(\nu n).P$, et la composition parallèle notée par le symbole "|" ont les mêmes significations que celles trouvées dans CCS et π-calcul.

Processus	P	$::=$	$n[P]$	Ambient.
		$\mid$	0	Processus inactive ou vide.
		$\mid$	$(\nu n).P$	Restriction.
		$\mid$	$P\|P'$	Composition Parallèle .
		$\mid$	$C.P$	Processus gardé par la capacité C.
		$\mid$	$\langle M \rangle$	Sortie d'un capacité.
		$\mid$	$\langle\langle N \rangle\rangle$	Sortie d'un nom.
		$\mid$	$(x).P$	Entrée d'un capacité.
		$\mid$	$((u)).P$	Entrée d'un nom.
Capacité	C	$::=$	$in\ N$	Entrer dans N.
		$\mid$	$out\ N$	Sortir de N.
		$\mid$	$open\ N$	Ouvrir N.
		$\mid$	x	Variable.
		$\mid$	ϵ	Chemin vide.
		$\mid$	$M.M'$	Chemin.
Nom	N	$::=$	n	nom.
		$\mid$	u	variable.

TAB. 3.5 – Syntaxe du calcul ambient.

- Le symbole $C.P$ représente un processus qui commence par executer la capacité C puis il continue comme P.
- Les symboles $\langle M \rangle$ et $(x)P$ expriment une communication qui permet de passer une capacité M d'un processus émetteur $\langle M \rangle$ à un processus récepteur $(x)P$ parallèle et voisin (localisé dans le même ambient). Le même principe est également utilisé pour les symboles $\langle\langle N \rangle\rangle$ et $((u))P$: une communication permet de passer un nom N d'un processus émetteur $\langle\langle N \rangle\rangle$ à un processus récepteur $((u))P$ parallèle et localisé dans le même ambient.
- La capacité ε est la capacité vide.
- La capacité $M.M'$ signifie la concaténation de deux capacités M et M'.
- Les capacités $in\ N$, $out\ N$ et $open\ N$: $in\ N$ fait entrer l'ambient conteneur dans un ambient parallèle N, $out\ N$ fait sortir l'ambient conteneur de son ambient parent N, tandis que $open\ N$ élimine les frontières de l'ambient fils N.

La sémantique opérationnelle du calcul ambient est définie via deux relations $\equiv$ et $\rightarrow$ comme les montrent Tab. 3.6 et Tab. 3.7 respectivement.

$P \equiv P$	$P \mid Q \equiv Q \mid P$
$P \equiv Q \Rightarrow Q \equiv P$	$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
$P \equiv Q,\ Q \equiv R \Rightarrow P \equiv R$	$!P \equiv P \mid !P$
$P \equiv Q \Rightarrow (vn)P \equiv (vn)Q$	$(vn)(vm).P \equiv (vm)(vn)P$
$P \equiv Q \Rightarrow P \mid Q \equiv Q \mid R$	$(vn)(P \mid Q) \equiv P \mid (vn)Q \ \ if\ \ n \notin fn(P)$
$P \equiv Q \Rightarrow\ !P \equiv\ !Q$	$(vn)(m[P]) \equiv P \mid m[(vn)P] \ \ if\ \ n \neq m$
$P \equiv Q \Rightarrow n[P] \equiv n[Q]$	$P \mid 0 \equiv P$
$P \equiv Q \Rightarrow M.P \equiv M.Q$	$(v0) \equiv 0$
	$!0 \equiv 0$
$P \equiv Q \Rightarrow M[P] \equiv M[Q]$	$\epsilon.P \equiv P$
$P \equiv Q \Rightarrow (x).P \equiv (x).Q$	$(M.M').P \equiv M.M'.P$

TAB. 3.6 – Définition de $\equiv$.

Les effets des règles "Entrée", "Sortie" et "Ouverture" peuvent être schématisés comme le montre Fig. 3.1.

D'autres exemples qui montrent les rôles des capacités in, out et $open$ sont donnés par

Entrée	$n[in\ m.P \mid Q] \mid m[R] \longrightarrow m[\ n[P \mid Q] \mid R]$
Sortie	$m[\ n[out\ m.P \mid Q] \mid R] \longrightarrow n[P \mid Q] \mid m[R]$
Ouverture	$open\ n.P \mid n[Q] \longrightarrow P \mid Q$
Communication	$\langle M \rangle \mid (x).P \longrightarrow P\{x \longleftarrow M\}$
Restriction	$\dfrac{P \longrightarrow Q}{(vn)P \longrightarrow (vn)Q}$
Parallélisme	$\dfrac{P \longrightarrow Q}{P \mid R \longrightarrow Q \mid R}$
Ambient	$\dfrac{P \longrightarrow Q}{n[P] \longrightarrow n[Q]}$
Équivalence	$\dfrac{P \equiv P',\ P' \longrightarrow Q',\ Q \equiv Q}{P \longrightarrow Q}$

TAB. 3.7 – Sémantique du calcul ambient.

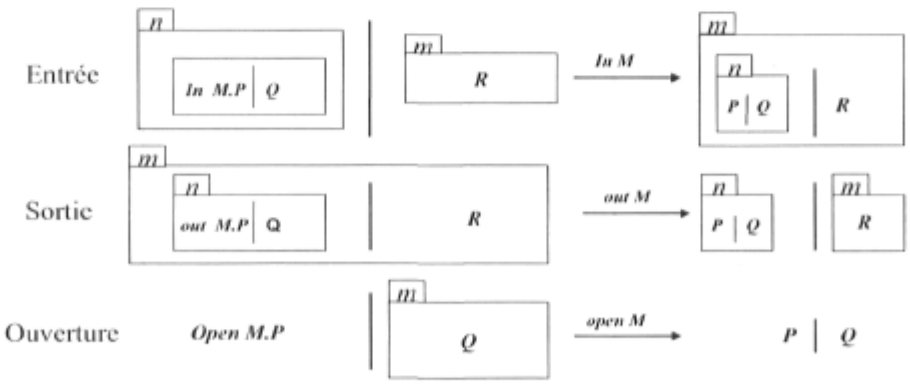


FIG. 3.1 – Rôle des capacités

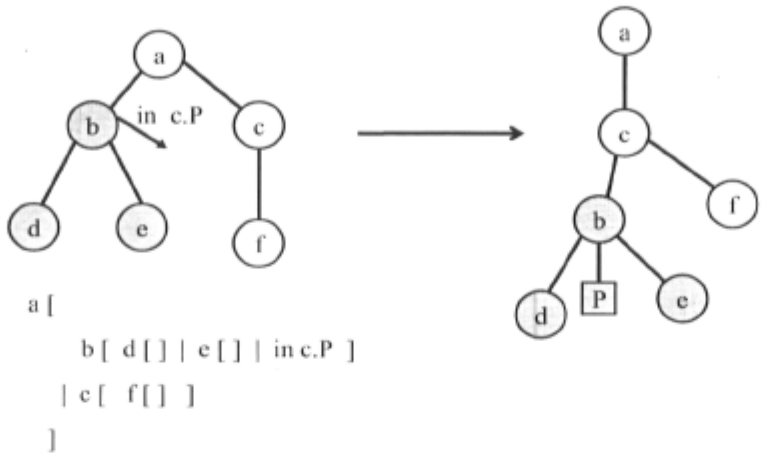


FIG. 3.2 – Rôle de la capacité *in*.

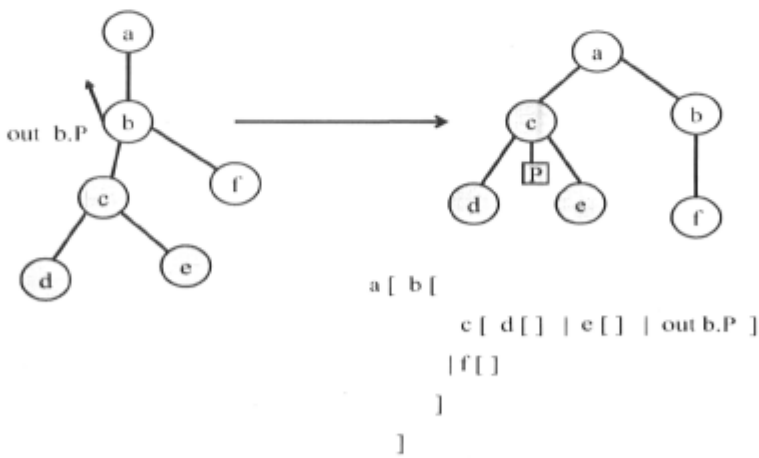


FIG. 3.3 – Rôle de la capacité *out*.

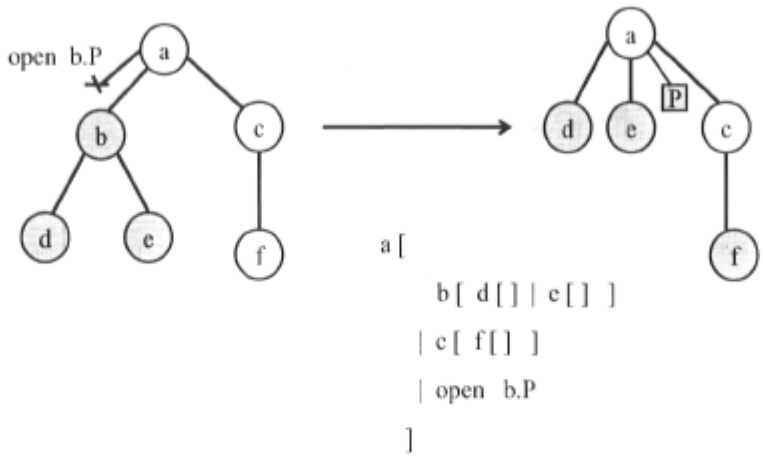


FIG. 3.4 – Rôle de la capacité *open*.

Fig. 3.2, Fig. 3.3 et Fig. 3.4.

- Dans Fig. 3.2, la capacité in permet à l'ambient b qui contient les deux sous ambients d et e d'entrer dans l'ambient c. En d'autres termes, la capacité in permet à l'ambient b de passer d'un état frère à l'état fils pour l'ambient c.
- Dans Fig. 3.3, la capacité out permet à l'ambient c qui contient les deux sous ambients d et e de sortir de l'ambient b. En d'autres termes, la capacité out permet à l'ambient c de passer de l'état fils à l'état frère pour l'ambient b.
- Dans Fig. 3.4, la capacité $open$ permet de briser la barrière entre l'ambient a et b. En d'autres termes, l'ambient b a disparu et ses sous ambients d et e sont attachés directement à l'ambient a.

3.2 Logique temporelle

Dans la section précédente, nous avons présenté les langages formelles les plus utilisés (comme CCS, π-calcul et calcul ambient) permettant de spécifier d'une manière non-ambiguë des systèmes concurrents. Nous avons montré également qu'il est possible, avec CCS par exemple, d'analyser des systèmes en utilisant les relations d'équivalence. Cependant, cette technique nécessite généralement beaucoup de compétence et d'expérience pour pouvoir traduire le système et sa spécification en terme de processus et les comparer. En effet, la spécification d'un système est généralement fournie sous forme d'un ensemble de propriétés de bon fonctionnement que le système doit respecter (propriétés de vivacité) et des propriétés qui indiquent les états ou le système ne doit jamais se trouver (propriétés de sûreté). Par ailleurs,

les logiques sont les langages les plus appropriés pour décrire ces propriétés d'une manière simple, précise et élégante. Une fois la spécification est donnée sous forme de propriétés dans une logique, la vérification consiste généralement à traduire le système préalablement décrite dans un langage approprié vers un modèle (graphe, système de transitions, etc.), sur lequel les propriétés de correction attendues sont vérifiées au moyen d'algorithmes spécifiques. Bien que limitée aux applications ayant un nombre "raisonnable" d'états, cette technique connue sous le nom "vérification par évaluation de modèle" (*Model-Checking*) a bien montrée son utilité et son efficacité pour analyser et détecter des problèmes dans des nombreux systèmes critiques (protocoles cryptographiques, etc.).

La littérature renferme un nombre important de logiques qui varie selon leurs complexités, leurs expressivités ainsi que leurs domaines d'application. Pour l'analyse de systèmes intéractifs, les logiques temporelles sont les mieux appropriées puisqu'elle permette de décrire facilement des propriétés liées à l'ordre de déroulement des événement dans le temps avec des opérateur modaux (*toujours*, *suivant*, *inévitablement*, *jusqu'à*, etc.) dont il est facile à trouver leurs significations en langue naturelle, bien que cette simplicité apparente nécessite quelques fois une grande expertise.

Lors du choix d'une logique temporelle, plusieurs aspects doivent être considérés, parmi lesquels : l'expressivité (la capacité de la logique à exprimer les classes de propriétés intéressantes, telles que la sûreté, la vivacité ou l'équité) ; la complexité d'évaluation (la complexité des algorithmes permettant de vérifier qu'un modèle satisfait une propriété) ; et la facilité d'utilisation (la capacité à exprimer les propriétés de manière simple et naturelle).

Dans ce qui suit, nous présentons LTL (Linear temporal logic) et CTL (Computation tree logic).

3.2.1 Logique temporelle linéaire (LTL)

Comme son nom l'indique, la logique temporelle linéaire [32] nous permet de vérifier des propriétés temporelles sur un modèle linéaire (une séquence d'actions ou d'événements). Sa syntaxe est décrite dans Tab. 3.8 avec p est une proposition atomique faisant partie d'un ensemble pérdéfini AP.

En d'autres termes :

1. Une proposition atomique p est une formule.

$$\phi ::= \quad p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid X\varphi \mid \varphi_1 U \varphi_2$$

TAB. 3.8 – Syntaxe de LTL.

2. Si φ est une formule, alors $\neg\phi$ est une formule.
3. Si φ_1 et φ_2 sont des formules, alors $\varphi_1 \vee \varphi_2$ est une formule.
4. Si φ est une formule, alors $X\varphi$, prononcée (*next* ou suivent) est une formule.
5. Si φ_1 et φ_2 sont des formules, alors $\varphi_1 U \varphi_2$ (prononcée *Until* ou jusqu'à) est une formule.
6. Toute autre chose n'est pas une formule. Cependant, nous pouvons faire appel aux macros dont en voici quelques exemples :

$$\begin{array}{rcl} ff & \equiv & \neg tt \\ \varphi_1 \wedge \varphi_2 & \equiv & \neg(\neg\varphi_1 \vee \neg\varphi_2) \\ \varphi_1 \rightarrow \varphi_2 & \equiv & \neg\varphi_1 \vee \varphi_2 \\ \varphi_1 \leftrightarrow \varphi_2 & \equiv & \varphi_1 \rightarrow \varphi_2 \wedge \varphi_2 \rightarrow \varphi_1 \end{array}$$

La sémantique de la LTL est définie par rapport à une séquence finie ou infinie d'états π. Dans ce qui suit, nous utilisons les notations suivantes :

$$\begin{array}{rcl} \pi & = & s_0.s_1 \ldots s_i \ldots \\ \pi[i] & = & s_i \\ \pi_i & = & s_i.s_{i+1} \ldots \\ S(\pi) & = & \{s_0, s_1, \ldots, s_i, \ldots\} \end{array}$$

Étant donnés un modèle π et une fonction d'interprétation $L : S(\pi) \longrightarrow 2^{AP}$ qui assigne à chaque état s dans $S(\pi)$ les propositions atomiques dans AP qui sont valides dans celui-ci. La sémantique de LTL sera comme indiqué dans Tab. 3.9 avec $\pi \models \varphi$ désigne que π respecte φ.

Pour faciliter la spécification de propriétés intéressantes, comme la vivacité et la sûreté, nous introduisons deux nouveaux macros comme suit :

$\pi \models \mathrm{p}$	ssi $\mathrm{p} \in L(\pi[0])$
$\pi \models \neg\, \varphi$	ssi $(\pi \not\models \varphi)$
$\pi \models \varphi_1 \vee \varphi_2$	ssi $(\pi \models \varphi_1)$ ou $(\pi \models \varphi_2)$
$\pi \models \mathrm{X}\varphi$	ssi $\pi_1 \models \varphi$
$\pi \models \varphi_1 \,\mathrm{U}\, \varphi_2$	ssi $\exists\, \mathrm{j} \geq 0.\ (\pi_j \models \varphi_2$ et $(\forall\, 0 \leq \mathrm{k} \leq \mathrm{j}$, on a $\pi_k \models \varphi_1))$

TAB. 3.9 – Sémantique de LTL.

$$\begin{aligned} \mathrm{F}\varphi &\equiv (\mathrm{tt} \cup \varphi) \\ \mathrm{G}\varphi &\equiv (\neg \mathrm{F} \neg\varphi) \end{aligned}$$

- $\mathrm{F}\varphi$ spécifie que φ sera satisfait *à un moment donnée dans le futur* : utilisé pour spécifier les propriétés de vivacité.
- $\mathrm{G}\varphi$ spécifie que ϕ doit être *toujours* respectée : utilisé pour spécifier les propriétés de sûreté.

3.2.2 Logique temporelle arborescente (CTL)

CTL a été introduite par Clarke, Emerson et Sistla dans [14, 20] comme étant une extension simple et naturelle de LTL permettant de spécifier des propriétés sur un modèle arborescent. De ce fait, mises à part les propriétés de chemin de LTL, des nouvelles propriétés d'états ont été introduites. Étant donné un état, ces nouvelles formules nous donnent la possibilité de dire si on veut vérifier des propriétés sur tous les chemins issus de cet état ($A\varphi$) ou sur un seul chemin ($E\varphi$). Ainsi, la syntaxe de CTL est celle donnée par Tab. 3.10

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid E(X\varphi) \mid A(X\varphi) \mid E(\varphi_1 U \varphi_2) \mid A(\varphi_1 U \varphi_2)$$

TAB. 3.10 – Syntaxe de CTL.

Soit $M = (s_0, S, \rightarrow, L)$ un système de transitions où $L : S \longrightarrow 2^{AP}$ est une fonction qui assigne à chaque état s dans S les propositions atomiques dans AP qui sont valides dans celui-ci. La sémantique de CTL est comme indiqué dans Tab. 3.11 avec $M, s \models \varphi$ signifie que M respecte la formule φ à l'état s. Par ailleurs, nous disons que M respecte φ si $M, s_0 \models \varphi$.

M, s	$\models$	p	ssi $p \in L(s)$
M, s	$\models$	$\neg\varphi$	ssi $M, s \not\models \varphi$
M, s	$\models$	$\varphi_1 \vee \varphi_2$	ssi $M, s \models \varphi_1$ ou $M, s \models \varphi_2$
M, s	$\models$	$EX\varphi$	ssi $\exists s_x \in \{s' \mid s \rightarrow s'\} : M, s_x \models \varphi$
M, s	$\models$	$AX\varphi$	ssi $\forall s_x \in \{s' \mid s \rightarrow s'\} : M, s_x \models \varphi$
M, s	$\models$	$E(\varphi_1 U \varphi_2)$	ssi $\exists \pi = s \rightarrow s_1 \rightarrow \ldots$ un chemin dans M tel que $\pi \models \varphi$
M, s	$\models$	$A(\varphi_1 U \varphi_2)$	ssi $\forall \pi = s \rightarrow s_1 \rightarrow \ldots$ un chemin dans M tel que $\pi \models \varphi$

TAB. 3.11 – Sémantique de CTL.

3.3 Conclusion

Dans ce chapitre, nous avons présenté des exemples de langages formelles, comme CCS, π-calcul et le calcul ambient, permettant de spécifier des systèmes réactifs. Nous avons présenté également certaines techniques (équivalence entre processus et logiques temporelles comme LTL et CTL) pour la spécification et la vérifications des propriétés de ces systèmes.

Dans le prochain chapitre, nous allons nous inspirer de ces travaux pour définir un langage mieux approprié pour la spécification des réseaux informatique. Nous allons également élaborer des techniques permettant de configurer automatiquement des réseaux de sorte qu'ils se comportent conformément à une politique de sécurité donnée.

Chapitre 4

Approche algèbrique pour la sécurité des réseaux

L'objectif du présent chapitre est de développer une technique formelle permettant de sécuriser un réseau informatique. Pour ce faire, nous avons besoin d'un langage formel pour la spécification des politiques de sécurité (une logique), et d'un langage formel permettant la specification des réseaux (une algèbre de processus). En ce qui a trait au renforcement de la politique de sécurité, nous présentons une technique qui permet d'insérer des moniteurs à l'intérieur des composantes d'un réseau. Ainsi, nous produisons une nouvelle version sécuritaire du réseau en question. Enfin, nous développons une technique de monitorage sélectif qui permet à l'utilisateur de choisir les composantes réseau dans lesquelles il veut insérer son moniteur.

Dans ce qui suit, nous débutons par présenter la problématique. Ensuite, nous présentons une algèbre de processus (CMN) pour la modélisation des réseaux informatiques en portant une attention particulière à son opérateur de surveillance qui permet de monitorer un processus. Puis, nous proposons une logique pour la spécification des politiques de sécurité. Ensuite nous continuons par définir notre opérateur de renforcement et nous montrerons son utilisation pour générer des réseaux sécurisés. Par la suite, nous définissons une relation d'équivalence basée sur la bissimulation faible et nous donnons quelques résultats sur la correction de notre technique de surveillance. Pour terminer, nous présentons une extension de notre travail, il s'agit d'une technique de monitorage sélectif qui donne la possibilité à l'usager de sélectionner les composantes dans lesquelles il veut placer le moniteur.

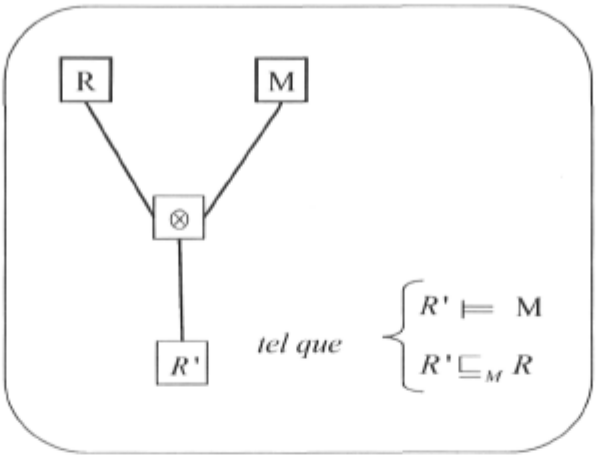


FIG. 4.1 – Opérateur de renforcement.

4.1 Problématique

D'une matière intuitive, à partir d'un réseau R et d'une politique de sécurité M, nous voulons définir un opérateur $\otimes$ qui permet de générer une nouvelle version R' de R (voir la figure 4.1). Cette nouvelle version du réseau R' doit satisfaire les deux propriétés suivantes :

1. $R' \models M$: R' doit se comporter d'une manière qui ne viole pas la politique de sécurité M. D'une manière plus précise, étant donné un système de transitions étiquetés R' (qui modélise le comportement du réseau), on dit que R' satisfait M, noté par $R' \models M$, si pour tout $(S, a, S') \in R'$ on a $a \models M$;
2. $R' \sqsubseteq_M R$: dans la nouvelle version du réseau R' toutes les actions qui ne satisfont pas le moniteur M seront bloquées. Une définition plus précise sera donnée plus loin dans ce chapitre.

4.2 Langage de modélisation des réseaux

4.2.1 Syntaxe

La syntaxe de l'algèbre proposée, notée CMN « Calculus for Monitored Network », englobe celle de CCS en ajoutant un nouvel opérateur de surveillance sur un processus. Cette algèbre nous permet de décrire le comportement des processus réseaux où les actions de ces processus sont exprimées par des envois de message ou des communications. La syntaxe est

Actions	$\alpha :=$	a	Envoi sur un canal a.
		$\overline{a}$	En provenance du canal a.
		$\vec{a}$	Message en circulation.
		τ	Action silencieuse.
Processus	$P :=$	0	Processus vide.
		$\alpha.P$	Préfixage.
		$X \stackrel{def}{=} P$	Définition.
		$\sum_{i \in I} P_i$	Choix.
		$P_1 \| P_2$	Parallélisme 1 .
		$\vec{a} \| P$	Parallélisme 2 .
		$P[f]$	Renommage.
		$P \backslash L$	Restriction.
		$[P]_M^N$	Surveillance.

TAB. 4.1 – Syntaxe de l'algèbre CMN.

illustrée dans le tableau 4.1. Les actions de l'algèbre sont énumérées comme suit :

- a : est une action qui représente un envoi d'un message sur un canal a.
- $\bar{a}$: est une action complémentaire qui représente un message en provenance du canal a.
- τ : est une action silencieuse.

L'opérateur de surveillance dénoté par $[\,]_M^N$ crée une sorte de boîte dans laquelle opère un processus. $[P]_M^N$ signifie que le processus P, qui s'exécute dans l'emplacement N, est contrôlé par le moniteur M. Cet opérateur, permet de contrôler toutes les actions externes de P. Notons que N désigne le nom de l'emplacement dans lequel s'exécute le processus P. Dans le reste du présent chapitre, nous représenterons N par l'ensemble des canaux de communication externes de P. En outre, nous désignerons par *I* et *O* les fonctions qui, pour un processus donné, retournent respectivement l'ensemble des canaux d'entrées et l'ensemble des canaux de sorties.

Notons que, contrairement à l'opérateur de restriction qui empêche un processus d'envoyer ou de recevoir sur une liste de canaux L en le bloquant, l'opérateur de surveillance ne

bloque pas le processus, plutôt il intercepte les messages envoyés.

4.2.2 Sémantique opérationnelle

Pour des raisons de clarté, nous allons séparer, dans ce qui suit, la sémantique de l'opérateur de surveillance de la sémantique des autres constructions.

1. **Sémantique opérationnelle des opérateurs de base**

   Le tableau 4.2 illustre la sémantique opérationnelle des opérateurs de base de l'algèbre CMN. La sémantique est presque la même que celle des opérateurs correspondant de CCS.

   Toutefois, nous avons modifié la sémantique de l'opérateur de préfixage et celui de communication. Pour être plus proche de ce qui se passe réellement dans un réseau informatique, nous avons modifié l'opérateur de préfixage. En effet, l'envoi est une action asynchrone tandis que la réception est synchrone. Nous supposons qu'un processus réseau peut avancer en exécutant une action d'envoi, et que $\vec{a}$ représente un message en circulation dans un réseau.
   Dans CCS, la communication est représentée par une action silencieuse. Pour notre étude nous avons besoin de garder trace de tous les échanges entre les composantes réseau afin de les contrôlées. C'est la raison principale pour laquelle nous avons introduit une nouvelle action, notée $c(a)$, qui est une action de communication qui représente une communication à travers le canal a.

   Par ailleurs, afin de réduire le nombre de règles de la sémantique opérationnelle, nous utilisons la relation $\equiv$ définie par les axiomes donnés par le tableau 4.3.

2. **Sémantique opérationnelle de l'opérateur de surveillance**

   Le tableau 4.4 montre la sémantique opérationnelle de l'opérateur de surveillance. Il y a sept règles qui montrent l'evolution des différentes formes de processus monitorés. Nous détaillons donc le rôle de chacune de ces règles dans les étapes suivantes :

   - $\mathbf{R}^1_{[\,]}$: cette règle exprime le fait que le moniteur n'a aucun contrôle sur les communications internes d'un processus.

$(R_{\equiv})\dfrac{P \equiv P' \quad P' \xrightarrow{\alpha} Q \quad Q \equiv Q'}{P \xrightarrow{\alpha} Q'}$		Équivalence
$(R_{.})\dfrac{\square}{a.P \xrightarrow{a} \vec{a} \parallel P}$		Préfixage
$(R_{=})\dfrac{P \xrightarrow{\alpha} P' \quad Q \overset{def}{=} P}{Q \xrightarrow{\alpha} P'}$		Définition
$(R_{+})\dfrac{P_i \xrightarrow{\alpha} P'}{\sum_{i \in I} P_i \xrightarrow{\alpha} P'}$		Choix
$(R_{\|})\dfrac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q}$		Entrelacement
$(R_{\|})\dfrac{\square}{\vec{a} \parallel \overline{a}.P \xrightarrow{c(a)} P}$		Communication
$(R_{\backslash})\dfrac{P \xrightarrow{a} P'}{P \backslash L \xrightarrow{a} P' \backslash L}$	$Si \quad a \notin L \cup \overline{L}$	Restriction
$(R_{[]})\dfrac{P \xrightarrow{a} P'}{P[f] \xrightarrow{f(a)} P'[f]}$		Renommage

TAB. 4.2 – Sémantique opérationnelle des opérateurs de base de CMN.

$P + Q$	$\equiv$	$Q + P$	(1)
$P + 0$	$\equiv$	P	(2)
$P \parallel Q$	$\equiv$	$Q \parallel P$	(3)
$P \parallel 0$	$\equiv$	P	(4)

TAB. 4.3 – Axiomes de l'algèbre CMN

- $\mathbf{R}^2_{[\,]}$: dans le même ordre d'idées que la règle précédente, celle-ci signifie que le moniteur n'a aucun contrôle sur les actions silencieuses. Par exemple, si un processus P peut exécuter une action silencieuse τ et devenir le processus P', alors le processus monitoré $[P]^N_M$ peut avancer en exécutant la même action silencieuse τ et devenir le processus monitoré $[P']^N_M$.

- $\mathbf{R}^3_{[\,]}$: cette règle exprime le fait que si un processus peut avancer en exécutant une action d'envoi a, alors le processus monitoré avance en exécutant la même action. Notons, que le paquet qui circule dans le réseau ($\vec{a}$) n'a pas quitté la zone contrôlé par le moniteur M.

- $\mathbf{R}^4_{[\,]}$ et $\mathbf{R}^5_{[\,]}$: ces deux règles permettent de contrôler les sorties externes d'un processus. La première exprime le fait qu'un message en circulation dans le réseau $\vec{a}$, peut quitter la zone contrôlé par un moniteur M si et seulement si ceci est autorisé par M. Par exemple, le processus monitorié $[\vec{a} \| P]^N_M$ peut executer l'action τ et devenir le processus $\vec{a} \| [P]^N_M$ si et seulement si les conditions suivantes ($a \in O(N)$ et $a \models M$) sont satisfaites.

  Contrairement à la règle $\mathbf{R}^4_{[\,]}$, la règle $\mathbf{R}^5_{[\,]}$ montre comment le moniteur intercepte tout paquet qui viole la politique de sécurité.

- $\mathbf{R}^6_{[\,]}$ et $\mathbf{R}^7_{[\,]}$: ces deux règles permettent de contrôler les entrées externes d'un processus de la même manière que celles des sorties externes.

3. **Exemple de modélisation de réseau**

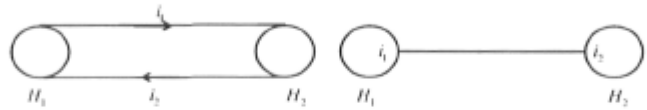
Dans ce qui suit, nous présentons un exemple de réseau exprimé en CMN. Dans cet exemple, nous montrons comment à partir d'une architecture réseau nous trouvons le processus réseau correspondant. La figure 4.3 montre un réseau R_1 formé de quatre machines H_1, H_2,H_3 et H_4.

Notons qu'afin de simplifier la définition des processus, nous utilisons l'abréviation suivante pour le passage de valeur :

$R^1_{[\,]}$:	$\dfrac{P \xrightarrow{c(a)} P'}{[P]^N_M \xrightarrow{c(a)} [P']^N_M}$	Communication
$R^2_{[\,]}$:	$\dfrac{P \xrightarrow{\tau} P'}{[P]^N_M \xrightarrow{\tau} [P']^N_M}$	Action silencieuse
$R^3_{[\,]}$:	$\dfrac{P \xrightarrow{a} \vec{a}\|P'}{[P]^N_M \xrightarrow{a} [\vec{a}\|P']^N_M}\ a \in \mathcal{A}$	Envoie
$R^4_{[\,]}$:	$\dfrac{\square}{[\vec{a}\|P]^N_M \xrightarrow{\tau} \vec{a}\|[P]^N_M}\ a \in O(N), a \models M$	Sortie externe
$R^5_{[\,]}$:	$\dfrac{\square}{[\vec{a}\|P]^N_M \xrightarrow{\tau} [P]^N_M}\ a \in O(N), a \not\models M$	Sortie bloquée
$R^6_{[\,]}$	$\dfrac{\square}{\vec{a}\|[P]^N_M \xrightarrow{\tau} [\vec{a}\|P]^N_M}\ a \in I(N), a \models M$	Entrée
$R^7_{[\,]}$:	$\dfrac{\square}{\vec{a}\|[P]^N_M \xrightarrow{\tau} [P]^N_M}\ a \in I(N), a \not\models M$	Entrée bloquée

TAB. 4.4 – La sémantique opérationnelle de l'opérateur de surveillance.

$a.P$ désigne $\sum_{m\in\mathcal{M}} a(m).P$. En d'autres mots, $a.P$ est un processus qui peut envoyer n'importe quel message sur le canal a. Notons que, $\mathcal{M}$ dénote l'ensemble des messages. De même pour la réception, $\bar{a}.P$ désigne $\bar{a}(x).P$ où le nom de la variable x n'est pas important (on peut recevoir n'importe quoi).
En outre, la notation $[P]^N$ désigne $[P]^N_M$ où M est un moniteur qui accepte n'importe quel message.
De plus, pour la représentation de l'architecture de réseaux nous utilisons la notation de la figure 4.2(b). En effet, étant données deux composantes réseaux H_1 ayant l'interface i_1 et $H2$ ayant l'interface i_2, H_1 peut envoyer des messages par l'interface i_1 et recevoir des messages par l'interface $\bar{i_2}$ (voir figure 4.2(a)).



(a) Représentation Originale (b) Abréviation

FIG. 4.2 – Architecture de réseaux.

Le réseau de la figure 4.3 est représenté par le processus suivant :

$$R_1 \stackrel{def}{=} [H_1]^{N_1} \| [H_2]^{N_2} \| [H_3]^{N_3} \| [H_4]^{N_4}$$

où :

$$H_1 \stackrel{def}{=} (\sum_{m\in\mathcal{M}} i_1(m).H_1) + \overline{i_2}(x).H_1 + (\sum_{m\in\mathcal{M}} i_5(m).H_1) + \overline{i_8}(y).H_1$$
$$H_2 \stackrel{def}{=} (\sum_{m\in\mathcal{M}} i_2(m).H_2) + \overline{i_1}(x).H_2 + (\sum_{m\in\mathcal{M}} i_6(m).H_2) + \overline{i_3}(y).H_2$$
$$H_3 \stackrel{def}{=} (\sum_{m\in\mathcal{M}} i_3(m).H_3) + \overline{i_6}(x).H_3 + (\sum_{m\in\mathcal{M}} i_7(m).H_3) + \overline{i_4}(y).H_3$$
$$H_4 \stackrel{def}{=} (\sum_{m\in\mathcal{M}} i_4(m).H_4) + \overline{i_7}(x).H_4 + (\sum_{m\in\mathcal{M}} i_8(m).H_4) + \overline{i_5}(y).H_4$$

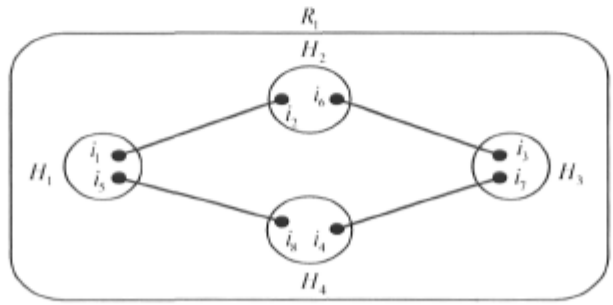


FIG. 4.3 – Une architecture d'un simple réseau.

et

$$I(N_1) = \{\overline{i_2}, \overline{i_8}\} \qquad O(N_1) = \{i_1, i_5\}$$
$$I(N_2) = \{\overline{i_1}, \overline{i_3}\} \qquad O(N_2) = \{i_2, i_6\}$$
$$I(N_3) = \{\overline{i_4}, \overline{i_6}\} \qquad O(N_3) = \{i_3, i_7\}$$
$$I(N_4) = \{\overline{i_5}, \overline{i_7}\} \qquad O(N_4) = \{i_4, i_8\}$$

Par exemple, nous pouvons lire H_1 comme suit : c'est un processus qui peut envoyer des messages à travers les interfaces i_1 et i_5 et se comporter de nouveau comme H_1 (récursivement) . De plus, il peut recevoir des message en provenance des interfaces i_2 et i_8 et se comporter de nouveau comme H_1

4.3 Politique de sécurité

L'objectif de cette section est de définir une logique propositionnelle simple, notée L_M, permettant la spécification de politiques de sécurité. Un moniteur n'est autre qu'une formule dans L_M.

4.3.1 Syntaxe

Moniteur	M	::=	$\top \mid p \mid a(p) \mid \bar{a}(p) \mid \neg M \mid M_1 \wedge M_2$
	p	::=	$Champ\ \ Op\ \ Val$
	$Champ$	::=	$ip\_src \mid ip\_dst \mid port\_src \mid port\_dst \mid prot$
	Op	::=	$= \mid < \mid \in$
	Val	::=	$Num \mid \texttt{TCP} \mid \texttt{UDP} \mid \texttt{ICMP} \mid Num.Num.Num.Num \mid *$
	a	::=	nom d'un canal

TAB. 4.5 – Syntaxe de L_M.

La syntaxe de la logique L_M est présentée dans le tableau 4.5. À noter que les valeurs possibles de $Champ$, $Operateur$ et $Valeur$ sont données à titre d'exemples et que la logique peut être facilement enrichie par d'autres possibilités. Par ailleurs, le symbole $*$ est utilisé pour désigner une valeur quelconque. Dans ce qui suit, nous expliquons les différentes constructions syntaxique de L_M.

- $\top$: est utilisé pour exprimer la valeur booléenne « true ».
- p : est une proposition qui représente une condition de la forme $\langle Champ\ Op\ Val \rangle$ (champ opérateur valeur). Par exemple, la proposition $ip\_src = 132.203.114.23$ désigne que l'adresse source est $132.203.114.23$.
- $a(p)$ (respectivement $\bar{a}(p)$) : est une formule qui exprime le fait que tous les messages envoyés (respectivement reçu) sur l'interface a doivent satisfaire p. Par exemple, la formule "$\neg(i_1(ip\_dst = 132.203.114.23) \wedge i_1(port\_dst = 25))$" exprime le fait que l'interface i_1 ne peut pas envoyer des messages ayant l'adresse destination $132.203.114.23$ et le port destination 25.
- $\neg M$: est un moniteur qui représente une interdiction.
- $M_1 \wedge M_2$: est un moniteur qui représente la conjonction de deux moniteurs M_1 et M_2.

En plus des formules données par la syntaxe de L_M, nous pouvons utiliser d'autres abréviations comme :

- $\bot = \neg\top$
- $M_1 \vee M_2 = \neg(\neg M_1 \wedge \neg M_2)$
- $a(p_1 \wedge p_2) = a(p_1) \wedge a(p_2)$

4.3.2 Sémantique

La sémantique du langage L_M est illustrée dans le tableau 4.6. Nous écrivons $a(m) \models M$ pour dire qu'un message m envoyé sur une interface a respecte M et par $\bar{a}(m) \models M$ pour dire que le message m en provenance de a respecte M.

Voici quelques exemples de moniteurs exprimés à l'aide de L_M.

- "$\neg(ip\_dst = 132.203.114.*)$" : Cette politique de sécurité dit que les messages envoyés par toutes les interfaces du réseau ne doivent pas avoir une adresse destination dans $132.203.114.*$.
- "$\neg i(port\_src = 80)$" : désigne le moniteur qui intercepte, au niveau de l'interface i, tout message transmis sur le port 80.

Finalement, on dit qu'un processus P respecte la politique de sécurité M, noté par $P \models M$ si pour toute action $\alpha : P \xrightarrow{\alpha} P' \Longrightarrow \alpha \models M \wedge P' \models M$.

$a(m) \models \top \quad \overline{a}(m) \models \top$		$\tau \models M$
$a(m) \models Champ\ Op\ Val$	si	$m.Champ\ Op\ Val = true$
$\overline{a}(m) \models Champ\ Op\ Val$	si	$m.Champ\ Op\ Val = true$
$a(m) \models i(p)$	si	$a = i$ et $m.Champ\ Op\ Val = true$
$\overline{a}(m) \models \overline{i}(p)$	si	$a = i$ et $m.Champ\ Op\ Val = true$
$a(m) \models \neg M$	si	$a(m) \not\models M$
$\overline{a}(m) \models \neg M$	si	$\overline{a}(m) \not\models M$
$a(m) \models M_1 \wedge M_2$	si	$a(m) \models M_1$ et $a(m) \models M_2$
$\overline{a}(m) \models M_1 \wedge M_2$	si	$\overline{a}(m) \models M_1$ et $\overline{a}(m) \models M_2$
$c(a(m)) \models M$	si	$a(m) \models M$ et $\overline{a}(m) \models M$

TAB. 4.6 – Sémantique de L_M.

4.4 Opérateur de renforcement ($\otimes$)

Rappelons que notre principal objectif est de définir un opérateur de renforcement $\otimes$ permettant, à partir d'un réseau R et d'une politique de sécurité M, de générer une nouvelle version configurée du réseau $R \otimes M$ telles que :

(i) $R \otimes M \models M$

(ii) $R \otimes M \sqsubseteq_M R$

Dans la section précédente, nous avons déjà défini la relation de satisfaction $\models$, ci-dessous nous présentons la définition formelle de $\sqsubseteq_M$.

4.4.1. Définition. Soient $P, Q \in \mathcal{P}$ deux processus et M un moniteur, on dit que P est inférieur à Q par rapport à M, notée $P \sqsubseteq_M Q$, si et seulement si :

- $P \xrightarrow{a} P'$ alors $Q \xrightarrow{a} Q'$ et $P' \sqsubseteq_M Q'$
- $Q \xrightarrow{a} Q'$ et $a \neq c(a)$ ou bien $Q \xrightarrow{c(a)} Q'$ et $(a \models M \wedge \bar{a} \models M)$ alors $P \xrightarrow{a} P'$ et $P' \sqsubseteq_M Q'$
- $Q \xrightarrow{c(a)} Q'$ et $(a \not\models M \vee \bar{a} \not\models M)$ alors $P \overset{c(a)}{\not\longrightarrow} P'$

Le problème que nous essayons de résoudre admet plusieurs solutions possibles. En d'autre mots, nous pouvons trouver plusieurs opérateurs de renforcement qui satisfont tous les propriétés (i) et (ii). En effet, une solution évidente au problème consiste à placer le moniteur M sur chaque composante du réseau R. Ce qui signifie que l'opérateur de renforcement suivant $\otimes$ satisfait les conditions (i) et (ii) :

$$
\begin{aligned}
[P]^{N}_{M_0} \otimes M &= [P]^{N}_{M_0 \wedge M} \\
([P_1]^{N_1}_{M_1} \| \ldots \| [P_1]^{N_n}_{M_n}) \otimes M &= [P_1]^{N_1}_{M_1 \wedge M} \| \ldots \| [P_1]^{N_n}_{M_n \wedge M} \\
(\vec{a_1} \| \ldots \| \vec{a_m} \| [P_1]^{N_1}_{M_1} \| \ldots \| [P_1]^{N_n}_{M_n}) \otimes M &= \vec{a_1} \| \ldots \| \vec{a_m} \| [P_1]^{N_1}_{M_1 \wedge M} \| \ldots \| [P_1]^{N_n}_{M_n \wedge M}
\end{aligned}
$$

avec : $\overrightarrow{R} = \vec{a}_1(m_1) \| \ldots \| \vec{a}_n(m_n)$.

4.4.1 Exemple

Considérons de nouveau l'exemple de la figure 4.3. Nous avons déjà trouvé que ce réseau peut être spécifié en CMN par le processus suivant :

$$R_1 \stackrel{def}{=} [H_1]^{N_1} \| [H_2]^{N_2} \| [H_3]^{N_3} \| [H_4]^{N_4}$$

Étant donnée la politique de sécurité suivante :

- H_1 ne peut pas envoyer à la machine H_2 sur le port 25 par l'interface i_1.
- H_2 ne peut pas recevoir de la machine H_3 par l'interface i_6.
- Les paquets partant de H_3 sur le port $= 80$ sont interceptés par le moniteur.

Cette politique de sécurité, dénotée par M, peut être spécifiée en L_M par :

- $M = M_1 \wedge M_2 \wedge M_3$:

avec :

- $M_1 = \neg i_1(ip\_src = ip\_src(H_1) \wedge ip\_dst = ip\_dst(H_2) \wedge port\_src = 25)$.
- $M_2 = \neg i_6(ip\_src = ip\_src(H_3) \wedge ip\_dst = ip\_dst(H_2)$.
- $M_3 = \neg(ip\_src = ip\_src(H_3) \wedge port\_src = 80)$.

Maintenant, après application de notre opérateur de renforcement nous obtenons le résultat suivant :

$$R_1 \otimes M \quad = \quad [H_1]_M^{N_1} \| [H_2]_M^{N_2} \| [H_3]_M^{N_3} \| [H_4]_M^{N_4}$$

4.5 Correction de l'opérateur de renforcement $\otimes$

Afin de faciliter la preuve de correction de notre opérateur de renforcement, nous proposons une technique en deux étapes :

- Nous allons étendre notre algèbre CMN en $\mathsf{CMN}_{[\,]}$ en ajoutant un nouvel opérateur de surveillance virtuel. Dans ce nouveau langage, le renforcement d'une politique de sécurité est directe et triviale. Cependant, ce langage manipule des moniteurs qui ne sont pas réaliste (dans le sens où ces moniteurs n'ont pas d'équivalent dans le monde réel). En effet, comme le montre la sémantique, ce nouveau moniteur peut bloquer même les messages qui ne le traversent pas (communication interne).

- Afin de pouvoir revenir à la version originale, nous allons définir une fonction qui permet de transformer la forme virtuelle en une forme réelle.

Il est à noter que la forme virtuelle de processus est ajoutée pour des fins de simplicité. En effet, nous allons prouver que les deux algèbres CMN et $\mathsf{CMN}_{[\,]}$ ont la même expressivité.

4.5.1 Sémantique de l'opérateur de surveillance virtuel

Le tableau 4.7 montre la sémantique opérationnelle de l'opérateur de surveillance virtuel. La sémantique des règles ($\mathrm{R}^2_{[\,]}$, $\mathrm{R}^3_{[\,]}$, $\mathrm{R}^4_{[\,]}$, $\mathrm{R}^6_{[\,]}$ et $\mathrm{R}^7_{[\,]}$) est exactement la même que celle de l'opérateur de surveillance réel. Cependant, la sémantique de l'opérateur de communication diffère de celle de l'opérateur de surveillance réel du fait que le nouvel opérateur contrôle les communications internes d'un processus. Nous avons modifié seulement les conditions des règles ($\mathrm{R}^1_{[\,]}$ et $\mathrm{R}^5_{[\,]}$)

- $\mathbf{R}^1_{[\,]}$: cette règle affirme que le processus P ne peut faire la communication interne à l'intérieur du domaine contrôlé qu'avec l'autorisation du moniteur M ($a \models M \wedge \bar{a} \models M$).

$R^1_{[\,]}$:	$\dfrac{P \xrightarrow{c(a)} P'}{[P]^N_M \xrightarrow{c(a)} [P']^N_M}\ a \models M \wedge \bar{a} \models M$	Communication
$R^2_{[\,]}$:	$\dfrac{P \xrightarrow{\tau} P'}{[P]^N_M \xrightarrow{\tau} [P']^N_M}$	Action silencieuse
$R^3_{[\,]}$:	$\dfrac{P \xrightarrow{a} \vec{a}\|P'}{[P]^N_M \xrightarrow{a} [\vec{a}\|P']^N_M}\ a \in \mathcal{A}$	Envoie
$R^4_{[\,]}$:	$\dfrac{\square}{[\vec{a}\|P]^N_M \xrightarrow{\tau} \vec{a}\|[P]^N_M}\ a \in O(N), a \models M$	Sortie externe
$R^5_{[\,]}$:	$\dfrac{\square}{[\vec{a}\|P]^N_M \xrightarrow{\tau} [P]^N_M}\ a \not\models M$	Sortie bloquée
$R^6_{[\,]}$	$\dfrac{\square}{\vec{a}\|[P]^N_M \xrightarrow{\tau} [\vec{a}\|P]^N_M}\ a \in I(N), a \models M$	Entrée
$R^7_{[\,]}$:	$\dfrac{\square}{\vec{a}\|[P]^N_M \xrightarrow{\tau} [P]^N_M}\ a \in I(N), a \not\models M$	Entrée bloquée

TAB. 4.7 – La sémantique opérationnelle de l'opérateur de surveillance virtuel.

- $\mathbf{R}^5_{[\,]}$: elle indique que le moniteur intercepte toute action interne ou externe d'un processus monitoré $[\![P]\!]^N_M$ qui tente de quitter le domaine du moniteur.

4.5.1. Définition. $\overrightarrow{R_1} < \overrightarrow{R_2}$ si et seulement s'il existe $\overrightarrow{R_3}$ tel que $\overrightarrow{R_1} = \overrightarrow{R_2}\|\overrightarrow{R_3}$

4.5.2. *Proposition.*

1. $\sqsubseteq_M$ *est une relation reflexive et transitive.*
2. $\forall \overrightarrow{R_1}, \overrightarrow{R_2}$ *si* $\overrightarrow{R_1} < \overrightarrow{R_2}$ *alors* $\overrightarrow{R_1}\|P \sqsubseteq_M \overrightarrow{R_2}\|P$
3. $\forall \overrightarrow{R}$, $\overrightarrow{R}\|[P]^N_M \sqsubseteq_M \overrightarrow{R}\|P$
4. *Pour tout* $P, Q \in \mathcal{P}$ *et tout moniteur* M, *si* $P \approx Q$ *et* $P \sqsubseteq_M R$ *alors* $Q \sqsubseteq_M R$.

5. *Pour tout* $P, Q \in \mathcal{P}$ *si* $\overrightarrow{a} \| P \approx \overrightarrow{a} \| Q$ *alors* $P \approx Q$.

Démonstration. Ces propriétés découlent de la définition de $\sqsubseteq_M$ et la definition de la sémantique de $\mathsf{CMN}_{\| \, \|}$.

■

Nous pouvons alors conclure le théorème suivant :

4.5.3. ***Théorème.*** *Pour tout processus* $P \in \mathcal{P}$, *tout moniteur* M *et tout emplacement* N , *nous avons :*

(i) $[\![P]\!]_M^N \models M$

(ii) $[\![P]\!]_M^N \sqsubseteq_M P$

Démonstration.

(i) La preuve est évidente à partir des règles de la sémantique opérationnelle de $[\![\ ]\!]_M^N$ (tableau 4.7).

(ii) La preuve est directe par la proposition 4.5.2 (cas 3), pour $\overrightarrow{R} = 0$.

■

Comme l'affirme le théorème 4.5.3, l'opérateur de surveillance virtuel nous permet d'obtenir une solution pour le problème que nous essayons de résoudre. Dans ce qui suit, nous allons définir une fonction de transition qui permet de transformer notre opérateur de surveillance virtuel vers l'opérateur de surveillance réel. Intuitivement, à partir de la forme $[\![P]\!]_M^N$ nous désirons développer une fonction qui nous permet d'obtenir la forme $[P]_M^N$.

4.5.2 Fonction de transition (C)

La fonction de transition est définie par la définition 4.5.4. Cette fonction nous permet de concrétiser notre opérateur de surveillance virtuel. En d'autres mots, la forme virtuelle ajoutée sera transformée en réelle par le biais de la fonction de transition C.

4.5.4. Définition.[Fonction de transition] Étant donné un processus réseau R dans CMN de la forme $[P_1]_M^{N_1} \| \ldots \| [P_n]_M^{N_n}$, la fonction de transition est définie par :

$$\mathcal{C}([\![P]\!]_M^N) = [P]_M^N$$

$$\mathcal{C}([\![P]\!]_M^N \| R) = \mathcal{C}([\![P]\!]_M^N) \| \mathcal{C}(R)$$

Dans ce qui suit, nous rappelons la définition de la bissimulation faible, définie par Milner [39].

4.5.5. Définition.[Bissimulation faible]

Une relation binaire $S \subseteq \mathcal{P} \times \mathcal{P}$ est une bissimulation faible, si pour tout couple $(P, Q) \in S$, et pour toute action $\alpha \in Act$:

1- $P \xrightarrow{\alpha} P'$ alors $\exists Q' \in \mathcal{P}$ tel que $Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ avec $(P', Q') \in S$.

2- $Q \xrightarrow{\alpha} Q'$ alors $\exists P' \in \mathcal{P}$ tel que $P \stackrel{\hat{\alpha}}{\Longrightarrow} P'$ avec $(P', Q') \in S$.

La plus grande bissimulation faible est notée par $\approx$:

4.5.6. Définition.[Bissimulation faible $\approx$] Deux processus P et Q sont dits faiblement bissimulaires, et nous écrivons $P \approx Q$, s'il existe une bissimulation faible S telle que $(P, Q) \in S$. Nous définissons $\approx$ comme étant la plus grande bissimulation de la manière suivante :

$$\approx = \bigcup \{S : S \; est \; une \; bissimulation \; faible\}$$

4.5.7. Définition.[Équivalence par test]

Soient deux processus P et Q appartenant à l'ensemble $\mathcal{P}$. Une relation binaire $T \subseteq \mathcal{P} \times \mathcal{P}$ est une équivalence par test, notée par $\sim$, ssi $(P, Q) \in T$ alors :

$$\forall \overrightarrow{R} : \overrightarrow{R} \| P \approx \overrightarrow{R} \| Q$$

En se basant sur la définition de la fonction de transition ($\mathcal{C}$), nous déduisons le théorème suivant :

4.5.8. *Théorème.*

$$\forall P : \mathcal{C}(P) = Q \text{ alors } P \sim Q$$

Démonstration. La preuve du théorème se déduit directement à partir des deux propositions 4.5.13 et 4.5.14 définies plus loin dans la présente section. ∎

Nous pouvons maintenant définir l'opérateur de renforcement $\otimes$ comme suit :

4.5.9. Définition. Étant donnés un processus réseau R et un moniteur M, nous définissons l'opérateur de renforcement $\otimes$ comme suit :

$$R \otimes M = \mathcal{C}([\![R]\!]_M^N)$$

Dans ce qui suit, nous énonçons le théorème principale qui montre la correction de l'opérateur de renforcement.

4.5.10. *Théorème.* *Soient un processus réseau R et un moniteur M, si $R' = R \otimes M$ alors :*

1. $R' \models M$
2. $R' \sqsubseteq_M R$

Démonstration.

1. Par la définition de l'opérateur de renforcement, nous avons $R' = \mathcal{C}([\![R]\!]_M^N)$. D'après les règles de la sémantique opérationnelle de $\mathsf{CMN}_{[\,]}$ (voir tableau 4.7), nous remarquons que le processus $[\![R]\!]_M^N$ transmet seulement les paquets qui sont autorisés par le moniteur M.
2. D'une part, par le théorème 4.5.8, nous pouvons conclure que $R' \approx [\![R]\!]_M^N$. D'autre part, d'après le théorème 4.5.3, nous avons que $[\![R]\!]_M^N \sqsubseteq_M R$. Par la proposition 4.5.2 (cas 4) nous déduisons que $R' \sqsubseteq_M R$.

∎

Dans le reste de cette section, nous présentons la congruence de l'équivalence par test par rapport à l'opérateur de communication ainsi que l'opérateur de surveillance virtuel, ensuite nous présentons deux propositions qui montrent la correction de la fonction de transition qui permet de passer de la version de l'opérateur virtuel vers l'opérateur de surveillance réel.

4.5.11. *Proposition.* *L'équivalence par test, $\sim$, est une congruence pour $\|$.*

Démonstration. Il suffit de montrer que :

$$E = \{(P\|R, Q\|R) \ : P, Q, R \in \mathcal{P} \text{ et } P \sim Q\}$$

est une équivalence par test.

D'après la définition de l'équivalence par test, il suffit de prouver que

$$\forall \overrightarrow{T} \ : P \parallel R \parallel \overrightarrow{T} \ \approx \ Q \parallel R \parallel \overrightarrow{T}$$

Selon la sémantique de l'algèbre, l'évolution du processus $P \parallel R \parallel \overrightarrow{T}$ doit faire intervenir, soit la règle d'entrelacement $R_{\|}$, soit la règle de communication, $R_{|}$. Intuitivement, $P \parallel R \parallel \overrightarrow{T}$ peut évoluer par l'un des cas suivants :

- P évolue ;
- R évolue ;
- P et $\overrightarrow{T}$ communiquent ;
- R et $\overrightarrow{T}$ communiquent ;
- P et R communiquent ;

Dans ce qui suit nous allons prouver le résultat pour les cinq cas listés ci-dessus.

Cas 1 Entrelacement avec $P \xrightarrow{\alpha} P'$.
Puisque $P \sim Q$, on aura $Q \stackrel{\alpha}{\Longrightarrow} Q'$ pour un certain $Q' \in \mathcal{P}$ avec $P' \sim Q'$. Selon la règle d'entrelacement, $Q\|R\|\overrightarrow{T} \stackrel{\alpha}{\Longrightarrow} Q'\|R\|\overrightarrow{T}$ et $(P'\|R, Q'\|R) \in E$.

Cas 2 Entrelacement avec $R \xrightarrow{\alpha} R'$.
Alors $Q\|R\|\overrightarrow{T} \xrightarrow{\alpha} Q\|R'\|\overrightarrow{T}$, et $(P\|R', Q\|R') \in E$.

Cas 3 Entrelacement avec communication tel que $P\|\overrightarrow{T} \xrightarrow{c(a)} P'\|\overrightarrow{T'}$.
Puisque $P \sim Q$, alors $Q\|\overrightarrow{T} \xrightarrow{c(a)} Q'\|\overrightarrow{T'}$, avec $P' \sim Q'$. Selon la règle d'entrelacement, $Q\|R\|\overrightarrow{T} \xrightarrow{c(a)} Q'\|R\|\overrightarrow{T'}$ et $(P\|R', Q\|R') \in E$.

Cas 4 Entrelacement avec communication tel que $R\|\overrightarrow{T} \xrightarrow{c(a)} R'\|\overrightarrow{T'}$.
Selon la règle d'entrelacement, $P\|R\|\overrightarrow{T} \xrightarrow{c(a)} P\|R'\|\overrightarrow{T'}$, aussi $Q\|R\|\overrightarrow{T} \xrightarrow{c(a)} Q\|R'\|\overrightarrow{T'}$ et donc $(P\|R', Q\|R') \in E$.

Cas 5 Communication entre P et R. À son tour ce cas se divise en deux. En effet, afin que P et R puissent communiquer, il faut que P soit de la forme $\vec{a}\|P'$ et que R soit de la forme $\bar{a}.R'$, et inversement.
Cas 5.1 P est de la forme $\vec{a}\|P'$ et R est de la forme $\bar{a}.R'$. Selon la règle de la communication suivit par la règle d'entrelacement, $\vec{a}\|P'\|\bar{a}.R'\|\overrightarrow{T} \xrightarrow{c(a)} P'\|R'\|\overrightarrow{T}$. Puisque

$P \sim Q$, alors $Q \| R \| \overrightarrow{T} \xrightarrow{c(a)} Q' \| R' \| \overrightarrow{T}$. avec $P' \sim Q'$ et donc $(P' \| R', Q' \| R') \in E$.

Cas 5.2 On applique les règles de la même façon que le cas précédent.

■

4.5.12. ***Proposition.*** *(**opérateur de surveillance** $[\![\ ]\!]$) : l'équivalence par test est une relation de congruence pour l'opérateur "$[\![\ ]\!]$".*

Démonstration. Il suffit de prouver que l'ensemble suivant :

$$T = \{([\![P]\!]_M^N, [\![Q]\!]_M^N) : \ P, Q \in \mathcal{P} \ et \ P \sim Q\} \tag{4.1}$$

est une équivalence par test.

Pour procéder à la démonstration, on sépare les cas selon la règle de l'opérateur de surveillance virtuel impliquée dans l'évolution du processus $[\![P]\!]_M^N$. Selon la sémantique opérationnelle de l'opérateur de surveillance virtuel, nous avons sept cas :

Cas 1 Règle $R^1_{[\![\ ]\!]}$ avec $P \xrightarrow{c(a)} P'$ tels que $a \models M$ et $\bar{a} \models M$.
Puisque $P \sim Q$, alors $Q \xRightarrow{c(a)} Q'$ pour un certain $Q' \in \mathcal{P}$ avec $P' \sim Q'$. Comme par hypothèse $a \models M$ et $\bar{a} \models M$ alors $R^1_{[\![\ ]\!]}$ s'applique et $[\![Q]\!]_M^N \xRightarrow{c(a)} [\![Q']\!]_M^N$, ce qui implique que $([\![P']\!]_M^N, [\![Q']\!]_M^N) \in T$.

Cas 2 Règle $R^2_{[\![\ ]\!]}$ avec $P \xrightarrow{\tau} P'$.
Puisque $P \sim Q$, alors $Q \xRightarrow{\tau} Q'$ pour un certain $Q' \in \mathcal{P}$ avec $P' \sim Q'$. $R^2_{[\![\ ]\!]}$ s'applique et $[\![Q]\!]_M^N \xRightarrow{\tau} [\![Q']\!]_M^N$, ce qui implique que $([\![P']\!]_M^N, [\![Q']\!]_M^N) \in T$.

Cas 3 Règle $R^3_{[\![\ ]\!]}$ avec $P \xrightarrow{a} \vec{a} \| P'$.
Puisque $P \sim Q$, alors $Q \xRightarrow{a} \vec{a} \| Q'$ pour un certain $Q' \in \mathcal{P}$ avec $P' \sim Q'$. $R^3_{[\![\ ]\!]}$ s'applique et $[\![Q]\!]_M^N \xRightarrow{a} [\![\vec{a} \| Q']\!]_M^N$, d'après la définition de $\sim$, on conclut que $([\![\vec{a} \| P']\!]_M^N, [\![\vec{a} \| Q']\!]_M^N) \in T$.

Cas 4 Règle $R^4_{[\![\ ]\!]}$ par hypothèse $P \sim Q$ et par la proposition 4.5.11 $\vec{a} \| P \sim \vec{a} \| Q$. Soit $a \in O(N)$ et $a \models M$, $R^4_{[\![\ ]\!]}$ s'applique et on obtient $[\![\vec{a} \| P]\!]_M^N \xrightarrow{\tau} \vec{a} \| [\![P]\!]_M^N$. De même, $R^4_{[\![\ ]\!]}$ s'applique une deuxième fois et on a $[\![\vec{a} \| Q]\!]_M^N \xrightarrow{\tau} \vec{a} \| [\![Q]\!]_M^N$ et $\vec{a} \| [\![P]\!]_M^N \sim \vec{a} \| [\![Q]\!]_M^N$.

Cas 5 Règle $R^5_{[\,]}$ par hypothèse $P \sim Q$ et par la proposition 4.5.11 $\vec{a}\|P \sim \vec{a}\|Q$. Soit $a \not\models M$, $R^5_{[\,]}$ s'applique et on obtient $[\![\vec{a}\|P]\!]^N_M \xrightarrow{\tau} [\![P]\!]^N_M$. De même, $R^5_{[\,]}$ s'applique une deuxième fois et on a $[\![\vec{a}\|Q]\!]^N_M \xrightarrow{\tau} [\![Q]\!]^N_M$ ce qui se traduit par $([\![P]\!]^N_M, [\![Q]\!]^N_M) \in T$.

Cas 6 La règle $R^6_{[\,]}$ implique que l'action a a la permission de pénétrer le moniteur M, on a donc par hypothèse que $a \in I(N)$ et $a \models M$. La règle $R^6_{[\,]}$ s'applique deux fois et on obtient $\vec{a}\|[\![P]\!]^N_M \xrightarrow{\tau} [\![\vec{a}\|P]\!]^N_M$ et $\vec{a}\|[\![Q]\!]^N_M \xrightarrow{\tau} [\![\vec{a}\|Q]\!]^N_M$, et puisque $\vec{a}\|P \sim \vec{a}\|Q$ (proposition 4.5.11), on a donc $([\![\vec{a}\|P]\!]^N_M, [\![\vec{a}\|Q]\!]^N_M) \in T$.

Cas 7 Règle $R^7_{[\,]}$ dans ce cas, il s'agit de l'absorption de l'action a, la preuve est directe vu que, $([\![P]\!]^N_M, [\![Q]\!]^N_M)$ est déjà dans T.

■

4.5.13. ***Proposition.*** *(Égalité entre les deux moniteurs) :*

$$[[P]^N]\!]^N_M \sim [P]^N_M$$

Démonstration. D'une manière intuitive, la proposition ci-dessus affirme que si le moniteur virtuel contrôle une composante isolée (un réseau formé d'une seule composante) alors on peut le remplacer par un moniteur réel.

■

4.5.14. ***Proposition.*** *(Éclatement du moniteur) : Soient N_P et N_Q représentent respectivement l'ensemble des canaux externes des processus P et Q, on a :*

$$[\![P\|Q]\!]^N_M \sim [[\![P]\!]^{N_P}_M \|[\![Q]\!]^{N_Q}_M]^N$$

Démonstration. Intuitivement, cette proposition affirme que pour renforcer une politique de sécurité sur un réseau donné, il suffit de placer un moniteur sur chaque composante du réseau.

■

4.6 Optimisation

À partir des exemples précédents, nous remarquons que notre technique de renfoncement n'est pas efficace. En effet, nous ajoutons notre moniteur dans toutes les composantes du

réseau. Afin d'optimiser notre technique, dans ce qui suit nous définissons une fonction d'optimisation qui permet de simplifier un moniteur. En effet, nous pouvons affaiblir un moniteur associé à un processus P si ce moniteur contrôle des canaux qui ne sont ni des canaux de sortie ni des canaux d'entrée de P. De plus, il va sans dire que si nous avons un moniteur vide, nous pouvons tout simplement supprimer le moniteur en question.

Ci-dessous nous présentons quelques règles de réécriture qui permettent de simplifier un moniteur.

4.6.1. Définition.[Opérateur de simplification] Soit un moniteur M et un emplacement N, la version simplifiée de M par rapport à N, notée par $M_{\downarrow_N}$, est la forme normale de M modulo AC (associativité et commutativité) calculée à partir des règles de réécriture suivantes :

$$\begin{array}{lll}
\neg\top & \longrightarrow_N \ \bot & \\
\top \wedge M & \longrightarrow_N \ M & \\
\bot \wedge M & \longrightarrow_N \ \bot & \\
M \wedge \neg M & \longrightarrow_N \ \bot & \\
a(m) & \longrightarrow_N \ \top & \text{si } a \notin N \\
\overline{a}(m) & \longrightarrow_N \ \top & \text{si } \overline{a} \notin N
\end{array}$$

Notons que les règles de réécritures ci-dessus sont classiques et qu'il n'est pas difficile de prouver que la forme normale recherchée existe. Ce qui nous permet de définir la fonction d'optimisation suivante.

4.6.2. Définition.[Fonction d'optimisation] Étant donné un processus réseau R dans CMN de la forme $[P_1]^{N_1}_M \| \ldots \| [P_n]^{N_n}_M$, la fonction d'optimisation est définie par :

$$\mathcal{O}([P]^N_M) = [P]^N_{M_{\downarrow_N}}$$

$$\mathcal{O}([P]^N_M \| R) = \mathcal{O}([P]^N_M) \| \mathcal{O}(R)$$

4.6.3. ***Théorème.*** *Pour tout processus réseau R dans CMN de la forme $[P_1]^{N_1}_M \| \ldots \| [P_n]^{N_n}_M$, on : $\mathcal{O}(R) \sim R$*

Démonstration. La preuve est directe par le fait que l'équivalence de test est congruente par rapport à l'opérateur $\|$. ∎

Example Nous avons montré dans l'exemple de la section 4.4.1 que l'application de l'opérateur de renforcement $\otimes$ donne le résultat suivant :

$$R_1 \otimes M \quad = \quad [\,[H_1]^{N_1}_{M} \| [H_2]^{N_2}_{M} \, \| [H_3]^{N_3}_{M} \| [H_4]^{N_4}_{M}]^{N}$$

Où la politique de sécurité M est égale à $M_1 \wedge M_2 \wedge M_3$, avec :

- $M_1 = \neg i_1(ip\_src = ip\_src(H_1) \wedge ip\_dst = ip\_dst(H_2) \wedge port\_src = 25)$.
- $M_2 = \neg i_6(ip\_src = ip\_src(H_3) \wedge ip\_dst = ip\_dst(H_2)$.
- $M_3 = \neg(ip\_src = ip\_src(H_3) \wedge port\_src = 80)$.

Dans ce qui suit nous appliquons la fonction d'optimisation sur $R_1 \otimes M$, ce qui nous donne :

$$\mathcal{O}(R_1 \otimes M) \quad = \quad [[H_1]^{N_1}_{M_1 \wedge M_3} \| [H_2]^{N_2}_{M} \, \| [H_3]^{N_3}_{M_2 \wedge M_3} \| [H_4]^{N_4}_{M_3}]^{N}$$

vu que :

$$\mathcal{O}([H_1]^{N_1}_{M}) = [H_1]^{N_1}_{M_1 \wedge M_3} \qquad \mathcal{O}([H_2]^{N_2}_{M}) = [H_2]^{N_2}_{M}$$

$$\mathcal{O}([H_3]^{N_3}_{M} = [H_3]^{N_3}_{M_2 \wedge M_3} \qquad \mathcal{O}([H_4]^{N_4}_{M}) = [H_4]^{N_4}_{M_3}$$

Comme nous l'avons déjà avancé, le problème que nous voulons résoudre possède en général plusieurs solutions. La plus simple est donnée par le l'opérateur $\otimes$ que nous avons défini précédemment. Une version améliorée de cette solution peut être obtenue après application de la fonction d'optimisation $\mathcal{O}$. La solution donnée par $\otimes$ consiste à placer le moniteur sur chaque nœud du réseau traité. Dans plusieurs situations de la vie courante, cette solution n'est pas intéressante. Afin de remédier à cet inconvénient, on peut essayer de trouver toutes les solutions possibles du problème. Cependant, il n'est pas toujours évident d'obtenir toutes les solutions possibles. En outre, donner toutes les solutions à l'utilisateur n'est pas nécessairement une bonne direction puisqu'il peut être confondu pour choisir la meilleur. Pour cette raison, nous trouvons intéressant le fait de donner à l'utilisateur la possibilité de fournir plus d'informations au sujet de la solution qu'il veut obtenir. Un exemple de ces informations pourrait être les nœuds sur lesquels il veut placer ses moniteurs (les noms des pare-feux dans le réseau, etc.). Ce genre d'informations, peut donner lieu à des solutions plus intéressantes et plus pratiques. La prochaine section formalise et résout ce problème.

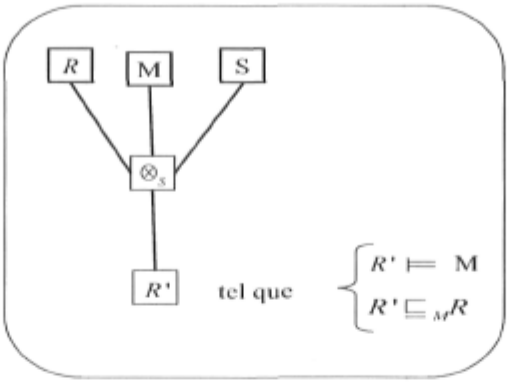


FIG. 4.4 – Opérateur de monitorage sélectif $\otimes_S$.

4.7 Monitorage sélectif

Comme l'indique la figure 4.4, le problème du monitorage sélectif consiste à trouver un nouvel opérateur $\otimes_S$, qui permet, comme $\otimes$, à partir d'un réseau R, d'une politique de sécurité (M) et d'un ensemble de composantes réseau S, de générer d'une manière automatique un nouveau réseau R' dans lequel les moniteurs sont placés seulement sur les éléments de S. Afin de simplifier la présentation, nous supposons qu'au départ le réseau analysé ne contient aucun moniteur. Ci-dessous une formalisation du problème que nous voulons résoudre :

4.7.1. ***Problème.****[Monitorage sélectif] Étant donnés un processus réseau R, un moniteur M, un ensemble de composantes sélectionnées S, nous désirons définir un opérateur $\otimes_S$, qui permet de générer, si possible, une nouvelle version configurée $R \otimes_S M$ qui respecte les propriétés suivantes :*

(i) $R \otimes_S M \models M$

(ii) $R \otimes_S M \sqsubseteq_M R$

(ii) Si $R \otimes_S M = [P]^N_{M'} \| R''$ *et* $[P]^N \notin S$ *alors* $M' = \top$

4.7.1 Approche

Afin de trouver la configuration du réseau qui respecte les propriétés listées ci-dessus, nous procédons comme suit :

- Dans un premier lieu, il faut définir un opérateur $\otimes$, qui respecte les conditions (i) et (ii) du problème 4.7.1. Il s'agit de l'opérateur de renforcement que nous avons défini dans les sections précédentes.
- En utilisant $\otimes$, nous définissons quelques règles de réécriture permettant de déplacer un moniteur d'une composante à une autre, de sorte que la propriété (iii) soit satisfaite.

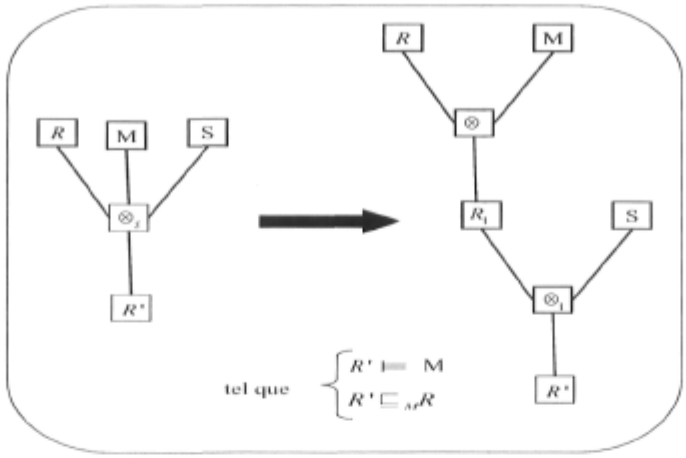


FIG. 4.5 – Rôle de l'opérateur de renforcement $\otimes_S$.

Comme indiquée dans la figure 4.5, l'opérateur $\otimes_1$ représente les règles de réécriture, et $R \otimes_S M = (R \otimes M) \otimes_1 S$.

D'une manière intuitive, afin que l'approche présentée fonctionne correctement, il faut s'assurer que l'opérateur $\otimes_1$ préserve les propriétés (i) et (ii) de $\otimes_S$. En effet, nous savons que l'équivalence par test, $\sim$, préserve ces propriétés. Par conséquent, pour prouver que $\otimes_1$ respecte les propriétés (i) et (ii) du problème, il suffit de prouver que :

$$\forall R, \forall S : \; R \otimes_1 S \sim R$$

.

Avant de définir $\otimes_1$ et prouver qu'il respecte les conditions désirées, nous avons besoin des définitions présentées dans la section suivante.

4.7.2 Notations et définitions

Pour simplifier la présentation, nous désignons, dans tout ce qui suit, par nœud une composante réseau. En d'autres termes un nœud est un processus ayant la forme $[P]^N_M$. Par ailleurs, si $\mathcal{N} = [P_1]^{N_1}_{M_1} || \ldots || [P_1]^{N_r}_{M_r}$, alors nous désignons par $\overline{\mathcal{N}}$ l'ensemble de ces nœuds, c.à.d., $\overline{\mathcal{N}} = \{[P_1]^{N_1}_{M_1}, \ldots, [P_1]^{N_r}_{M_r}\}$.

Dans ce qui suit, nous introduisons le concept de disque de rayon d et de centre η dans un réseau $\mathcal{N}$ en tant que l'ensemble de tous les nœuds qui peuvent être atteints à partir de η sur un chemin de longueur inférieur au égale à d (la longueur d'un chemin est le nombre de ses arcs).

4.7.2. Définition.[Disque] Soient $\mathcal{N}$ un réseau et η un nœud dans $\overline{\mathcal{N}}$. Un disque de centre η et de rayon d, dans $\overline{N}$, noté par $D^d_{\mathcal{N}}(\eta)$, est défini de la manière suivante :

$$\begin{array}{lcl} D^0_{\mathcal{N}}(\eta) & = & \{\eta\} \\ D^{d+1}_{\mathcal{N}}(\eta) & = & D^d_{\mathcal{N}}(\eta) \cup \\ & & \{\eta' \in \mathcal{N} \mid \exists \eta'' \in D^d_{\mathcal{N}}(\eta) \wedge (O((\eta'')) \cap I(\eta') \neq \emptyset \vee O(\eta') \cap I(\eta'') \neq \emptyset)\} \end{array}$$

Par ailleurs, nous introduisons la notion d'un disque limité par un ensemble de nœuds S comme suit :

4.7.3. Définition.[Disque limité par S] Soient $\mathcal{N}$ un réseau, η un nœud dans $\overline{\mathcal{N}}$ et S un ensemble de nœuds dans $\overline{\mathcal{N}}$. Un disque, dans $\overline{N}$, de centre η de rayon d est limité par S, notés par $D^d_{\mathcal{N}}(\eta, S)$, est défini de la manière suivante :

$$\begin{array}{lcl} D^0_{\mathcal{N}}(\eta, S) & = & \{\eta\} \\ D^{d+1}_{\mathcal{N}}(\eta) & = & D^d_{\mathcal{N}}(\eta, S) \cup \\ & & \{\eta' \in \mathcal{N} \mid \exists \eta'' \in (D^d_{\mathcal{N}}(\eta, S) - S) \\ & & \quad \wedge (O((\eta'')) \cap I(\eta') \neq \emptyset \vee O(\eta') \cap I(\eta'') \neq \emptyset)\} \end{array}$$

À partir de la définition d'un disque, nous pouvons retrouver la notion des voisins d'un nœud comme suit :

4.7.4. Définition.[Voisins d'un nœud] Soient $\mathcal{N}$ un réseau et η un nœud dans $\overline{\mathcal{N}}$. Les voisins de niveau i de η dans $\mathcal{N}$, noté par $V^i_{\mathcal{N}}(\eta)$, sont définis comme suit :

$$\begin{array}{lcl} V^0_{\mathcal{N}}(\eta) & = & \emptyset \\ V^{i+1}_{\mathcal{N}}(\eta) & = & D^{i+1}_{\mathcal{N}}(\eta) - D^i_{\mathcal{N}}(\eta) \end{array}$$

Nous pouvons également définir les voisins limités par S comme suit :

4.7.5. Définition.[Voisins limités par S] Soient $\mathcal{N}$ un réseau et η un nœud dans $\overline{\mathcal{N}}$. Les voisins de η dans $\mathcal{N}$ limités par S ayant un niveau i , noté par $V^i_{\mathcal{N}}(\eta, S)$, sont définis comme suit :

$$\begin{array}{lcl} V^0_{\mathcal{N}}(\eta, S) & = & \emptyset \\ V^{i+1}_{\mathcal{N}}(\eta, S) & = & D^{i+1}_{\mathcal{N}}(\eta, S) - (D^i_{\mathcal{N}}(\eta, S) - S) \end{array}$$

4.7.6. Définition.[Trafic contrôlé par M] Soient $\mathcal{N}$ un réseau, M un moniteur et η_1 et η_2 deux nœuds dans $\overline{\mathcal{N}}$. Nous disons que le trafic entre η_1 et η_2 est contrôlé par M et nous le notons par $\eta_1 \triangleright_M \eta_2$ si l'une des conditions suivantes est respectée :

– il existe m et $i \in O(\eta_1)$ tels que $\overline{i} \in I(\eta_2)$ et $(i(m) \not\models M$ ou $\overline{i}(m) \not\models M)$.

– il existe un message m tel que ($m.ip_{src} \in O(\eta_1) \wedge m.ip_{dst} \in I(\eta_2)$ et $m \not\models M$

Nous utilisons aussi les notations suivantes :

– $\eta_1 \lhd\rhd_M \eta_2$ pour dire que $\eta_1 \rhd_M \eta_2$ ou $\eta_2 \rhd_M \eta_1$.
– $S_1 \lhd\rhd_M S_2$ pour dire qu'ils existent $\eta_1 \in S_1$ et $\eta_2 \in S_2$ tels que $\eta_1 \lhd\rhd_M \eta_2$; avec S_1 et S_2 deux ensembles de nœuds.

4.7.7. Définition.[$S_1[S_2 \dagger M]$] Soient S_1 et S_2 deux ensembles de nœuds et M un moniteur. Nous notons par $S_1[S_2 \dagger M]$ l'ensemble définie de la manière suivante :

$$\begin{array}{rcll} (\emptyset)[S_2 \dagger M] & = & \emptyset & \\ (\{[P]^N_{M_0}\} \cup S_1)[S_2 \dagger M] & = & \{[P]^N_{M_0}\} \cup (S_1[S_2 \dagger M]) & \text{si } [P]^N_{M_0} \notin S_2 \\ (\{[P]^N_{M_0}\} \cup S_1)[S_2 \dagger M] & = & \{[P]^N_{M_0 \wedge M}\} \cup (S_1[S_2 \dagger M]) & \text{si } [P]^N_{M_0} \in S_2 \end{array}$$

4.7.3 Définition de $\otimes_1$

La définition de $\otimes_1$ est basée sur les deux règles du tableau 4.8 :

$$\otimes_1 = (R_1^*.R_2)^*$$

La notation $(R_1^*.R_2)^*$ exprime le fait qu'il faut appliquer la règle R_1 jusqu'à l'obtention d'un point fixe (R_1 ne s'applique plus), ensuite il faut appliquer la règle R_2 et répéter ces deux opérations jusqu'à l'obtention d'un point fixe global. Il est à noter que $\otimes_1$ peut échouer à placer les moniteurs sur les nœuds sélectionnés. Cependant, dans de tels cas, il n'existe pas de solution pour la configuration donnée par l'usager.

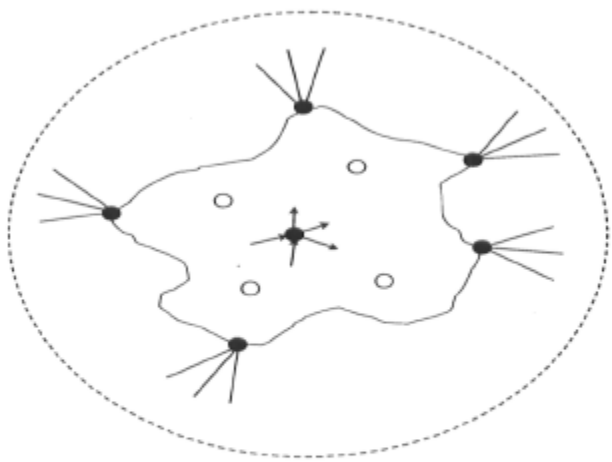


FIG. 4.6 – Déplacement de moniteurs.

D'une manière intuitive, le rôle des règles R_1 et R_2 est :

- R_1 permet de simplifier les moniteurs associés à chaque nœud du réseau analysé.
- R_2 permet de déplacer le moniteur associé à un nœud donné vers ses voisins dans S donnés par $V^{i+1}(\eta, S)$ (voisins de niveau i limités par S) si et seulement si le moniteur en question ne contrôle pas le trafic entre les nœuds de l'ensemble $(\neg((D^{i-1}(\eta, S) - S) \lhd\rhd_M (D^{i-1}(\eta, S) - S)))$. La figure 4.6 représente l'idée de la règle. Elle exprime le fait, que nous pouvons déplacer le moniteur du nœud noir au centre vers les nœud placé sur le bord, si et seulement si, ces derniers (nœuds noir sur le bord) sont tous dans S et qu'aucun paquet ne peut arriver au nœud du centre sans passer par les nœuds du bord. c'est ce qui explique le fait que sous ces conditions nous pouvons déplacer le moniteur d'une manière sécuritaire si ce dernier ne contrôle pas le trafic entre les nœuds limités par la ligne continue. Notons que, dans la section suivante, un exemple complet est présenté qui illustre l'application des règles.

$\mathbf{R}_1$	$\dfrac{< \overline{R} \cup \{[P]^N_M\},\ S >}{< \overline{R} \cup \{[P]^N_{M_{1_N}}\},\ S >}$	
$\mathbf{R}_2$	$\dfrac{< \overline{R} \cup \{[P]^N_M\},\ S >}{< \overline{R}[V^i(\eta, S) \dagger M] \cup \{[P]^N\},\ S >}$	$[P]^N_M \notin S$ et C

$$C = ((\exists i > 0 \mid V^i(\eta, S) \subseteq S) \ \wedge\ (\neg((D^{i-1}(\eta, S) - S) \lhd\rhd_M (D^{i-1}(\eta, S) - S)))$$

TAB. 4.8 – Règles de l'algorithme

4.7.4 Correction de $\otimes_S$

4.7.8. *Proposition.* *Étant donnés un réseau R exprimé en CMN et un ensemble de nœuds sélectionnés S, nous avons :*

$$R \otimes_1 S \sim R$$

Démonstration. Vu que $\otimes_1 = (R_1^*.R_2)^*$, il suffit alors de prouver que R_1 et R_2 préserve l'équivalence par test $\sim$.

- Il est clair que R_1 préserve $\sim$. En effet, R_1 n'est autre que la fonction d'optimisation définie dans 4.6.2.

- De même, l'équivalence par test est préservée par R_2. L'intuition de la preuve est captée par la figure 4.6. Notons que, la preuve ne sera pas détaillée dans ce travail.

∎

4.7.9. ***Théorème.*** *Étant donnés un réseau R exprimé en CMN, un moniteur M de L_M et un ensemble de composantes sélectionnées S. Si le problème 4.7.1 possède une solution, alors $N \otimes_S S$ en est une :*

(i) $R \otimes_S M \models M$

(ii) $R \otimes_S M \sqsubseteq_M R$

(ii) Si $R \otimes_S M = [P]^N_{M'} \| R''$ *et* $[P]^N \notin S$ *alors* $M' = \top$

Démonstration. Dans ce qui suit, nous présentons la preuve de chaque propriété :

(i) Nous savons, à partir de la section 4.4, que $R \otimes M \models M$. Par la proposition 4.7.8, nous pouvons conclure que $(R \otimes M) \otimes_1 S \sim R \otimes M$. Par la proposition 4.5.2 on a $(R \otimes M) \otimes_1 S \models M$, et par conséquent $R \otimes_S M \models M$

(ii) Nous savons, à partir de la section 4.4, que $R \otimes M \sqsubseteq_M M$. Par la proposition 4.7.8, nous pouvons conclure que $(R \otimes M) \otimes_1 S \sim R \otimes M$. Par la proposition 4.5.2 on a $(R \otimes M) \otimes_1 S \sqsubseteq_M M$, et par conséquent $R \otimes_S M \sqsubseteq_M M$

(iii) La preuve de cette propriété se déduit directement de la définition de la règle R_2. En effet, si l'algorithme termine et cette propriété n'est pas satisfaite, nous pouvons alors conclure que le problème n'a pas de solution pour la configuration donnée par l'usager.

∎

4.8 Étude de cas

Le réseau que nous essayons de sécuriser dans cette section est représenté par la figure 4.7.

La politique de sécurité que nous voulons renforcer est :

- H_1 ne peut pas envoyer à la machine H_7 sur le protocole icmp en utilisant l'interface i_8.
- H_4 ne peut pas envoyer à la machine H_5 sur le port 21.
- H_6 ne peut pas recevoir de la machine H_5 sur le port 25 en utilisant l'interface i_{15}.

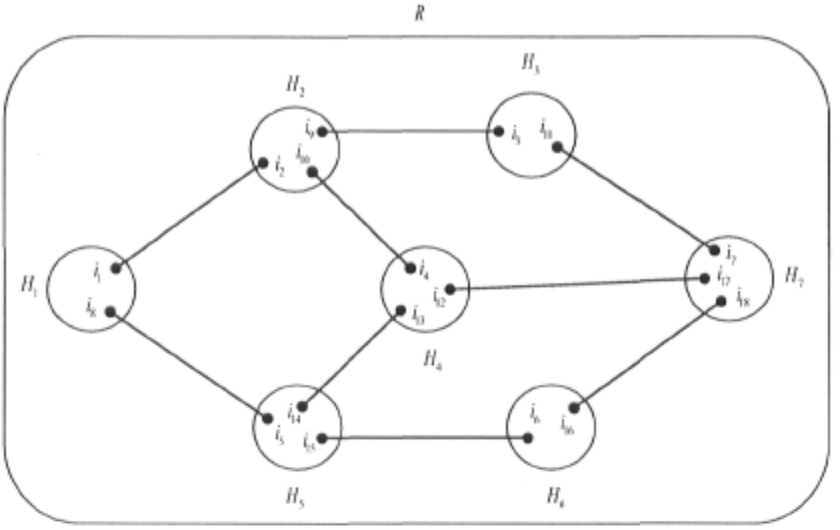


FIG. 4.7 – Un réseau spécifié en CMN.

Les composantes dans lesquels nous voulons ajouté le moniteur sont, H_2, H_4 et H_6. Afin d'atteindre notre objectif, dans ce qui suit nous appliquons toutes les étapes de l'approche présentée dans ce mémoire.

Spécification du réseau : En utilisant les mêmes idées et abréviations du premier exemple de ce chapitre, le réseau modélisé par la figure 4.7 est spécifié en CMN comme suit :

$$
\begin{aligned}
H_1 &\stackrel{def}{=} i_1.H_1 + \overline{i_2}.H_1 + i_8.H_1 + \overline{i_5}.H_1 \\
H_2 &\stackrel{def}{=} i_2.H_2 + \overline{i_1}.H_2 + i_9.H_2 + \overline{i_3}.H_2 + i_{10}.H_2 + \overline{i_4}.H_2 \\
H_3 &\stackrel{def}{=} i_3.H_3 + \overline{i_9}.H_3 + i_{11}.H_3 + \overline{i_7}.H_3 \\
H_4 &\stackrel{def}{=} i_4.H_4 + \overline{i_{10}}.H_4 + i_{12}.H_4 + \overline{i_{17}}.H_4 + i_{13}.H_4 + \overline{i_{14}}.H_4 \\
H_5 &\stackrel{def}{=} i_5.H_5 + \overline{i_8}.H_5 + i_{14}.H_5 + \overline{i_{13}}.H_5 + i_{15}.H_5 + \overline{i_6}.H_5 \\
H_6 &\stackrel{def}{=} i_6.H_6 + \overline{i_{15}}.H_6 + i_{16}.H_6 + \overline{i_{18}}.H_6 \\
H_7 &\stackrel{def}{=} i_7.H_7 + \overline{i_{11}}.H_7 + i_{17}.H_7 + \overline{i_{12}}.H_7 + i_{18}.H_7 + \overline{i_{16}}.H_7 \\
R &\stackrel{def}{=} [H_1]^{N_1} \| [H_2]^{N_2} \| [H_3]^{N_3} \| [H_4]^{N_4} \| [H_5]^{N_5} \| [H_6]^{N_6} \| [H_7]^{N_7}
\end{aligned}
$$

avec :

$$
\begin{array}{ll}
I(N_1) = \{\overline{i_2}, \overline{i_5}\} & O(N_1) = \{i_1, i_8\} \\
I(N_2) = \{\overline{i_1}, \overline{i_3}, \overline{i_4}\} & O(N_2) = \{i_2, i_9, i_{10}\} \\
I(N_3) = \{\overline{i_7}, \overline{i_9}\} & O(N_3) = \{i_3, i_{11}\} \\
I(N_4) = \{\overline{i_{10}}, \overline{i_{14}}, \overline{i_{17}}\} & O(N_4) = \{i_4, i_{12}, i_{13}\} \\
I(N_5) = \{\overline{i_6}, \overline{i_8}, \overline{i_{13}}\} & O(N_5) = \{i_5, i_{14}, i_{15}\} \\
I(N_6) = \{\overline{i_{15}}, \overline{i_{18}}\} & O(N_6) = \{i_6, i_{16}\} \\
I(N_7) = \{\overline{i_{11}}, \overline{i_{12}}, \overline{i_{16}}\} & O(N_7) = \{i_7, i_{17}, i_{18}\}
\end{array}
$$

Spécification de la politique de sécurité : La politique de sécurité est spécifiée en L_M par le moniteur M suivant :

$$M = M_1 \wedge M_2 \wedge M_3$$

avec

- $M_1 = \neg(ip\_src = ip\_src(H_1) \wedge ip\_dst = ip\_dst(H_7) \wedge prot = icmp)$.
- $M_2 = \neg(ip\_src = ip\_src(H_4) \wedge ip\_dst = ip\_dst(H_5) \wedge port\_src = 21)$.
- $M_3 = \neg i_{15}(ip\_src = ip\_src(H_5) \wedge ip\_dst = ip\_dst(H_6) \wedge port\_src = 25)$.

Calcul de $R_1 = R \otimes M$

$$R_1 = R \otimes M = [H_1]_M^{N_1} \| [H_2]_M^{N_2} \| [H_3]_M^{N_3} \| [H_4]_M^{N_4} \| [H_5]_M^{N_5} \| [H_6]_M^{N_6} \| [H_7]_M^{N_7}$$

Pour des raisons de simplification, dans le reste de cet exemple nous allons utiliser les notations suivantes :

$$R_1 = 1_M \| 2_M \| 3_M \| 4_M \| 5_M \| 6_M \| 7_M$$

avec $1 = [H_1]^{N_1}$, $2 = [H_2]^{N_2}$, ... et $7 = [H_7]^{N_7}$

Calcul de $R \otimes_S M = (R \otimes M) \otimes_1 S = R_1 \otimes_1 S$ **:** Nous devons appliquer les règles R_1 et R_2 selon la manière dictée par l'algorithme $(R_1^*.R_2)^*$ en utilisant la configuration $< \overline{R_1}, S >$.

Dans ce qui suit, le détail du calcul sera donné seulement pour la première étape, pour les autres étapes elle se font de la même manière que la première.

$$\begin{cases} S = \{2,4,6\} \\ \overline{R_2} = \{1_M, 2_M, 3_M, 4_M, 5_M, 6_M, 7_M\} \end{cases}$$

$\Longrightarrow$ {Appliquer R_1 sur chaque nœud}

$$\begin{cases} S = \{2,4,6\} \\ \overline{R_1} = \{1_{M_1 \wedge M_2}, 2_{M_1 \wedge M_2}, 3_{M_1 \wedge M_2}, 4_{M_1 \wedge M_2}, 5_M, 6_M, 7_{M_1 \wedge M_2}\} \end{cases}$$

$\Longrightarrow$ { Appliquer R_2 sur le nœud 1}

Dans un premier temps, il faut trouver un entier i satisfaisant les conditions suivantes :

$$V^{i+1}_{\mathcal{R}_\infty}(1_{M_1 \wedge M_2}, S) = D^{i+1}_{\mathcal{R}_\infty}(1_{M_1 \wedge M_2}, S) - D^{i}_{\mathcal{R}_\infty}(1_{M_1 \wedge M_2}, S)$$

Ceci peut se faire récursivement à partir de $i = 1$ jusqu'à, dans le pire cas, la valeur du diamètre du réseau (la distance maximale entre deux nœuds).

$$\begin{array}{lrcl} \text{Cas } i = 1 : & V^1_{\mathcal{R}_\infty}(1_{M_1 \wedge M_2}, S) & = & D^1_{\mathcal{R}_\infty}(1_{M_1 \wedge M_2}, S) - D^0_{\mathcal{R}_\infty}(1_{M_1 \wedge M_2}, S) \\ & & = & \{1,2,5\} - \{1\} = \{2,5\} \not\subseteq S \end{array}$$

$$\begin{array}{lrcl} \text{Cas } i = 2 : & V^2_{\mathcal{R}_\infty}(1_{M_1 \wedge M_2}, S) & = & D^2_{\mathcal{R}_\infty}(1_{M_1 \wedge M_2}, S) - D^1_{\mathcal{R}_\infty}(1_{M_1 \wedge M_2}, S) \\ & & = & \{1,2,4,5,6\} - \{1,5\} = \{2,4,6\} \subseteq S \end{array}$$

Ainsi, nous pouvons conclure que $i = 2$. Maintenant, nous devons vérifier que la deuxième condition de R_2 est satisfaite :

$$\neg((D^{i-1}_{\mathcal{R}_\infty}(\eta, S) - S) \lhd\rhd_M (D^{i-1}_{\mathcal{R}_\infty}(\eta, S) - S))$$

$$\begin{array}{rcl} D^1_{\mathcal{R}_\infty}(1_{M_1 \wedge M_2}, S) - S & = & \{1,2,5\} - \{2\} \\ & = & \{1,5\} \end{array}$$

La condition précédente est satisfaite du fait que $M_1 \wedge M_2$ ne contrôlent pas la communication interne entre H_1 et H_5.

Étant donné que les deux conditions de R_2 sont satisfaites, nous pouvons alors déplacer le moniteur du nœud 1 à ses voisins de niveau 2, ce qui donne :

$$\{1, 2_{M_1 \wedge M_2 \wedge M_1 \wedge M_2}, 3_{M_1 \wedge M_2}, 4_{M_1 \wedge M_2 \wedge M_1 \wedge M_2}, 5_M, 6_{M \wedge M_1 \wedge M_2}, 7_{M_1 \wedge M_2}\}$$

$$\begin{cases} S = \{2, 4, 6\} \\ \overline{R_1} = \{1, 2_{M_1 \wedge M_2 \wedge M_1 \wedge M_2}, 3_{M_1 \wedge M_2}, 4_{M_1 \wedge M_2 \wedge M_1 \wedge M_2}, 5_M, 6_{M \wedge M_1 \wedge M_2}, 7_{M_1 \wedge M_2}\} \end{cases}$$

$$\Longrightarrow \{R_1^*\}$$

$$\begin{cases} S = \{2, 4, 6\} \\ \overline{R_1} = \{1, 2_{M_1 \wedge M_2}, 3_{M_1 \wedge M_2}, 4_{M_1 \wedge M_2}, 5_M, 6_M, 7_{M_1 \wedge M_2}\} \end{cases}$$

$\Longrightarrow$ { Appliquer R_2 sur le nœud 3}

$$\begin{cases} S = \{2, 4, 6\} \\ \overline{R_1} = \{1, 2_{M_1 \wedge M_2 \wedge M_1 \wedge M_2}, 3, 4_{M_1 \wedge M_2 \wedge M_1 \wedge M_2}, 5_M, 6_{M \wedge M_1 \wedge M_2}, 7_{M_1 \wedge M_2}\} \end{cases}$$

$$\Longrightarrow \{R_1^*\}$$

$$\begin{cases} S = \{2, 4, 6\} \\ \overline{R_1} = \{1, 2_{M_1 \wedge M_2}, 3, 4_{M_1 \wedge M_2}, 5_M, 6_M, 7_{M_1 \wedge M_2}\} \end{cases}$$

$\Longrightarrow$ { Appliquer R_2 sur le nœud 5}

$$\begin{cases} S = \{2, 4, 6\} \\ \overline{R_1} = \{1, 2_{M_1 \wedge M_2 \wedge M}, 3, 4_{M_1 \wedge M_2 \wedge M}, 5, 6_{M \wedge M}, 7_{M_1 \wedge M_2}\} \end{cases}$$

$$\Longrightarrow \{R_1^*\}$$

$$\begin{cases} S = \{2,4,6\} \\ \overline{R_1} = \{1, 2_{M_1 \wedge M_2}, 3, 4_{M_1 \wedge M_2}, 5, 6_M, 7_{M_1 \wedge M_2}\} \end{cases}$$

$\Longrightarrow$ { Appliquer R_2 sur le nœud 7}

$$\begin{cases} S = \{2,4,6\} \\ \overline{R_1} = \{1, 2_{M_1 \wedge M_2 \wedge M_1 \wedge M_2}, 3, 4_{M_1 \wedge M_2 \wedge M_1 \wedge M_2}, 5, 6_{M \wedge M_1 \wedge M_2}, 7\} \end{cases}$$

$$\Longrightarrow \{R_1^*\}$$

$$\begin{cases} S = \{2,4,6\} \\ \overline{R_1} = \{1, 2_{M_1 \wedge M_2}, 3, 4_{M_1 \wedge M_2}, 5, 6_M, 7\} \end{cases}$$

Ce qui nous permet de conclure que la solution est :

$$R \otimes_S M = [H_1]^{N_1} \| [H_2]^{N_2}_{M_1 \wedge M_2} \| [H_3]^{N_3} \| [H_4]^{N_4}_{M_1 \wedge M_2} \| [H_5]^{N_5} \| [H_6]^{N_6}_{M} \| [H_7]^{N_7}$$

4.9 Conclusion

À travers ce chapitre, nous avons présenté deux langages formels L_M (logique temporelle) et CMN (algèbre de processus). Ces deux langages permettent respectivement de spécifier les politiques de sécurité et les réseaux informatiques. Ensuite, nous avons proposé un opérateur de renforcement qui à partir d'un réseau et d'une politique de sécurité, génère une nouvelle version du réseau qui d'une part satisfait la politique de sécurité et d'autre part est équivalent au réseau original. En outre, nous avons proposé un autre opérateur de renforcement qui a donné la possibilité de déplacer notre moniteur vers des composantes sélectionnés par l'utilisateur. Enfin, nous avons montré comment nous utilisions ces opérateurs de renforcement pour générer des réseaux qui respecte des politiques de sécurité données.

Chapitre 5

Conclusion et perspectives

Dans cette recherche, nous avons présenté une approche algébrique permettant de modéliser et de sécuriser formellement des réseaux informatiques. Cette approche consiste à développer une nouvelle algèbre pour la modélisation des réseaux et une logique pour la spécification des politiques de sécurité. Ensuite, nous avons défini un opérateur de renforcement qui permet, à partir d'un réseau et d'une politique de sécurité, de produire une nouvelle version du réseau sécuritaire et équivalente à l'originale.

Avant de détailler notre contribution dans le troisième chapitre, nous avons fait un survol de l'état de l'art dans le domaine de sécurité des réseaux. Cette partie est constitué de deux chapitres : le premier présente les deux techniques de sécurisation les plus connues (les systèmes de détection d'intrusions et les pare-feux) alors que le deuxième présente les trois algèbres de processus les plus répondues (CCS, π-calcul et le calcul ambient) et deux logiques temporelles (LTL et CTL).

Dans le premier chapitre, nous avons commencé par présenter les systèmes de détection d'intrusions. Ces derniers se basent sur deux grandes approches : l'approche comportementale et l'approche par scénarios. La première consiste à analyser les comportements des utilisateurs par rapport à leurs comportements habituels : si un utilisateur dévie de son comportement habituel par un certain seuil, ce nouveau comportement sera considéré comme une tentative d'attaque. La deuxième approche utilise une base de signatures, chacune de ces signatures décrit une attaque bien déterminée. Ainsi, une attaque est détectée par la présence d'une signature dans une trace d'événements. Cependant, cette approche permet de détecter seulement les attaques connues dont elle possède déjà les signatures. La deuxième partie du premier chapitre a été consacrée à une deuxième technique de sécurisation appelée pare-feu. Cette technique protège le réseau en question des accès non autorisés provenant de l'extérieur.

Dans le troisième chapitre, nous avons exposé la partie principale du travail qui consiste en l'élaboration d'une technique formelle destinée à la sécurisation des réseaux informatiques. Pour ce faire, nous avons utilisé une logique propositionnelle : elle ne prend pas en considération la trace globale des paquets, mais elle traite chaque message d'une manière indépendante des autres. D'autre part, nous avons choisi l'algèbre de processus CCS que nous avons muni d'un nouvel opérateur. Ce dernier appelé opérateur de surveillance, ajoute au comportement d'un processus une composante pour définir l'ensemble de ses canaux externes et un moniteur qui contrôle ces canaux. Le moniteur a le pouvoir d'absorber certains paquets voulant traverser le processus (entrée ou sortie).

Après la définition de notre algèbre que nous avons appelé CMN, nous avons défini un opérateur de renforcement qui permet de configurer automatiquement, quand cela est possible, certaines machines d'un réseau de sorte que le tout soit forcé à respecter une politique de sécurité donnée. Finalement, nous avons défini un autre opérateur de renforcement qui permet à l'utilisateur de faire un monitorage sélectif : il peut choisir un ensemble des composantes dans lesquels il veut placer la surveillance.

Comme perspectives, nous proposons les quatre axes de recherche suivant :

1. Étendre la logique proposée en une logique temporelle comme LTL permettant de renforcer des propriétés plus intéressantes.
2. Étendre l'algèbre proposée pour étudier la mobilité des processus au lieu de celle des paquets. Ceci permettra au moniteur d'analyser le comportement de ces processus à la recherche de comportements indésirables (virus, etc.).
3. Implanter les opérateurs déjà définis afin d'automatiser l'approche.
4. Attacher le programme visé par l'étape précédente à un outil existant comme Nmap qui permet de tracer automatiquement la carte d'un réseau. À partir de cette carte, la version processus pourrait être extraire automatiquement et le renforcement sera alors appliqué sur ce dernier.

Bibliographie

[1] M. Mejri A. Lacasse and B. Ktari. Formal implementation of network security policies. In *Second annual conference on privacy security and trust*, pages 161–166. Wu Centre, University of New Brunswick, Fredericton, Privacy, security and trust 2004 conference, octobre 2004.

[2] H. Ben Abdallah. Graphical communicating shared resources : A language for the specification, refinement and analysis of real-time systems. Master's thesis, PhD thesis, University of Pennsylvania, May 1995.

[3] T. Acadmics. Source release 1.3.1 for unix users manual 2.1.0. Technical report, Tripwire Security Systems Inc., April 1999.

[4] B. Alpern and F.B. Schneider. Defining liveness. information processing letters. pages 181–185, 1985.

[5] D. Anderson, T. Frivold, A. Tamaru, and A. Valdes. Next-generation intrusion detection expert system (nides), software users manual, beta-update release. Technical Report SRI-CSL-95-07, Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025-3493, may 1994.

[6] D. Anderson, T. Frivold, A. Tamaru, and A. Valdes. NIDES : software users manual - beta-update release. dec 1994.

[7] J. Anderson. Computer security threat monitoring and surveillance. Technical Report 56, Box 40 Fort Washington, pa. 19034, February 26, 1980.

[8] K. R. Apt and E. Olderog. *Verification of sequential and concurrent programs (2nd ed.)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[9] D. Austry and G. Boudol. Algèbre de processus et synchronisation. *Theoretical Computer Science*, 30 :91–131, 1984.

[10] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. Technical Report 60, 1984.

[11] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science.*, (37), 1985.

[12] P. Biondi. *Architecture expérimentale pour la détection d'intrusions dans un système informatique.* PhD thesis, ENST Bretagne, 2001.

[13] P.de Boer and M.Pels. Host-based intrusion detection systems. Technical report, February 4, 2005.

[14] E.M. Clarke and E.Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. *In Procedings of workshop on Logic of Programs. Spring erlag*, 1981.

[15] F. Cuppens and R. Ortalo. Lambda : A language to model a database for detection of attacks. In *RAID '00 : Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection*, pages 197–216, London, UK, 2000. Springer-Verlag.

[16] R. de Simone. Higher-level synchronising devices in meije-sccs. *Theoretical Computer Science*, (37) :245–267, 1985.

[17] D. Denning. Protection and defense of intrusion. *presented Presented at Conf. on National Security in the Information Age, US Air Force Academy*, Feb.1996.

[18] D.E. Denning. An intrusion-detection model. *IEEE Trans. Softw. Eng, Piscataway, NJ, USA*, 13(2) :222–232, 1987.

[19] S. Edwards. Network intrusion detection systems : Important ids network security vulnerabilities. Technical Report Technical Evangelist, September 2002.

[20] E.A. Emerson E.M. Clarke and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions On Programming Languages And Systems*, 8 :244–263, April 1983.

[21] E. Allen Emerson. Temporal and modal logic. *MIT Press, Cambridge, MA, USA*, pages 995–1072, 1990.

[22] K.L. Fox and Al. A neural network approach towards intrusion detection. Technical report, Harris Corporation, 1990.

[23] C. Green and M. Roesch. Snort users manual 2.1.0. Technical report, the snort project, Décember 2003.

[24] D. Harel. Statecharts : A visual formalism for complex systems. *Science of Computer Programming*, 8 :231–274, 1987.

[25] M. Hennessy and T. Regan. A process algebra for timed systems. *Inf. Comput. Academic Press, Inc. Duluth, MN, USA*, 117(2) :221–239, 1995.

[26] C. A. R. Hoare. *Communicating sequential processes*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1985.

[27] J.H. Holland. *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[28] D. Isacoff E. Spafford D. Zamboni J. Balasubramaniyan, J. Garcia-Fernandex. An architecture for intrusion detection using autonomous agents. Technical Report COAST TR 98-05, COAST Laboratory, Juin 1998.

[29] K. Cutler J. Wack and J. Pole. Guidelines on firewalls and firewall policy. Technical Report NIST Special Publication 800-41, Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, January 2002.

[30] A. EL-KABBAL K. ADI and L. PENE. Distributed firewall verification with mobile ambients. *Computer Security Research laboratory. Université du Québec en outaouais, Gatinaeau, QC, Canada*, 2005.

[31] K.I.Consulting K.Ingham and S.Forrest. A history and survey of network firewalls. *The University of New Mexico Computer Science Department Technical Report*, 2002.

[32] F. Kröger. *Temporal logic of programs*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[33] Mé L. et V. Alanou. Intrusion detection : A bibliography. Technical Report SSIR-2001-01, Supélec, Rennes, France, September 2001.

[34] C. Michel H. Débar L. Mé, Z. Marrakchi and F. Cuppens. La détection d'intrusion : les outils doivent coopérer. *Revue de l'Electricité et de l'Electronique*, (5), 2001.

[35] P. Lespérance. Détection des variations d'attaques à l'aide d'une logique temporelle. Master's thesis, Université Laval, 2006.

[36] U. Lindqvist and P.A. Porras. Detecting computer and network misuse through the production-based expert system toolset (p-BEST). In *IEEE Symposium on Security and Privacy*, pages 146–161, 1999.

[37] J. Loeckx, K. Sieber, and R. Stansifer. *The foundations of program verification*. John Wiley & Sons, Inc., New York, NY, USA, 1984.

[38] T. Lunt. Ides : An intelligent system for detecting intruders. *In Proceedings of the Symposium : Computer Security, Treat and Countermeasures, Rome, Italy*, November 1990.

[39] R. Milner. *Communication and concurrency*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1995.

[40] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts I. Technical Report 86, 1989.

[41] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts II. Technical Report 86, 1989.

[42] V. Natarajan. *Degrees of Delay : Semantic Theories for Priority, Efficiency, Fairness, and Predictability in Process Algebras*. PhD thesis, PhD thesis, North Carolina State University,Raleigh, NC, USA, August 1996.

[43] V. Natarajan, I. Christoff, L. Christoff, and R. Cleaveland. Priority and abstraction in process algebra. In *Proceedings of the 14th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 217–230, London, UK, 1994. Springer-Verlag.

[44] N.Boukhatem. Les agents mobiles et leurs applications. *DNAC Paris*, 1999.

[45] S. News. Inquiry board traces ariane 5 failure to overflow error. Number 9. Society for Industrial and Applied Mathematics, 1996.

[46] M. Nielsen, G. D. Plotkin, and G. Winskel. Petri nets, event structures and domains. In *Proceedings of the International Sympoisum on Semantics of Concurrent Computation*, pages 266–284, London, UK, 1979. Springer-Verlag.

[47] J. Parrow. *Handbook of Process Algebra*, chapter An introduction to the pi-calculus. Elsevier, 2001.

[48] S. Perret. Agents mobiles pour l'accès nomade à l'information répartie dans les réseaux de grande envergure. Master's thesis, Thèse en Sciences Informatique, LSR-IMAG, Grenoble I, France, 19 Novembre 1997.

[49] A. Pnueli and M. Shalev. What is in a step : On the semantics of statecharts. In Takayasu Ito and Albert R. Meyer, editors, *TACS*, volume 526 of *Lecture Notes in Computer Science*, pages 244–264. Springer, 1991.

[50] B. Fred. Schneider. Enforceable security policies. Technical report, Ithaca, NY, USA, 1998.

[51] D. Spinellis and D. Gritzalis. Panoptis : Intrusion detection using a domain-specific language, juin 2002.

[52] A. Sundaram. An introduction to intrusion detection. *Crossroads, ACM Press, New York, NY, USA*, 2(4) :3–7, 1996.

[53] G. Vigna. A topological characterization of tcp/ip security. *Dipartimento di Electtronica e Informazione Politecnico di Milano Piazza Leonardo da Vinci, 3220133 Milano, Italy*, December 19, 1996.

[54] J. E. White. Telescript technology, the foundation for the electronic marketplace. White paper, Genenral Magic, 1994.